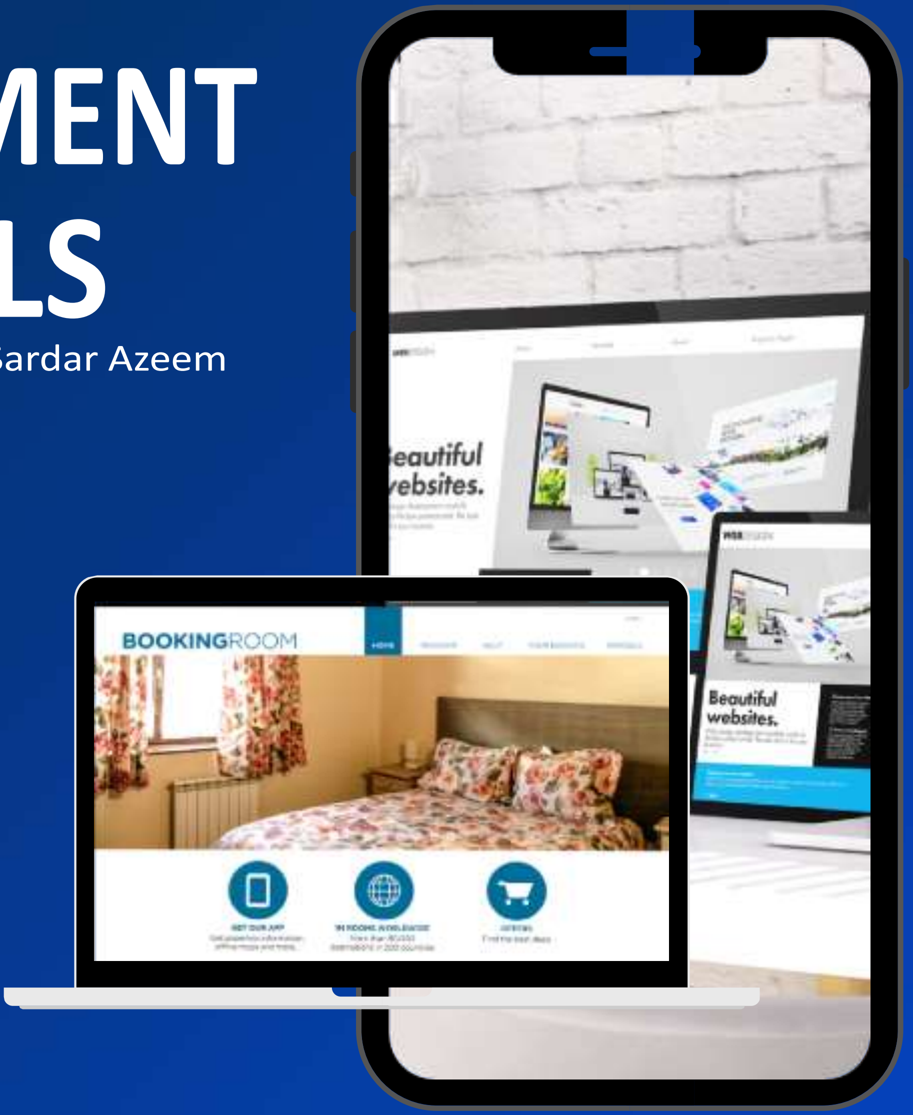
For DIT Part 2nd Students

WEBSITE DEVELOPMENT ESSENTIALS

Written and compiled By Sardar Azeem

- Creative Website Design
- Social Media Integration
- Analysis and Statistics
- Digital Strategy Consulting

0313-5879331



Visit Our Website
https://pictacademy.com

Course Outline

Section 1

WEBSITE DEVELOPMENT INTRODUCTION

Lectured By : Sardar Azeem

What is Internet?

The Internet is a global network of interconnected computers and devices that allows users to access and share information and devices, allowing them to communicate and exchange data. It enables users to access a wide range of services, such as websites, emails, social media, online applications, and cloud storage. Through standardized communication protocols, the Internet facilitates the sharing of information, resources and media across vast distances, making it an essential tool for education, business, entertainment and personal communication worldwide.

Key Features of the Internet:

1. Decentralized Network – No single entity owns or controls the entire Internet.
2. Uses TCP/IP Protocols – Ensures reliable data transmission between devices.
3. World Wide Web (WWW) – A major part of the Internet, consisting of websites and web apps.
4. Email, social media, Streaming, Cloud Services – Common uses of the Internet.
5. Accessible via ISPs – Users connect through Internet Service Providers (e.g., Comcast, AT&T).

The Internet has revolutionized modern life, making information and services available instantly worldwide

History of the Internet

The Internet came in the year 1960 with the creation of the first working model called ARPANET (Advanced Research Projects Agency). It allowed multiple computers to work on a single network which was their biggest achievement at that time. ARPANET uses packet switching to communicate multiple computer systems under a single network. In October 1969, using ARPANET first message was transferred from one computer to another. After that technology continues to grow.

How is the Internet Set Up?

The internet is set up with the help of physical optical fiber data transmission cables or copper wires and various other networking mediums like LAN, WAN, MAN, etc. For accessing the Internet even, the 2G, 3G and 4G services and the Wifi require these physical cable setups to access the Internet.

The ICANN (Internet Corporation for Assigned Names and Numbers), based in the USA, manages critical aspects of the Internet, including IP addresses, the Domain Name System (DNS) and protocols that ensure the smooth functioning and global connectivity of the Internet.

How Does the Internet Work?

- The actual working of the internet takes place with the help of clients and servers.
- Here, the client is a laptop that is directly connected to the internet.
- Servers are computers connected indirectly to the Internet and they store all the websites in those large computers.
- These servers are connected to the internet with the help of ISP (Internet Service Providers) and are identified with an IP address.
- Each website has its Domain Name, as it is difficult for a person to remember long numbers or strings (IP addresses).
- When you search for a domain name in the browser's search bar, the request is sent to the server.
- The server tries to find the IP address from the domain name because it cannot understand the domain name directly.
- After getting the IP address, the server will try to search the IP address of the domain name in a huge phone directory, known in networking as a DNS server (Domain Name Server).
- Example: Just like if we have the name of a person, we can easily find their Aadhaar number from a long directory — it works the same way with domain names and IP addresses.
- Once the IP address is found, the browser will pass on the request to the respective server.
- The server then processes the request and displays the content of the website that the client wants.
- If you are using a wireless medium of internet like 3G, 4G, or other mobile data, then data flows from optical cables to towers. From towers, signals reach your cell phones and PCs through electromagnetic waves.
- If you are using routers, then Optical fiber connects to your router and converts light-induced signals to electrical signals. Using ethernet cables, the internet reaches your computers, delivering the required information.

Difference Between World Wide Web and the Internet

World Wide Web	Internet
All the web pages and web documents are stored there on the World wide web and to find all that stuff you will have a specific URL for each website.	The Internet is a global network of computers that is accessed by the World wide web.
The world wide web is a service.	The Internet is an infrastructure.
The world wide web is a subset of the Internet.	The Internet is the superset of the world wide web.
The world wide web is software-oriented.	The Internet is hardware-oriented.
The world wide web uses HTTP.	The Internet uses IP Addresses.
The world wide web can be considered as a book from the different topics inside a Library.	The Internet can be considered a Library.
Examples: Websites, e-commerce, blogs	Example: The network connecting all online services

Uses of the Internet

- E-Commerce & Online Shopping: Platforms like Amazon, Flipkart and Myntra allow users to buy products with ease, providing a seamless shopping experience, home delivery and multiple payment options.
- Digital Payments & Cashless Transactions: UPI-based platforms (e.g., Paytm, Google Pay) enable users to make instant payments, bank transfers and bill payments, driving the growth of the cashless economy.
- Remote Education & Online Learning: Educational platforms like Coursera, Khan Academy and YouTube offer a wide range of online courses, making quality education accessible to people worldwide.
- Social Connectivity: Social media apps like Facebook, Instagram and LinkedIn allow people to stay connected, share content, network professionally and create online communities.
- Streaming & Entertainment: Services like Netflix, Spotify and YouTube provide on-demand access to movies, music, TV shows and online gaming, offering endless entertainment options.

- Telemedicine & Health Services: Online consultations, fitness apps and digital health tracking allow individuals to monitor their well-being and consult healthcare professionals remotely.
- Online Banking & Financial Services: Internet banking enables users to transfer funds, pay bills and manage finances without visiting a bank. Cryptocurrencies are also becoming more mainstream through online platforms.
- News & Information Access: The Internet serves as a hub for real-time news, articles, blogs and live updates from around the world, making it easier to stay informed on global events.
- Travel Planning & Navigation: Travel websites and navigation apps like Google Maps and Waze help users plan trips, book accommodations and navigate through traffic efficiently.
- Remote Work & Collaboration: Cloud-based tools such as Google Drive, Slack and Zoom enable teams to collaborate, store files and work remotely, improving productivity and communication across distances.

Security and the Internet

Very huge amount of data is managed across the Internet almost the time, which leads to the risk of data breaching and many other security issues. Both Hackers and Crackers can lead to disrupting the network and can steal important information like Login Credentials, Banking Credentials, etc.

Steps to Protect the Online Privacy

- Install Antivirus or Antimalware.
- Create random and difficult passwords, so that it becomes difficult to guess.
- Use a private browsing window or VPN for using the Internet.
- Try to use HTTPS only for better protection.
- Try to make your Social Media Account Private.
- If you are not using any application, which requires GPS, then you can turn GPS off.
- Do not simply close the tab, first log out from that account, then close the tab.
- Try to avoid accessing public WIFI or hotspots.
- Try to avoid opening or downloading content from unknown sources.

There is an element of the Internet called the Dark Web, which is not accessible from standard browsers. To keep safe our data, we can use Tor and I2P, which helps in keeping our data anonymous, that helps in protecting user security and helps in reducing cybercrime.

Social Impact of the Internet

The social impact of the Internet can be seen in both ways. Some say it has a positive impact as it helps in gaining civic engagement, etc. whereas some say it has a negative impact as it increased the risk of getting fooled by someone over the internet, getting withdrawal from society, etc.

Whatever the impact of social media, one thing is that it changed the way of connecting and interacting with others in society. The number of people increasing day by day on social media platforms which helps in constructing new relationships over social media, new communities are made on social media in the interest of the people. Social Media platforms like Facebook, Instagram, LinkedIn, etc are the most used social media platform for both individual and business purposes where we can communicate with them and perform our tasks.

Advantages of the Internet

- Online Banking and Transaction: The Internet allows us to transfer money online through the net banking system. Money can be credited or debited from one account to the other.
- Education, Online Jobs, Freelancing: Through the Internet, we are able to get more jobs via online platforms like Linkedin and to reach more job providers. Freelancing on the other hand has helped the youth to earn a side income and the best part is all this can be done via the INTERNET.
- Entertainment: There are numerous options for entertainment online we can listen to music, play games can watch movies and web series and listen to podcasts, YouTube itself is a hub of knowledge as well as entertainment.
- New Job Roles: The Internet has given us access to social media and digital products so we are having numerous new job opportunities like digital marketing and social media marketing online businesses are earning huge amounts of money just because the Internet is the medium to help us to do so.
- Best Communication Medium: The communication barrier has been removed from the Internet. You can send messages via email, Whatsapp and Facebook. Voice chatting and video conferencing are also available to help you to do important meetings online.
- Comfort to humans: Without putting any physical effort you can do so many things like shopping online it can be anything from stationeries to clothes, books to personal items, etc. You can books train and plane tickets online.
- GPS Tracking and google maps: Yet another advantage of the internet is that you are able to find any road in any direction and areas with less traffic with the help of GPS on your mobile.

Disadvantages of the Internet

- Time Wastage: Wasting too much time on the internet surfing social media apps and doing nothing decreases your productivity rather than wasting time on scrolling social media apps one should utilize that time in doing something skillful and even more productive.
- Bad Impacts on Health: Spending too much time on the internet causes bad impacts on your health physical body needs some outdoor games exercise and many more things. Looking at the screen for a longer duration causes serious impacts on the eyes.
- Cyber Crimes: Cyberbullying, spam, viruses, hacking and stealing data are some of the crimes which are on the verge these days. Your system which contains all the confidential data can be easily hacked by cybercriminals.
- Effects on Children: Small children are heavily addicted to the Internet watching movies and games all the time is not good for their overall personality as well as social development.
- Bullying and Spreading Negativity: The Internet has given a free tool in the form of social media apps to all those people who always try to spread negativity with very revolting and shameful messages and try to bully each other which is wrong.

Explain The Services of Internet

The Internet provides a wide range of services that enable communication, information sharing, entertainment, and business operations. Here are the key services of the Internet:

1. Communication Services

- Email (Gmail, Outlook) – Send and receive messages instantly.
- Instant Messaging (WhatsApp, Telegram, Signal) – Real-time text & voice chats.
- VoIP & Video Calls (Zoom, Skype, FaceTime) – Free or low-cost calls over the Internet.
- Social Media (Facebook, Instagram, Twitter/X, LinkedIn) – Connect, share, and network.

2. Information Services

- World Wide Web (WWW) – Access websites via browsers (Chrome, Firefox).
- Search Engines (Google, Bing, DuckDuckGo) – Find information quickly.
- Online Encyclopedias (Wikipedia) – Free knowledge resource.
- News & Blogs – Stay updated on global events.

3. Entertainment Services

- Video Streaming (YouTube, Netflix, TikTok) – Watch videos, movies, and shows.

- Music & Podcasts (Spotify, Apple Music, SoundCloud) – Stream songs and audio content.
- Online Gaming (Fortnite, PUBG, Roblox) – Multiplayer gaming with global players.

4. Business & E-Commerce Services

- Online Shopping (Amazon, eBay, Alibaba) – Buy and sell products globally.
- Digital Payments (PayPal, Venmo, UPI, Bitcoin) – Secure online transactions.
- Banking & FinTech (Online Banking, Robinhood, PayPal) – Manage money digitally.
- Cloud Computing (Google Drive, Dropbox, AWS) – Store and access files remotely.

5. Education & Work Services

- E-Learning (Coursera, Khan Academy, Udemy) – Online courses and certifications.
- Remote Work (Zoom, Slack, Microsoft Teams) – Virtual offices and meetings.
- Open-Source Knowledge (GitHub, Stack Overflow, arXiv) – Collaborate on projects.

6. Other Essential Services

- IoT (Smart Home, Wearables) – Internet-connected devices (Alexa, smartwatches).
- VPN & Cybersecurity – Protect privacy and bypass restrictions.
- Government & Public Services – Online tax filing, voting, and utilities.

World Wide Web (WWW)

The World Wide Web (WWW or "the Web") is a system of interconnected web pages and websites accessed via the Internet using web browsers (like Chrome, Firefox, or Safari). It was invented by Tim Berners-Lee in 1989 to share scientific documents but has since evolved into the primary way people access information online.

Key Features of the WWW

1. Web Pages & Websites
   - Built using HTML (HyperText Markup Language).
   - Accessed via URLs (e.g., https://www.google.com).
2. Hyperlinks (Clickable Links)
   - Allow navigation between pages (the "web" of connections).
3. Web Browsers
   - Software (Chrome, Firefox, Edge) that fetches & displays web pages.
4. Web Servers
   - Computers that store and deliver websites to users.

5. HTTP/HTTPS Protocols
    - Rules for transferring data securely between browsers and servers.

How the WWW Works

1. You type a URL (e.g., www.example.com) into a browser.
2. The browser uses DNS to convert the URL into an IP address.
3. A request is sent via HTTP/HTTPS to the web server.
4. The server sends back the HTML, CSS, and JavaScript files.
5. Your browser renders the page for you to view and interact with.

WWW vs. Internet – What's the Difference?

WWW (Web)	Internet
A service built on top of the Internet.	The global network of connected computers.
Uses HTTP/HTTPS to access websites.	Uses TCP/IP for all data transfer.
Includes websites, web apps, and browsers.	Includes email, FTP, gaming, IoT, and more.

Importance of the WWW

✓ Information Access (Google, Wikipedia)
✓ Communication (social media, email)
✓ E-Commerce (Amazon, online banking)
✓ Entertainment (YouTube, Netflix)
✓ Cloud Services (Google Drive, Zoom)

What is Email

Email (short for electronic mail) is a digital messaging system that allows users to send and receive text, files, images, and other data over the Internet. It works similarly to traditional mail but is instant, free (mostly), and accessible worldwide.

Key Features of Email

✓ Fast & Global – Delivers messages in seconds across the world.
✓ Attachments – Supports sending documents, photos, and videos.
✓ Stored Electronically – Messages are saved in an inbox (no paper).
✓ Accessible on Any Device – Works on phones, computers, and tablets.

How Email Works

1. Sender writes an email (e.g., in Gmail, Outlook).
2. Email is sent via SMTP (Simple Mail Transfer Protocol).
3. Recipient's email server (e.g., Yahoo, ProtonMail) receives it.

4. Recipient opens the email using POP3/IMAP protocols.

Common Email Providers

- Gmail (Google)
- Outlook (Microsoft)
- Yahoo Mail
- ProtonMail (Secure)
- iCloud Mail (Apple)

Email Address Format

📧 Example: username@domain.com

- Username (e.g., john.doe)
- @ symbol (separates name from domain)
- Domain (e.g., gmail.com, company.org)

Uses of Email

📨 Personal Communication (Friends, family)
💼 Business & Work (Official documents, meetings)
🛒 Online Services (Sign-ups, receipts, newsletters)
🔐 Security & Verification (Password resets, 2FA)

Advantages of Email

✅ Instant delivery (No delays like postal mail).
✅ Cost-effective (Most services are free).
✅ Eco-friendly (No paper waste).
✅ Organized (Folders, labels, spam filters).

Disadvantages of Email

❌ Spam & Phishing (Junk/scam emails).
❌ Security Risks (Hacking, data leaks).
❌ Overload (Too many emails can be overwhelming).

Email vs. Traditional Mail

Feature	Email	Traditional Mail
Speed	Seconds/minutes	Days/weeks
Cost	Free (mostly)	Paid (stamps)
Environment	Paperless	Uses paper
Accessibility	Anywhere (Internet needed)	Physical delivery

Social Networking

Social networking refers to the use of online platforms (social media) to connect with people, share content, and interact in virtual communities. These platforms allow users to create profiles, post updates, chat, and engage with others worldwide.

Key Features of Social Networking

1. User Profiles
   - Each person has a personal page (e.g., Facebook profile, Instagram bio).
   - Includes photos, bio, interests, and activity.
2. Friends/Followers System
   - Users can connect (Facebook friends) or follow (Twitter, Instagram).
   - Helps build a network of contacts.
3. Content Sharing
   - Text posts (Twitter/X, Facebook)
   - Photos & videos (Instagram, TikTok, Snapchat)
   - Live streaming (Facebook Live, YouTube Live)
4. Interaction Tools
   - Likes 👍, comments 💬, shares 🔄
   - Direct messaging (DMs) – Private chats (WhatsApp, Messenger)
   - Groups & Communities – People with shared interests (Facebook Groups, Reddit)
5. Algorithm-Based Feeds
   - Platforms show content based on user behavior (likes, searches).
   - Example: Instagram Reels, YouTube Shorts.

Popular Social Networking Platforms

Platform	Main Purpose	Key Features
Facebook	General social networking	Posts, Groups, Marketplace
Instagram	Photo/video sharing	Stories, Reels, IGTV
Twitter/X	Short text updates	Tweets, Threads, Trends
LinkedIn	Professional networking	Jobs, Resumes, B2B connections
TikTok	Short-form videos	Viral trends, Duets
Snapchat	Disappearing media	Snaps, Filters, Streaks
Reddit	Forum-based discussions	Subreddits, Upvotes/Downvotes

Uses of Social Networking

✓ Staying Connected – Chat with friends & family.
✓ News & Trends – Follow updates on global events.
✓ Business & Branding – Promote products/services (Facebook Ads, Influencers).
✓ Entertainment – Watch memes, live streams, and viral videos.
✓ Learning & Networking – LinkedIn for jobs, Twitter for industry news.

Advantages of Social Networking

☑ Global Reach – Connect with anyone, anywhere.
☑ Free Communication – No cost for basic use.

☑ Business Opportunities – Digital marketing, freelancing.
☑ Community Support – Find groups for hobbies, health, or causes.

Disadvantages of Social Networking

✕ Privacy Risks – Data leaks, hacking.
✕ Fake News & Misinformation – Spread of rumors.
✕ Addiction & Mental Health Issues – Excessive use leads to anxiety.
✕ Cyberbullying & Trolling – Harassment online.

Social Networking vs. Real-Life Networking

Aspect	Social Networking	Real-Life Networking
Speed	Instant connections	Takes time (meetings, events)
Reach	Global (billions of users)	Limited (local/face-to-face)
Depth	Often superficial	Stronger personal bonds
Record	Permanent (posts stay online)	Temporary (unless documented)

Mailing List

A mailing list is a collection of email addresses used to send messages (like newsletters, updates, or promotions) to multiple people at once. It's commonly used by businesses, organizations, and content creators to communicate with subscribers efficiently.

Types of Mailing Lists

1. Announcement List (One-Way Communication)
    - Used for sending updates (e.g., news, product launches).
    - Example: Company newsletters, blog updates.
2. Discussion List (Two-Way Communication)
    - Allows members to reply and engage in group conversations.
    - Example: Google Groups, Yahoo Groups (now defunct).
3. Opt-In vs. Opt-Out Lists
    - Opt-in – Users voluntarily subscribe (e.g., signing up on a website).
    - Opt-out – Users are added by default but can unsubscribe (less common, riskier).

How Mailing Lists Work

1. Subscription
    - Users sign up via a form (website, social media, event registration).
    - Example: "Subscribe to our newsletter!" pop-up.
2. Email Collection & Storage
    - Addresses are stored in a database (e.g., Mailchimp, Constant Contact).
3. Email Campaign Creation
    - A marketer writes an email (promotional, informational).
    - May include personalization (e.g., "Hi [Name]!").
4. Sending & Tracking
    - Emails are sent in bulk.
    - Tools track open rates, clicks, and unsubscribes.

Popular Mailing List Services

Service	Best For	Key Features
Mailchimp	Small businesses	Free tier, automation
Constant Contact	Email marketing	Templates, surveys
SendinBlue (Brevo)	Transactional emails	SMS + Email
ConvertKit	Creators & bloggers	Subscriber tagging
Google Groups	Discussion lists	Free, basic

Uses of Mailing Lists

✓ Marketing – Promote products/services.
✓ Newsletters – Share blog posts, industry news.
✓ Community Updates – Schools, nonprofits, clubs.
✓ Transactional Emails – Order confirmations, password resets.

Pros & Cons of Mailing Lists

✅ Advantages

- Cost-effective – Cheaper than SMS or postal mail.
- Targeted – Send personalized content.
- Automated – Schedule emails in advance.
- Measurable – Track opens, clicks, conversions.

❌ Disadvantages

- Spam Risks – If misused, emails go to junk folders.
- Unsubscribes – People may opt out if emails are irrelevant.
- Legal Compliance – Must follow laws like GDPR (EU) / CAN-SPAM (US).

Mailing List vs. Social Media

Feature	Mailing List	Social Media
Ownership	You control the data	Platform owns your reach
Delivery	Direct to inbox	Algorithm-dependent
Engagement	Higher conversion	More casual
Privacy	Email is private	Public/shared

Best Practices for Mailing Lists

◈ Get permission (use double opt-in).
◈ Segment lists (e.g., customers vs. leads).
◈ Avoid spammy content (follow CAN-SPAM rules).
◈ Test before sending (check formatting).

News Group

A newsgroup is an online discussion platform where users can post and read messages on specific topics, similar to a public bulletin board or forum. Newsgroups are part of the Usenet system, one of the oldest decentralized communication networks (even older than the World Wide Web!).

Key Features of Newsgroups

1. Topic-Based Discussions
   - Each newsgroup focuses on a specific subject (e.g., comp.software, sci.space).
   - Organized hierarchically (e.g., alt.fan.harry-potter).
2. Decentralized & Distributed
   - Messages are stored on multiple Usenet servers worldwide.
   - No single company controls them (unlike Facebook or Reddit).
3. Text-Based (Mostly)
   - Originally designed for text discussions, but some support binaries (files).
4. No Real-Time Chat
   - Works like an asynchronous forum (not live like Slack or Discord).

How Newsgroups Work

1. User Posts a Message
   - Sent to a Usenet server using NNTP (Network News Transfer Protocol).
2. Server Propagates the Message
   - The post gets shared across multiple servers globally.
3. Others Read & Reply
   - Users access the newsgroup via a newsreader client (e.g., Mozilla Thunderbird).

Types of Newsgroups

Category	Example	Description
comp.	comp.software	Computer-related topics
sci.	sci.astronomy	Science discussions
rec.	rec.games.chess	Recreational hobbies
soc.	soc.culture.india	Social/cultural topics
alt.	alt.music.beatles	Alternative/unofficial groups

Newsgroups vs. Modern Forums

Feature	Newsgroups (Usenet)	Modern Forums (Reddit, etc.)
Ownership	Decentralized (no single owner)	Company-controlled (e.g., Reddit Inc.)
Access	Requires a newsreader (e.g., Thunderbird)	Web-based (no special software)
Speed	Slower propagation	Instant posts

Content	Mostly text, some binaries	Rich media (images, videos, GIFs)
Popularity	Declined after the 2000s	Dominant (Reddit, Discord, Facebook Groups)

Uses of Newsgroups

✓ Technical Support – Experts help troubleshoot issues.
✓ Hobby Discussions – Share interests (e.g., movies, books).
✓ File Sharing – Some groups distribute binaries (e.g., alt.binaries.*).
✓ Historical Archive – Early internet discussions preserved.

Pros & Cons of Newsgroups

✅ Advantages

- No censorship (unlike corporate social media).
- Long-lasting threads (some date back decades!).
- Privacy-focused (no tracking like Facebook/Google).

❌ Disadvantages

- Steep learning curve (requires NNTP/newsreader setup).
- Spam & malware risks (unmoderated groups).
- Declining activity (most users moved to Reddit/forums).

How to Access Newsgroups Today

1. Get a Usenet Provider (e.g., News hosting, Giga news).
2. Use a Newsreader (e.g., Mozilla Thunderbird, Newsbin).
3. Subscribe to Groups (search for topics like comp.* or alt.*).

What is a web browser?

A web browser is a software application that lets you access and interact with content on the internet. When you type in a website address (like www.example.com) or click on a link, the browser fetches the information from a server and displays it on your device in a way that's easy to read and navigate.

Some popular web browsers are Google Chrome, Mozilla Firefox, Safari, Microsoft Edge, and Opera.

Browsers can show websites, play videos, run web apps, manage bookmarks, and even install extensions to add extra features.
Behind the scenes, they use technologies like HTML, CSS, and JavaScript to render web pages properly.

In short:
Without a web browser, you wouldn't be able to surf the web.

Functions of Web Browsers

Web browsers have a few major jobs:

1. Access Websites
   - They retrieve web pages from servers using web addresses (URLs).
2. Render Web Content
   - They display text, images, videos, and interactive elements by interpreting code (HTML, CSS, JavaScript).
3. Navigation
   - Browsers let you move between web pages (back, forward, refresh, home, bookmarks).
4. Security
   - They protect users from dangerous websites, block pop-ups, support private browsing, and handle SSL certificates (the "lock" icon 🔒).
5. Data Management
   - They store cookies, cache files, browsing history, and saved passwords to make web browsing faster and more convenient.
6. Extensions and Add-ons
   - They allow extra features to be added, like ad blockers, password managers, VPNs, etc.
7. Synchronization
   - Many browsers let you sync your bookmarks, passwords, and settings across devices.

Types of Web Browsers

Browsers can be categorized mainly by popularity or by their underlying technology (their "engine"). Here's a list:

Popular Web Browsers

- Google Chrome
  (Fast, popular, uses the Blink engine)
- Mozilla Firefox
  (Open-source, focuses on privacy, uses Gecko engine)
- Safari
  (Made by Apple, optimized for macOS and iOS, uses WebKit engine)
- Microsoft Edge
  (New versions are Chromium-based; old versions used EdgeHTML)

- Opera
  (Feature-rich with built-in VPN, ad blocker, based on Chromium/Blink)

Special/Alternative Web Browsers

- Brave
  (Privacy-focused, blocks ads and trackers automatically)
- Tor Browser
  (Built for anonymity and accessing the dark web)
- Vivaldi
  (Highly customizable, aimed at power users)
- DuckDuckGo Browser (Mobile-first)
  (Focuses heavily on privacy and blocking trackers)

What is a Web Server

A web server is basically a computer (or a program) that stores websites and delivers them to users over the internet.

When you type a web address like www.example.com into your browser, here's what happens:

- Your browser sends a request to the web server where that website is stored.
- The web server finds the right page (like index.html) and sends it back to your browser.
- Then your browser shows the website on your screen.

Simple way to remember:

Web browser = asks for a page
Web server = gives the page

Functions of a Web Server:

- Store website files (HTML, images, videos, CSS, JavaScript, etc.)
- Handle requests from browsers (like "Hey, give me the homepage!")
- Send responses back with the correct web pages
- Manage communication using protocols like HTTP or HTTPS
- Ensure security (with SSL certificates, access control, etc.)
- Log activity (tracking who accessed what and when)

Examples of Popular Web Servers:

Web Server	Notes
Apache HTTP Server	One of the oldest and most used servers
Nginx	Very fast and good at handling many users
Microsoft IIS	Integrated with Windows systems
LiteSpeed	High performance and lightweight
Google Web Server (GWS)	Used internally by Google

What are Web Directories?

A Web Directory is a collection of websites that are organized by categories and subcategories.
Instead of searching the whole internet like a search engine (e.g., Google), a web directory lets you browse through topics to find websites manually.

Think of it like a library:

- Instead of *searching* blindly, you *browse shelves* labeled Science, Art, Technology, etc.
- Inside each shelf, you find related books (or in this case, websites).

Functions of Web Directories

- Categorize Websites: Group similar websites together (like Education, Sports, Business).
- Help Discovery: Let users find websites by browsing topics, not just keywords.
- Provide Quality Listings: Often websites were manually reviewed before being added (better quality than random search results).
- Improve Website SEO: Websites listed in good directories could get better visibility.

Examples of Early Web Directories

Web Directory	Notes
Yahoo Directory	One of the first big web directories (closed in 2014)
DMOZ (Open Directory Project)	Massive open web directory (closed in 2017)
Best of the Web (BOTW)	Still running, curated directory
Business.com	Directory focused on business-related websites

Difference Between Web Directory and Search Engine

Feature	Web Directory	Search Engine
Organized by	Categories	Keywords and Algorithms
Website Addition	Manual (human reviewed)	Automatic Crawling (bots)
Example	Yahoo Directory, DMOZ	Google, Bing
Search Style	Browse categories	Enter keywords and search

Simple Diagram:

```
[Web Directory]
  ├─── Business
  │     ├─── Marketing Websites
  │     └─── Finance Websites
  ├─── Education
  │     ├─── Schools
  │     └─── Online Courses
  ├─── Technology
        ├─── Software Companies
        └─── Gadget Reviews
```

In Short:

A Web Directory is like a well-organized phone book 📖 for websites, while a Search Engine is like a fast detective 🕵 that finds websites based on your clues (keywords).

What is a website ?

A website is a collection of many web pages, and web pages are digital files that are written using HTML(Hypertext Markup Language). To make your website available to every person in the world, it must be stored or hosted on a computer connected to the Internet round a clock. Such computers are known as a Web Server.

The website's web pages are linked with hyperlinks and hypertext and share a common interface and design. The website might also contain some additional documents and files such as images, videos, or other digital assets.

With the Internet invading every sphere, we see websites for all kinds of causes and purposes. So, we can also say that a website can also be thought of as a digital environment capable of delivering information and solutions and promoting interaction between people, places, and things to support the goals of the organization it was created for.

Components of a Website

We know that a website is a collection of a webpages hosted on a web-server. These are the components for making a website.

- Webhost: Hosting is the location where the website is physically located. Group of webpages (linked webpages) licensed to be called a website only when the webpage is hosted on the webserver. The webserver is a set of files transmitted to user computers when they specify the website's address..
- Address: Address of a website also known as the URL of a website. When a user wants to open a website then they need to put the address or URL of the website into the web browser, and the asked website is delivered by the webserver.
- Homepage : Home page is a very common and important part of a webpage. It is the first webpage that appears when a visitor visits the website. The home page of a website is very important as it sets the look and feel of the website and directs viewers to the rest of the pages on the website.
- Design : It is the final and overall look and feel of the website that has a result of proper use and integration elements like navigation menus, graphics, layout, navigation menus etc.
- Content : Every web pages contained on the website together make up the content of the website. Good content on the webpages makes the website more effective and attractive.
- The Navigation Structure: The navigation structure of a website is the order of the pages, the collection of what links to what. Usually, it is held together by at least one navigation menu.

How to access Websites?

When we type a certain URL in a browser search bar, the browser requests the page from the Web server and the Web server returns the required web page and its content to the browser. Now, it differs from how the server returns the information required in the case of static and dynamic websites.

Types of Website

1. Static Website:

In Static Websites, Web pages are returned by the server which are prebuilt source code files built using simple languages such as HTML, CSS, or JavaScript. There is no processing of content on the server (according to the user) in Static Websites. Web pages are returned by the server with no change therefore, static Websites are fast. There is no interaction with databases. Also, they are less costly as the host does not need to support server-side processing with different languages.

2. Dynamic Website:

In Dynamic Websites, Web pages are returned by the server which is processed during runtime means they are not prebuilt web pages, but they are built during runtime according to the user's demand with the help of server-side scripting languages such as PHP, Node.js, ASP.NET and many more supported by the server. So, they are slower than static websites but updates and interaction with databases are possible. Dynamic Websites are used over Static Websites as updates can be done very easily as compared to static websites (Where altering in every page is required) but in Dynamic Websites, it is possible to do a common change once, and it will reflect in all the web pages.

3. Other Types Of Websites

There are different types of websites on the whole internet, we had chosen some most common categories to give you a brief idea –

- Blogs: These types of websites are managed by an individual or a small group of persons, they can cover any topics — they can give you fashion tips, music tips, travel tips, fitness tips. Nowadays professional blogging has become an external popular way of earning money online.
- E-commerce: These websites are well known as online shops. These websites allow us to make purchasing products and online payments for products and services. Stores can be handled as standalone websites.
- Portfolio: These types of websites acts as an extension of a freelancer resume. It provides a convenient way for potential clients to view your work while also allowing you to expand on your skills or services.
- Brochure: These types of websites are mainly used by small businesses, these types of websites act as a digital business card, and used to display contact information, and to advertise services, with just a few pages.
- News and Magazines: These websites needs less explanation, the main purpose of these types of websites is to keep their readers up-to-date from current affairs whereas magazines focus on the entertainment.
- Social Media: We all know about some famous social media websites like Facebook, Twitter, Reddit, and many more. These websites are usually created to let people share their thoughts, images, videos, and other useful components.
- Educational: Educational websites are quite simple to understand as their name itself explains it. These websites are designed to display information via audio or videos or images.
- Portal: These types of websites are used for internal purposes within the school, institute, or any business, These websites often contain a login process allowing students to access their credential information or allows employees to access their emails and alerts.

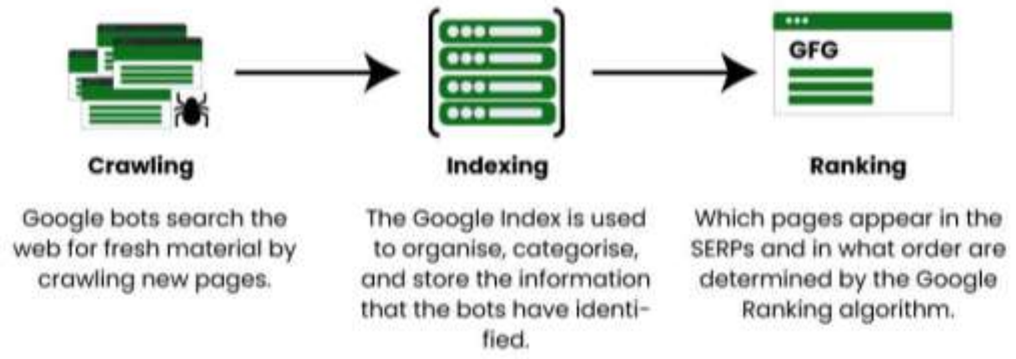
What are Search Engines?

A Search Engine is a software system that helps you find information on the internet by typing in keywords or questions.
It searches through millions of websites, picks the most relevant ones, and shows you a list of results — usually in seconds.

When you search something like "best pizza near me", the search engine looks into its massive database and brings you the best websites, maps, or reviews related to your request.

4. Main Functions of a Search Engine:

- Crawling
  (The search engine scans the internet to find new or updated web pages using bots called *crawlers* or *spiders*.)
- Indexing
  (It organizes and stores the information found into a huge database called the *index*.)
- Ranking/Search Results
  (When you search, it ranks the most relevant results based on things like keywords, quality, and popularity.)
- Serving Results
  (It displays the most useful links, images, videos, or news articles for your query.)


GFG
Crawling
Google bots search the web for fresh material by crawling new pages.
Indexing
The Google Index is used to organise, categorise, and store the information that the bots have identi-fied.
Ranking
Which pages appear in the SERPs and in what order are determined by the Google Ranking algorithm.

Popular Search Engines:

Search Engine	Special Feature
Google	Smartest and fastest, biggest database
Bing	Made by Microsoft, good with images
Yahoo! Search	Older player, now powered by Bing
DuckDuckGo	Focuses on privacy, no tracking

Baidu	Major search engine in China
Yandex	Popular search engine in Russia

Difference Between Search Engine and Web Directory

Feature	Search Engine	Web Directory
Search Method	Keyword-based	Category-based
Listing	Automatic (by bots)	Manual (human reviewed)
Examples	Google, Bing, DuckDuckGo	Yahoo Directory, DMOZ
Speed	Very fast	Slower browsing
Scale	Massive, millions of pages	Smaller, curated collection

In short:

A Search Engine is like a super smart librarian that finds exactly what you ask for — almost instantly!

Usage of Search Engine

Search engines have so many usages and some of them are:

- Searching for information: People use a search engine to search for any kind of information present on the internet. For example, Rohit wants to buy a mobile phone but he does not know which one is the best mobile phone. So he searches “best mobile phones in 2021” in the search engine and gets the list of best mobile phones along with their features, reviews, and prices.
- Searching images and videos: Search engines are also used to search images and videos. There are so many videos and images available on the internet in different categories like plants, animals, flowers, etc., you can search them according to your need.
- Searching location: Search engines are also used to find locations. For example, Seema is on a Goa trip but she doesn’t know the location of Palolem beach. So she searches “Palolem beach” on the search engine and then the search engine gives the best route to reach Palolem beach.
- Searching people: Search engines are also used to find people on the internet around the world.
- Shopping: Search engines are also used for shopping. Search engines optimize the pages to meet the needs of the user and give the lists of all the websites that contain the specified product according to the best price, reviews, free shipping, etc.
- Entertainment: Search engines are also used for entertainment purposes. It is used to search videos, movies, games, movie trailers, reviews of movies, social networking sites, etc. For example, Rohan wants to watch a movie named “Ram”,

then he searches this movie on a search engine and the search engine returns a list of links (of the websites) that contain the Ram movie.

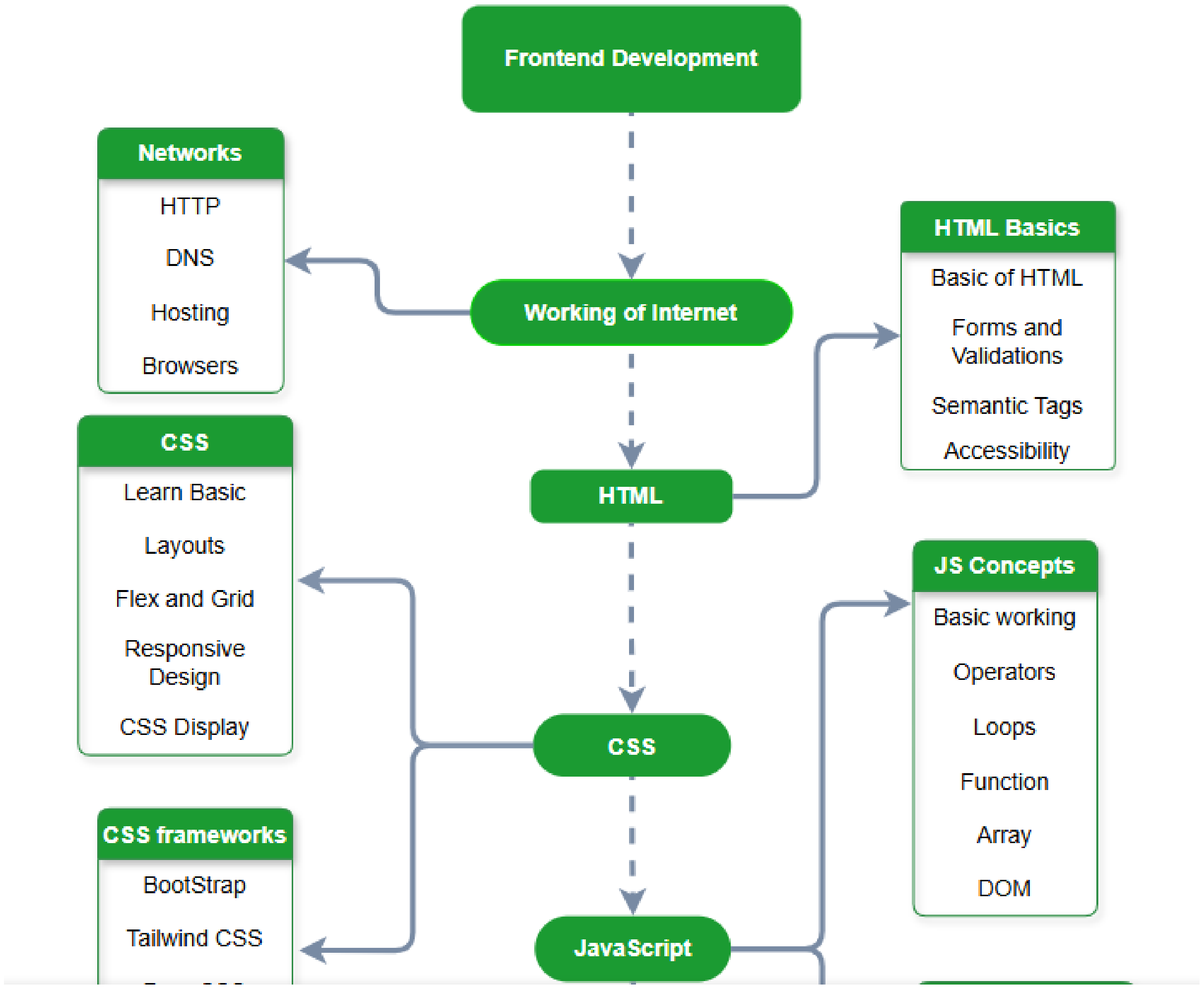
- Education: Search engines are also used for education. With the help of search engines, people can learn anything they wanted to learn like cooking, programming languages, home decorations, etc. It is like an open school where you can learn anything for free.

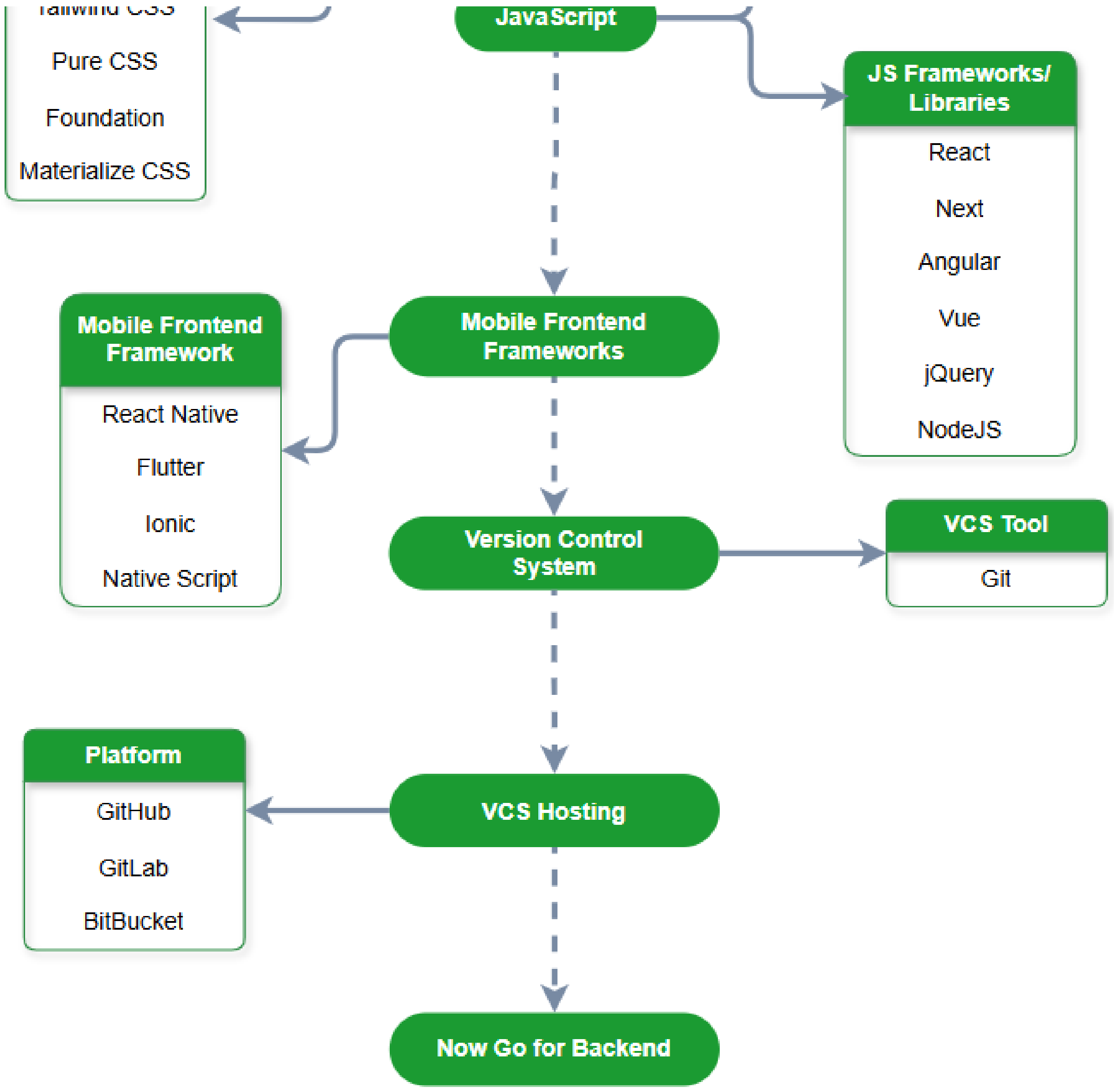
Web Page Program Development

Web development is the process of creating, building, and maintaining websites and web applications. It involves everything from web design to programming and database management. Web development is generally divided into three core areas: Frontend Development, Backend Development, and Full Stack Development.

Frontend Development

Frontend development refers to everything that users see and interact with on the website. It involves the design, structure, and layout of the website and is often referred to as the 'client side' of an application.





Frontend Technologies

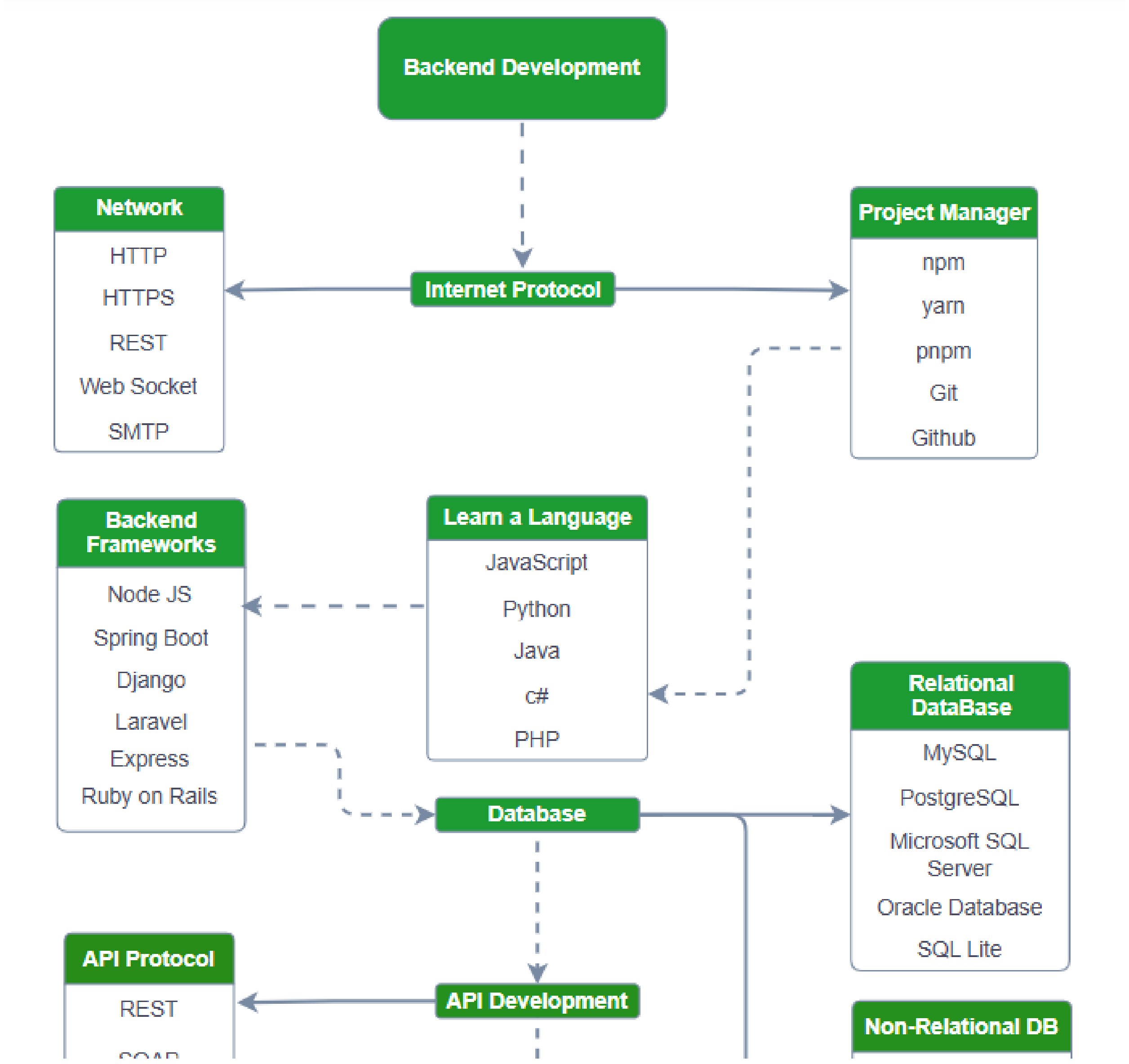
- HTML: HTML stands for HyperText Markup Language. It is the standard markup language used to create and design web pages, defining their structure and layout.
- CSS: Cascading Style Sheets fondly referred to as CSS is a simply designed language intended to simplify the process of making web pages presentable. It is used to style our website.
- JavaScript: JavaScript is a scripting language used to provide a dynamic behavior to our website.

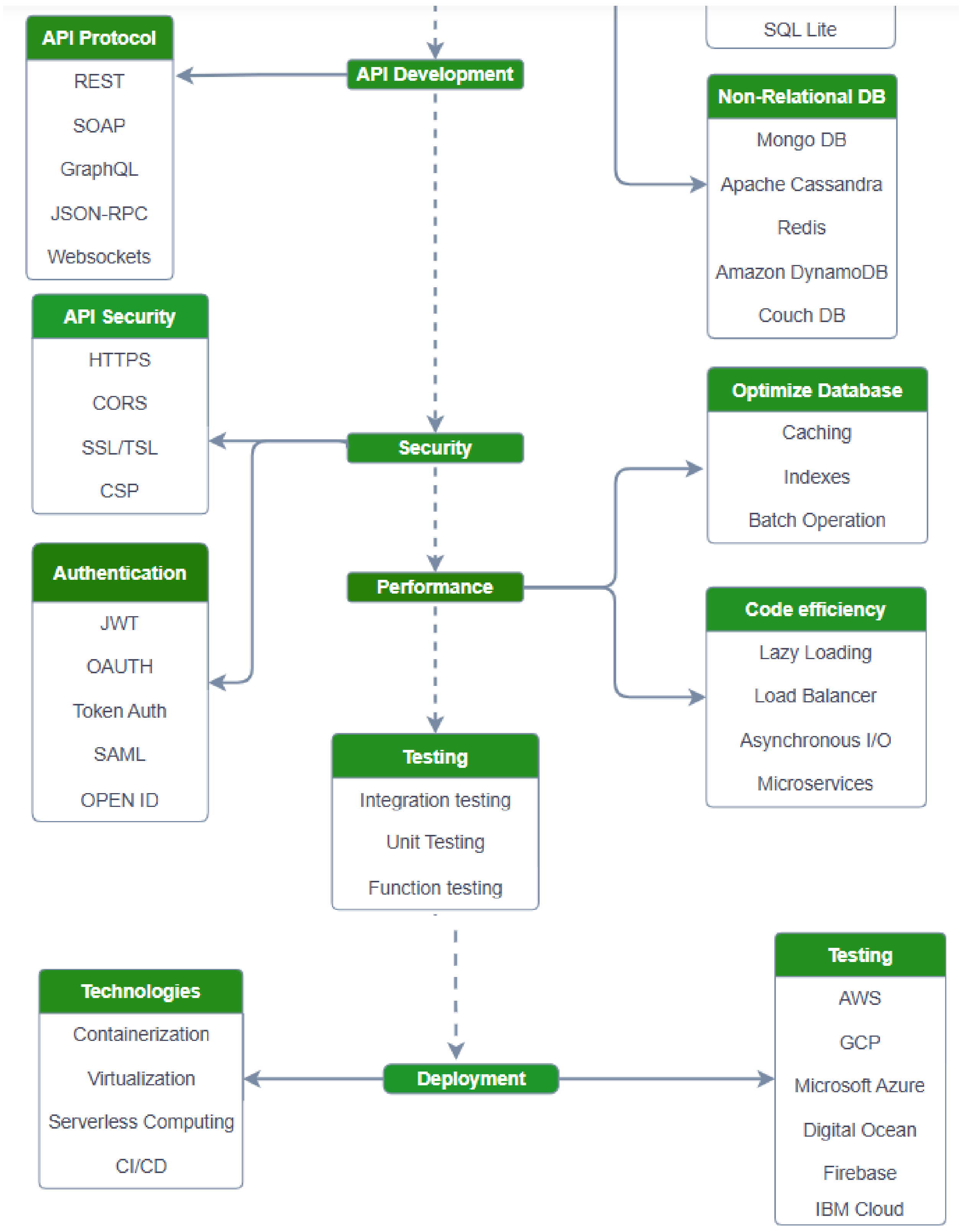
Frontend Frameworks/Libraries

- React.js : A popular JavaScript library for building dynamic, component-based user interfaces.
- Angular : A full-fledged framework for building single-page applications (SPAs), with features like two-way data binding and dependency injection.
- Vue.js : A progressive JavaScript framework that is flexible and can be used for building both simple and complex user interfaces.

Backend Development

Backend development refers to the server side of a website, where the logic and data are processed and stored. Users do not directly interact with this part, but it ensures that the website works properly.




SQL Lite
API Protocol
REST
SOAP
GraphQL
JSON-RPC
Websockets
API Development
Non-Relational DB
Mongo DB
Apache Cassandra
Redis
Amazon DynamoDB
Couch DB
API Security
HTTPS
CORS
SSL/TSL
CSP
Security
Optimize Database
Caching
Indexes
Batch Operation
Authentication
JWT
OAUTH
Token Auth
SAML
OPEN ID
Performance
Code efficiency
Lazy Loading
Load Balancer
Asynchronous I/O
Microservices
Testing
Integration testing
Unit Testing
Function testing
Testing
AWS
GCP
Microsoft Azure
Digital Ocean
Firebase
IBM Cloud
Technologies
Containerization
Virtualization
Serverless Computing
CI/CD
Deployment

Server-side Programming Languages and Frameworks

- PHP: PHP is a server-side scripting language designed specifically for web development.
- Java: Java is one of the most popular and widely used programming languages. It is highly scalable.
- Python: Python is a programming language that lets you work quickly and integrate systems more efficiently.
- Node.js: Node.js is an open source and cross-platform runtime environment for executing JavaScript code outside a browser.
- Ruby: Ruby is a dynamic, reflective, object-oriented, general-purpose programming language.
- C# : C# is a high-level, general-purpose programming language developed by Microsoft.

Backend Languages	Backend Frameworks
PHP	Laravel, Wordpress
Java	Spring, Hibernate
Python	Django, Flask, Python PIP
Node.js	Express
Ruby	Ruby on Rails
C#	.NET

Databases

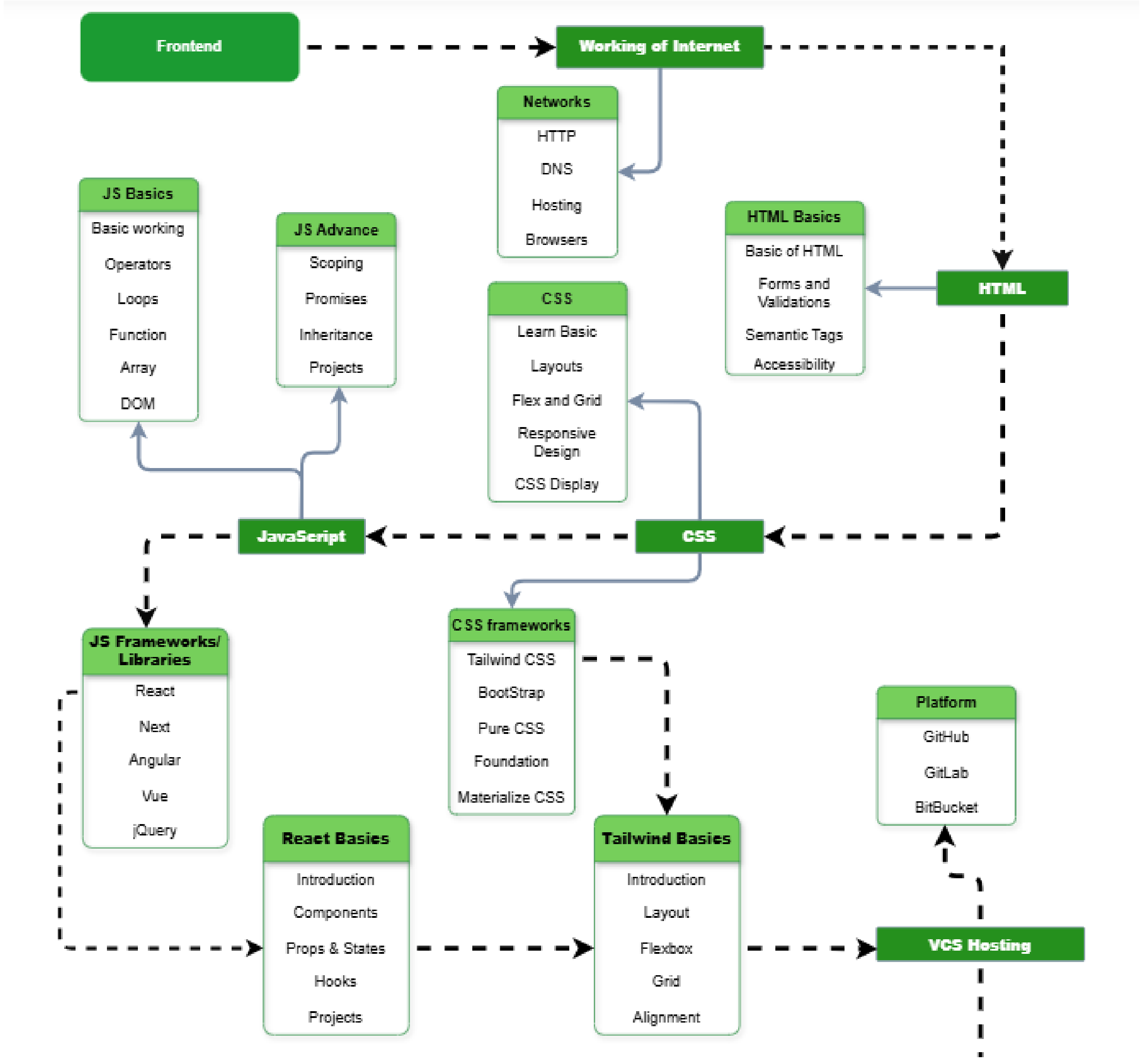
- MySQL
- PostgreSQL
- MongoDB
- MariaDB
- SQLite

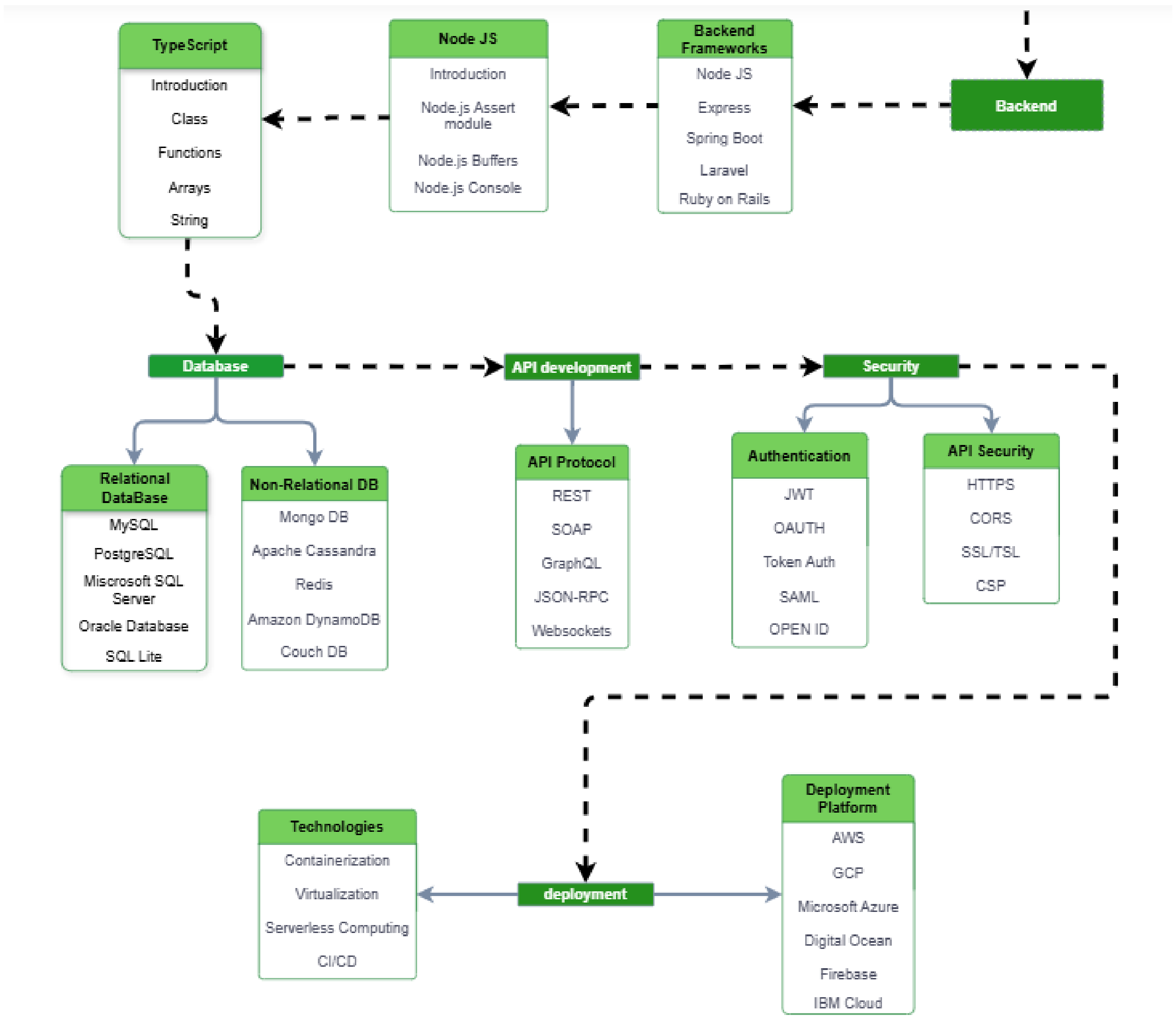
APIs (Application Programming Interfaces)

- RESTful API's
- GraphQL

Full Stack Development

Full-stack development refers to the practice of developing both the frontend and backend of a website or web application. Full-stack developers have a deep understanding of both areas and can build end-to-end solutions.





Full Stack Technologies:

- MERN Stack : MongoDB, Express.js, React, Node.js
- MEAN Stack : MongoDB, Express.js, Angular, Node.js
- JAMstack : JavaScript, APIs, Markup
- Django Stack : Django, MySQL/PostgreSQL, HTML/CSS/JavaScript
- Spring Boot Stack : Spring Boot, MySQL/PostgreSQL, Java
- LAMP Stack : Linux, Apache, MySQL, PHP
- LEMP Stack : Linux, Engine-X, MySQL, PHP

Databases

In web technology, a database is a structured collection of data that is stored electronically and accessed via a web application. It serves as the backend component where data is stored, managed, and retrieved. Databases can be relational (like MySQL, PostgreSQL) using structured tables and SQL for queries, or non-relational (like MongoDB, CouchDB) which store data in flexible, document-oriented formats. They enable web applications to handle dynamic content, user data, transactions, and more by providing efficient storage, retrieval, and manipulation capabilities. Database management systems (DBMS) are used to interact with the database, ensuring data integrity, security, and performance.

1. Relational Databases

A relational database stores data in tables, similar to a spreadsheet, where each table has rows and columns. The rows hold individual records, and the columns define the data attributes. Tables can be linked to each other through special keys, allowing related data to be connected.

- Postgre SQL : PostgreSQL is a powerful, open-source relational database that supports advanced SQL features and complex queries. It handles structured data, ensures ACID compliance, and is known for its reliability and extensibility.
- MariaDB : MariaDB is an open-source relational database that evolved from MySQL, offering improved performance, security, and features. It supports SQL queries, ACID compliance, and is highly compatible with MySQL.
- MySQL : MySQL is an open-source relational database management system that uses SQL for managing structured data. It's known for its reliability, ease of use, and performance, widely used in web applications.

2. NoSQL Databases

A NoSQL database stores data in a flexible, non-tabular format, unlike traditional relational databases. Instead of using tables with rows and columns, NoSQL databases might use documents, key-value pairs, wide-columns, or graphs to store data. This allows them to handle large amounts of unstructured or semi-structured data efficiently. They are designed to scale easily and manage big data applications.

- Mongodb : MongoDB is a NoSQL database storing data in JSON-like documents. It handles unstructured data, supports powerful queries, and scales easily across servers, making it popular for flexible, scalable applications.
- Cassandra : Apache Cassandra is an open-source NoSQL database that is used for handling big data. It has the capability to handle structure, semi-structured, and unstructured data.
- Redis : Redis is an in-memory NoSQL database known for its speed. It supports various data structures like strings, hashes, and lists, making it ideal for caching, real-time analytics, and messaging.

Main Roles in a Website Development Team

Role	What They Do
Project Manager	Plans the project, manages the team, communicates with the client.
Web Designer	Designs the layout, colors, fonts, and overall look of the website (UI/UX design).
Front-End Developer	Builds the visible parts of the website (what users see and interact with) using HTML, CSS, JavaScript.
Back-End Developer	Works on the server side — databases, APIs, authentication, logic.
Full-Stack Developer	Can handle both front-end and back-end tasks.
Content Writer	Creates the text (content) for the website — articles, product descriptions, blog posts.
Graphic Designer	Designs logos, banners, icons, images, and other visuals.
SEO Specialist	Optimizes the website to rank better on search engines like Google.
Quality Assurance (QA) Tester	Tests the website for bugs, broken links, security issues, and usability problems.
System Administrator (SysAdmin)	Manages the server where the website is hosted, handles deployment and maintenance.
Digital Marketing Specialist	Promotes the website through social media, email campaigns, ads, etc.

Simple Diagram of Workflow:

[Client]

↓

[Project Manager]

↓

[Web Designer] → (UI/UX Design)

↓

[Front-End Developer] → (Builds Interface)

↓

[Back-End Developer] → (Server + Database)

↓

[Content Writer] → (Adds Content)

↓

[SEO Specialist] → (Optimizes Search)

↓

[QA Tester] → (Checks Everything)

↓

[System Admin] → (Launches Website)

↓

[Digital Marketing] → (Promotes Website)

Quick Example:

Suppose you're building an online store:

- Designer makes the store look attractive 🛒
- Front-End Developer makes "Add to Cart" button work 🎯
- Back-End Developer makes sure products are saved in a database 📦
- Content Writer writes cool product descriptions ✍️
- SEO Specialist helps the store appear on Google search 🕵️
- Tester ensures customers don't face errors 🧪
- SysAdmin makes sure the server doesn't crash 🔧

In short:

Building a website is like making a movie — you need directors, actors, camera crew, editors... all working together!

Scope of Web Development

Web Development means building and maintaining websites.
Its scope (range of opportunities) is huge and keeps growing because everything is going online — businesses, education, shopping, banking, entertainment, even healthcare.

Major Areas Covered by Web Development:

Area	Examples
Front-End Development (Client-side)	Designing what users see: websites, mobile apps (HTML, CSS, JavaScript, React, etc.)
Back-End Development (Server-side)	Managing data, servers, databases (Node.js, PHP, Python, Java, etc.)

Full-Stack Development	Combination of Front-End + Back-End skills.
Web Design	Focus on how the website looks and feels (UI/UX).
E-commerce Development	Building online stores (like Amazon, Flipkart).
Web App Development	Making apps like Gmail, Instagram Web, Google Docs.
Content Management Systems (CMS)	Developing with platforms like WordPress, Joomla.
API Development and Integration	Building or connecting APIs (for apps to talk to each other).
Web Security	Protecting websites from hacking, malware, and data theft.
Progressive Web Apps (PWA)	Apps that work like mobile apps but inside a browser.

Why the Scope is Growing:

- Explosion of E-commerce
  (Every business needs an online shop.)
- Rise of Mobile Internet
  (Websites must work smoothly on phones/tablets.)
- Remote Work and Online Services
  (Web apps for online meetings, classes, banking.)
- Startup Culture
  (Startups need websites and apps to launch.)
- Government and Education Going Digital
  (Online public services, digital learning platforms.)
- Cloud Computing
  (Hosting websites and apps remotely is easier now.)

Job Opportunities in Web Development:

Position	Work
Web Developer	Build websites and web apps
UI/UX Designer	Design user-friendly interfaces
Full-Stack Developer	Handle both front-end and back-end
Web Project Manager	Lead web projects from start to finish
Web Security Specialist	Keep websites safe from attacks
CMS Developer	Build sites using WordPress, etc.

E-commerce Developer	Create online shopping sites

Technologies That Are Hot in Web Dev:

- HTML5, CSS3, JavaScript (basic building blocks)
- React, Angular, Vue.js (modern front-end frameworks)
- Node.js, Django, Laravel (back-end frameworks)
- WordPress, Shopify (popular platforms)
- APIs, Cloud hosting (AWS, Azure)

What are Scripting Languages?

A scripting language is a type of programming language that is mainly used to automate tasks, control other programs, or build small programs that don't need to be compiled first.

- Normal programming languages (like C++ or Java) often need to be compiled (translated into machine code first).
- Scripting languages are usually interpreted — they run line-by-line without compiling.

They are faster to write, easy to test, and great for web development, automation, games, and data processing.

Main Features of Scripting Languages:

Feature	Details
Interpreted	Code runs directly without a separate compilation step.
Easy to learn	Simple syntax and quicker to write programs.
Dynamic Typing	No need to declare the type of variables.
Automation	Great for automating repetitive tasks.
Embedded in applications	Can be used inside web pages, games, software (e.g., JavaScript in websites).

Examples of Scripting Languages:

Language	Used For
JavaScript	Web page interactivity (front-end and back-end with Node.js)
Python	Web apps, AI/ML, automation, scripting tasks
PHP	Server-side scripting for websites (like WordPress)
Ruby	Web development (Ruby on Rails framework)
Perl	Text processing, network programming
Bash	Automating tasks in Linux/Unix systems
Lua	Scripting in games and embedded systems

Where Scripting Languages Are Used:

- Web development (JavaScript, PHP)
- System administration (Bash, Python)
- Data analysis (Python, R)
- Game development (Lua in Roblox, Python in game engines)
- Task automation (Scripts for backups, file conversions, etc.)

What is JavaScript?

JavaScript (JS) is a programming language mainly used to make websites interactive.

When you visit a website:

- HTML structures the content 📄
- CSS styles it 🎨
- JavaScript makes it come alive with movements, popups, sliders, games, and more! ⚡

JavaScript can change web pages in real-time, respond to user actions, and even communicate with servers.

Main Features of JavaScript:

Feature	Details
Client-side scripting	Runs directly in your browser (like Chrome, Firefox).
Event-driven	Reacts to user actions like clicks, typing, scrolling.
Lightweight	Fast and easy to load.

Dynamic content	Can change parts of a webpage without reloading it.
Works with HTML/CSS	Enhances the basic website built with HTML and CSS.
Also used server-side	With Node.js, JavaScript now runs on servers too!

What JavaScript Can Do:

- Show pop-up messages (alerts)
- Validate forms (check if the user entered a valid email)
- Create animations and image sliders
- Build interactive maps, games, apps
- Load new content without refreshing (AJAX)
- Control audio and video players
- Create full web applications (like Gmail, Facebook)

Where JavaScript is Used:

Area	Example
Front-End Web Development	React.js, Vue.js, Angular
Back-End Development	Node.js
Mobile App Development	React Native
Game Development	Phaser, Babylon.js
Serverless Applications	AWS Lambda with JS

What is PHP?

PHP stands for "Hypertext Preprocessor" (yes, it's a tricky name because it's a *recursive acronym*).
It is a server-side scripting language mainly used to build dynamic websites and web applications.

- Server-side means: PHP code runs on the web server, not in your browser.
- Dynamic websites = Websites that change content automatically, like Facebook, WordPress, E-commerce stores.

✅ When you visit a page that ends with .php, it's likely running PHP behind the scenes!

Main Features of PHP:

Feature	Details
Server-side scripting	Code runs on the web server, not on your device.
Open-source	Free to use and modify.
Easy to learn	Simple syntax, especially for beginners.
Works with databases	Especially MySQL — perfect for storing user data.
Cross-platform	Runs on Windows, Linux, macOS servers.
Fast and efficient	Good for creating websites quickly.

What PHP Can Do:

- Create dynamic page content (like showing different data to different users)
- Handle forms (collect user inputs like login, signup)
- Manage sessions and cookies (like remembering if you're logged in)
- Interact with databases (store and fetch data, like posts, users, orders)
- Build entire CMS platforms (like WordPress, Joomla, Drupal)
- Create e-commerce sites (online stores like Shopify-like websites)

Famous Websites Built with PHP:

Website	Notes
Facebook	Originally built with PHP.
WordPress	The world's biggest CMS built on PHP.
Wikipedia	Written in PHP.
Slack (early version)	Started with PHP.

What are Web Hosting Services?

Web Hosting is a service that provides the space and technology needed to store and make your website available on the internet.

When you create a website (with HTML, CSS, PHP, etc.), it's just files.
To let the world see it, those files must be saved (hosted) on a special computer called a server that's connected to the internet 24/7.
That's exactly what a Web Hosting Service does!

✅ Without web hosting, your website would only live on your own computer — no one else could visit it.

Main Functions of Web Hosting Services:

Function	Details
Storage	Save your website files, images, videos, code.
Internet Connection	Keep your website online and accessible worldwide.
Security	Protect your site with firewalls, SSL certificates, backups.
Domain Support	Connect your website to your domain name (like www.example.com).
Technical Support	Help you if something goes wrong.

Popular Web Hosting Companies:

Company	Special Features
Bluehost	Beginner-friendly, good for WordPress.
HostGator	Affordable plans, reliable uptime.
GoDaddy	Hosting + domain services together.
SiteGround	Known for speed and customer service.
AWS (Amazon Web Services)	Powerful cloud hosting for big websites.

Types of Web Hosting:

Type	Explanation
Shared Hosting	Many websites share the same server. (Cheap, best for beginners)

VPS Hosting	Virtual private server — shared server, but more isolated. (Better performance)
Dedicated Hosting	You get an entire server just for your site. (Expensive, for big websites)
Cloud Hosting	Your website uses multiple servers (scalable, flexible, reliable).
Managed WordPress Hosting	Special hosting optimized for WordPress sites.

🍪 What Are Cookies (in Computers)?

In the world of websites, a cookie is a small piece of data that a website stores on your device (like your computer, phone, or tablet) to remember information about you.

✅ Cookies help websites "remember" you — your preferences, login info, or things you did on the site.

Main Purpose of Cookies:

Use	Example
Remembering logins	Stay signed in without typing your password again.
Tracking user activity	Knowing what products you looked at on a shopping site.
Personalizing content	Showing your name, favorite settings, or location.
Shopping carts	Remembering items you added to cart even if you leave the site.

How Cookies Work (Simple Steps):

1. You visit a website.
2. Website sends a cookie to your browser.
3. Your browser saves it on your device.
4. When you visit again, the browser sends the cookie back to the website.
5. The website uses that data to customize your experience.

Types of Cookies:

Type	Meaning
Session Cookies	Temporary, deleted when you close the browser.

Persistent Cookies	Stay on your device for a set time (like 1 week, 1 year).
First-party Cookies	Set by the website you are visiting.
Third-party Cookies	Set by other companies (like advertisers) to track you across different websites.

Simple Example:

When you log into Gmail and check "Remember me":

- Gmail stores a cookie in your browser.
- Next time you visit Gmail, it sees the cookie and keeps you logged in automatically.

Security Concerns Of Cookies:

- Cookies are NOT viruses or malware.
- But too many cookies can slow down your browser.
- Some cookies track your behavior for ads, which is why websites now ask your permission ("This site uses cookies" popups).

What is Web 2.0 and Web 3.0?

When we talk about Web 2.0 and Web 3.0, we're talking about different "generations" or "versions" of the internet — how it has changed over time.

First, Quick Timeline:

Version	Period	Main Idea
Web 1.0	1990s – early 2000s	"Read-only web" — Static websites, no user interaction.
Web 2.0	2004 – Now	"Read and write web" — Social media, user-generated content.
Web 3.0	Now – Future	"Read, write, and own web" — Decentralization, blockchain, AI-powered web.

What is Web 2.0? (The Current Internet)

Web 2.0 is the version of the internet where users are active participants — they create content, interact, and share information easily.

It's about social web, collaboration, sharing.

Main Features of Web 2.0:

Feature	Example
User-generated content	Users create blogs, videos, posts (YouTube, Instagram).
Social networking	Connect and interact (Facebook, Twitter).
Dynamic websites	Pages update instantly without reloading (AJAX).
Mobile and apps	Internet services on phones easily.
Cloud computing	Store data online (Google Drive, Dropbox).

✅ In Web 2.0, big companies like Google, Facebook, Amazon control a lot of the web.

What is Web 3.0? (The Next Generation)

Web 3.0 is about making the internet more intelligent, private, decentralized, and user-owned.

It's like a smarter, freer, more secure web where you have more control.

Main Features of Web 3.0:

Feature	Example
Decentralization	No single company controls the data (Blockchain, IPFS).
Ownership and control	Users own their own data, not corporations.
Smart Contracts	Programs that run automatically on blockchains (Ethereum).
Cryptocurrency Payments	Use Bitcoin, Ethereum for payments instead of banks.
Artificial Intelligence (AI)	Smarter websites that understand you (personalized search, recommendations).
Metaverse and Virtual Reality	New 3D worlds connected to Web 3.0.

✅ Web 3.0 is still growing — it's not everywhere yet, but it's the future vision of the internet.

Simple Comparison Table:

Feature	Web 2.0	Web 3.0
Ownership	Companies own data	Users own their data
Architecture	Centralized servers	Decentralized networks (blockchain)
Monetization	Ad-based economy (sell your data)	Token-based economy (earn crypto)
Examples	Facebook, YouTube, Twitter	Ethereum, OpenSea, Decentralized apps (dApps)
Technology	Social media, cloud computing	Blockchain, AI, crypto, VR/AR

In Short:

Web 2.0 = "You share your life, but companies control it."
Web 3.0 = "You control your life and data, and can earn from it."

Section 2

HTML 5.0 LANGUAGE WITH EXAMPLES

Lectured By : Sardar Azeem

What is HTML

HTML is an acronym which stands for Hyper Text Markup Language which is used for creating web pages and web applications. Let's see what is meant by Hypertext Markup Language, and Web page.

Hyper Text: Hyper Text simply means "Text within Text." A text has a link within it, is a hypertext. Whenever you click on a link which brings you to a new webpage, you have clicked on a hypertext. HyperText is a way to link two or more web pages (HTML documents) with each other.

Markup language: A markup language is a computer language that is used to apply layout and formatting conventions to a text document. Markup language makes text more interactive and dynamic. It can turn text into images, tables, links, etc.

Web Page: A web page is a document which is commonly written in HTML and translated by a web browser. A web page can be identified by entering an URL. A Web page can be of the static or dynamic type. With the help of HTML only, we can create static web pages.

Hence, HTML is a markup language which is used for creating attractive web pages with the help of styling, and which looks in a nice format on a web browser. An HTML document is made of many HTML tags and each HTML tag contains different content.

Example Program

```
<!DOCTYPE html>
<html>
<head>
<title>Web page title</title>
</head>
<body>
<h1>Write Your First Heading</h1>
<p>Write Your First Paragraph.</p>
</body>
</html>
```

Description of HTML Example

<!DOCTYPE>: It defines the document type or it instruct the browser about the version of HTML.

<html > :This tag informs the browser that it is an HTML document. Text between html tag describes the web document. It is a container for all other elements of HTML except <!DOCTYPE>

<head>: It should be the first element inside the <html> element, which contains the metadata(information about the document). It must be closed before the body tag opens.

<title>: As its name suggested, it is used to add title of that HTML page which appears at the top of the browser window. It must be placed inside the head tag and should close immediately. (Optional)

<body> : Text between body tag describes the body content of the page that is visible to the end user. This tag contains the main content of the HTML document.

<h1> : Text between <h1> tag describes the first level heading of the webpage.

<p> : Text between <p> tag describes the paragraph of the webpage.

Brief History of HTML

In the late 1980's , a physicist, Tim Berners-Lee who was a contractor at CERN, proposed a system for CERN researchers. In 1989, he wrote a memo proposing an internet based hypertext system.

Tim Berners-Lee is known as the father of HTML. The first available description of HTML was a document called "HTML Tags" proposed by Tim in late 1991. The latest version of HTML is HTML5, which we will learn later in this tutorial.

HTML Versions

Since the time HTML was invented there are lots of HTML versions in market, the brief introduction about the HTML version is given below:

HTML 1.0: The first version of HTML was 1.0, which was the barebones version of HTML language, and it was released in1991.

HTML 2.0: This was the next version which was released in 1995, and it was standard language version for website design. HTML 2.0 was able to support extra features such as form-based file upload, form elements such as text box, option button, etc.

HTML 3.2: HTML 3.2 version was published by W3C in early 1997. This version was capable of creating tables and providing support for extra options for form elements. It can also support a web page with complex mathematical equations. It became an official standard for any browser till January 1997. Today it is practically supported by most of the browsers.

HTML 4.01: HTML 4.01 version was released on December 1999, and it is a very stable version of HTML language. This version is the current official standard, and it provides added support for stylesheets (CSS) and scripting ability for various multimedia elements.

HTML5 : HTML5 is the newest version of HyperText Markup language. The first draft of this version was announced in January 2008. There are two major organizations one is W3C (World Wide Web Consortium), and another one is WHATWG(Web Hypertext Application Technology Working Group) which are involved in the development of HTML 5 version, and still, it is under development.

Features of HTML

1) It is a very easy and simple language. It can be easily understood and modified.

2) It is very easy to make an effective presentation with HTML because it has a lot of formatting tags.

3) It is a markup language, so it provides a flexible way to design web pages along with the text.

4) It facilitates programmers to add a link on the web pages (by html anchor tag), so it enhances the interest of browsing of the user.

5) It is platform-independent because it can be displayed on any platform like Windows, Linux, and Macintosh, etc.

6) It facilitates the programmer to add Graphics, Videos, and Sound to the web pages which makes it more attractive and interactive.

7) HTML is a case-insensitive language, which means we can use tags either in lower-case or upper-case.

HTML text Editors

- An HTML file is a text file, so to create an HTML file we can use any text editors.
- Text editors are the programs which allow editing in a written text, hence to create a web page we need to write our code in some text editor.
- There are various types of text editors available which you can directly download, but for a beginner, the best text editor is Notepad (Windows) or TextEdit (Mac).
- After learning the basics, you can easily use other professional text editors which are, Notepad++, Sublime Text, Vim,VSCode etc.
- We will use Notepad and sublime text editor. Following are some easy ways to create your first web page with Notepad, and sublime text.

HTML Editors

Notepad

Notepad++

VScode

Sublime text

HTML Execution

internet explorer

Microsoft Edge

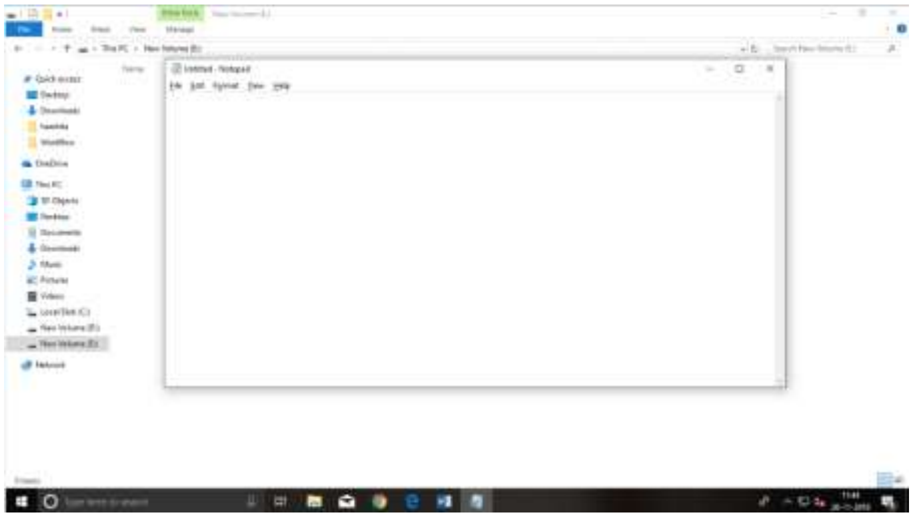
Google Chrome

Mozella Firefox

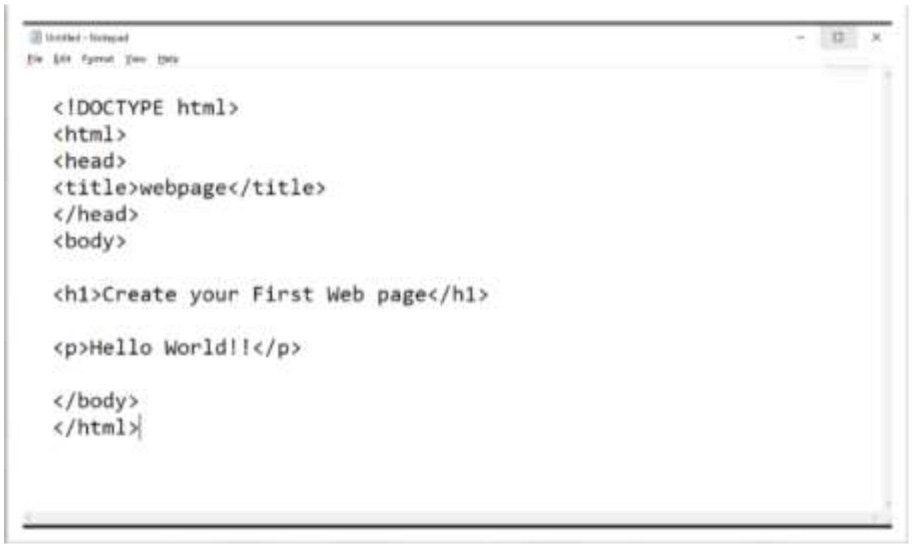
HTML code with Notepad. (Recommended for Beginners)

Notepad is a simple text editor and suitable for beginners to learn HTML. It is available in all versions of Windows, from where you easily access it.

Step 1: Open Notepad (Windows)

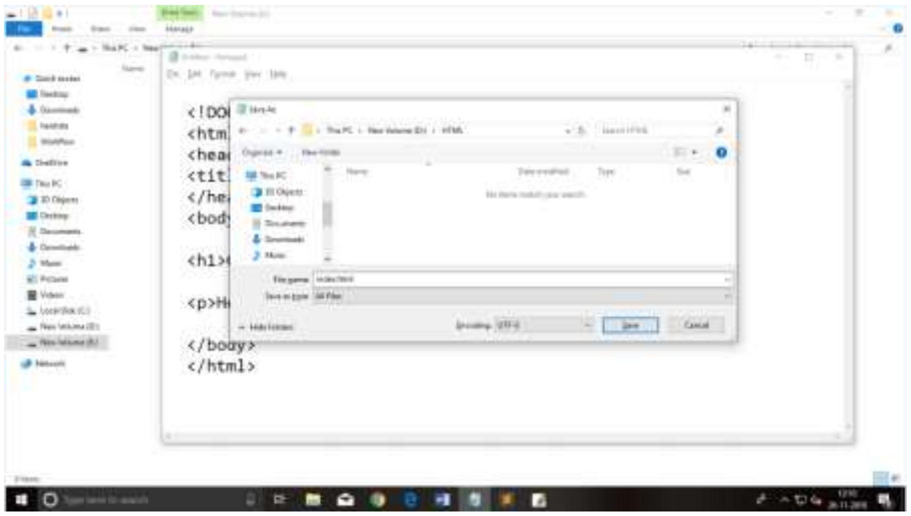


Step 2: Write code in HTML



```
<!DOCTYPE html>
<html>
<head>
<title>webpage</title>
</head>
<body>

<h1>Create your First Web page</h1>

<p>Hello World!!</p>

</body>
</html>
```

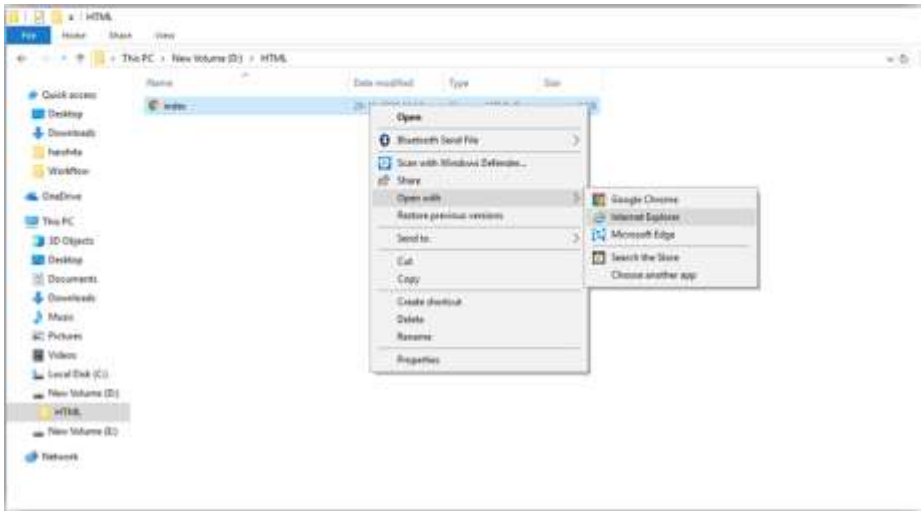
Step 3: Save the HTML file with .htm or .html extension.



```
<!DO
<htm
<hea
<tit
</he
<bod

<h1>

<p>H

</body>
</html>
```

Step 4: Open the HTML page in your web browser.

To run the HTML page, you need to open the file location, where you have saved the file and then either double-click on file or click on open with option

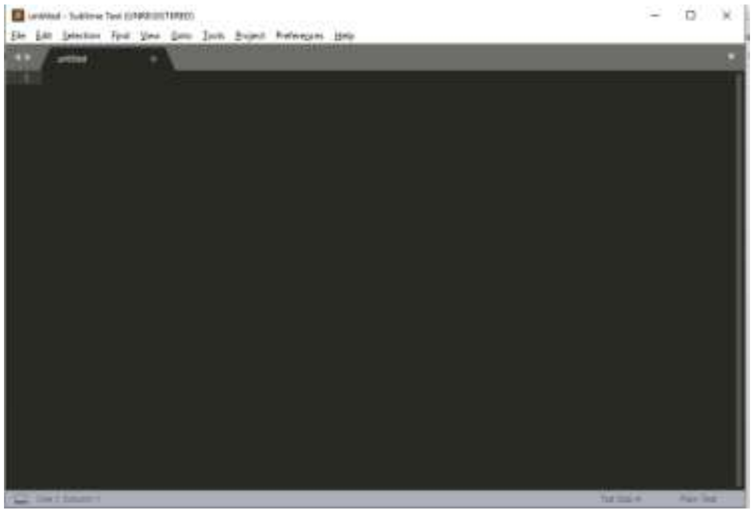


HTML code with Sublime Text-editor. (Recommended after learning basics of HTML)

When you will learn the basics of HTML, then you can use some professional text editors, which will help you to write an efficient and fast code. So to use Sublime Text editors, first it needs to download and install from internet. You can easily download it from this https://www.sublimetext.com/download link and can install in your PC. When installation of Sublime text editor done then you can follow the simple steps to use it:

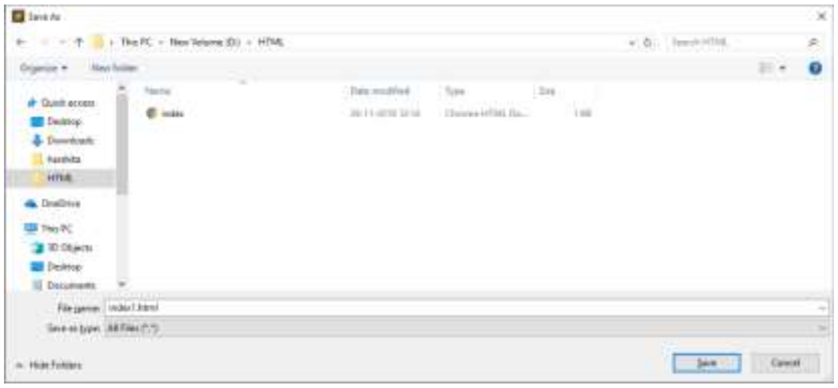
Step 1: Open Sublime Text editor(Windows 8):

To open Sublime Text editor go to Start screen --> type Sublime Text--> Open it. To open a new page press CTRL+N.


untitled - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
untitled

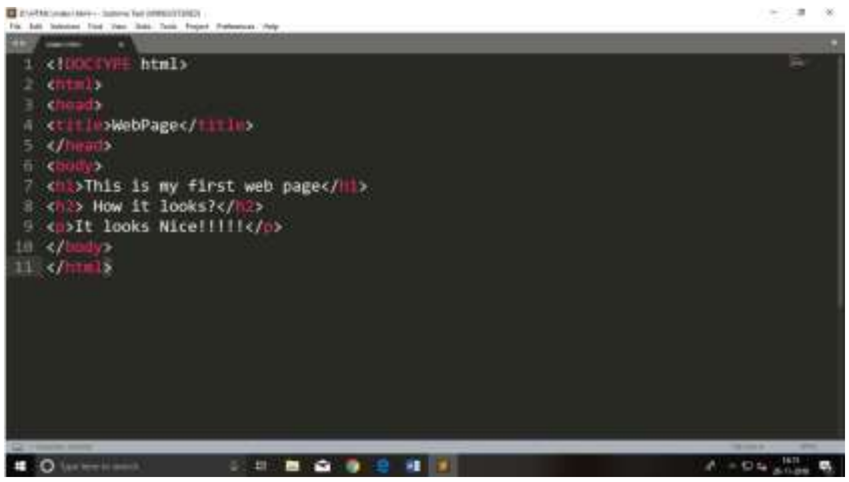
Step 2: Save the page before writing any code.

To save your page in Sublime Text press Ctrl+S or go to File option --› save, to save a file use extension .htm or .html. We recommend to save the file first then write the code because after saving the page sublime text editor will give you suggestions to write code.



File name: index1.html
Save as type: All Files (*.*)

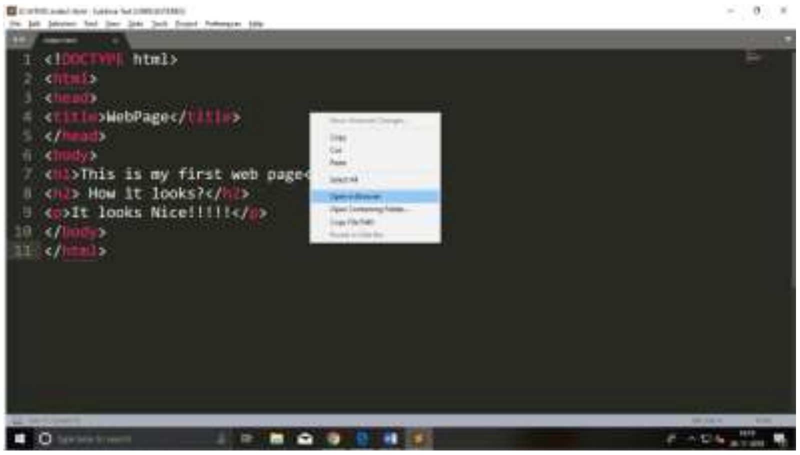
Step 3: Write the code in Sublime Text editor



```
<!DOCTYPE html>
<html>
<head>
<title>WebPage</title>
</head>
<body>
<h1>This is my first web page</h1>
<h2> How it looks?</h2>
<p>It looks Nice!!!!!</p>
</body>
</html>
```

Step 4: Open the HTML page in your Browser

To execute or open this page in Web browser just right click by mouse on sublime text page and click on Open in Browser.

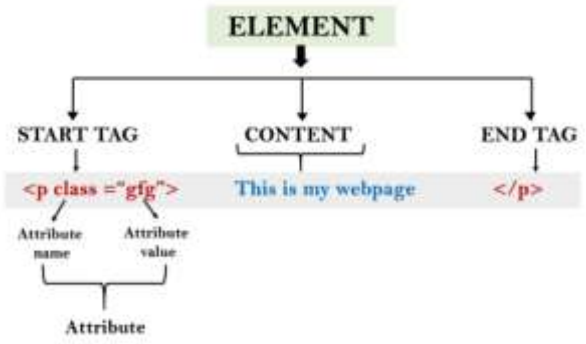


```
<!DOCTYPE html>
<html>
<head>
<title>WebPage</title>
</head>
<body>
<h2>This is my first web page<
<h2> How it looks?</h2>
<p>It looks Nice!!!!!</p>
</body>
</html>
```

Building blocks of HTML

An HTML document consist of its basic building blocks which are:

- Tags: An HTML tag surrounds the content and apply meaning to it. It is written between < and > brackets.
- Attribute: An attribute in HTML provides extra information about the element, and it is applied within the start tag. An HTML attribute contains two fields: name & value.
- Elements: An HTML element is an individual component of an HTML file. In an HTML file, everything written within tags are termed as HTML elements.



Syntax

1. <tag name attribute_name= " attr_value"> content </ tag name>

Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>The basic building blocks of HTML</title>
 </head>
  <body>
      <h2>The building blocks</h2>
      <p>This is a paragraph tag</p>
      <p style="color: red">The style is attribute of paragraph tag</p>
      <span>The element contains tag, attribute and content</span>
  </body>
</html>
```

HTML Tags

HTML tags are like keywords which defines that how web browser will format and display the content. With the help of tags, a web browser can distinguish between an HTML content and a simple content. HTML tags contain three main parts: opening tag, content and closing tag. But some HTML tags are unclosed tags.

An HTML file must have some essential tags so that web browser can differentiate between a simple text and HTML text. You can use as many tags you want as per your code requirement.

- All HTML tags must be enclosed within < > these brackets.
- Every tag in HTML performs different tasks.
- If you have used an open tag <tag>, then you must use a close tag </tag> (except some tags)

Syntax

<tag> content </tag>

Unclosed HTML Tags

Some HTML tags are not closed, for example br and hr.

<br> Tag: br stands for break line, it breaks the line of the code.

<hr> Tag: hr stands for Horizontal Rule. This tag is used to put a line across the webpage.

HTML Meta Tags

DOCTYPE, title, link, meta and style

HTML Text Tags

<p>, <h1>, <h2>, <h3>, <h4>, <h5>, <h6>, <strong>, <em>, <abbr>, <acronym>, <address>, <bdo>, <blockquote>, <cite>, <q>, <code>, <ins>, <del>, <dfn>, <kbd>, <pre>, <samp>, <var> and <br>

HTML Link Tags

<a> and <base>

HTML Image and Object Tags

<img>, <area>, <map>, <param> and <object>

HTML List Tags

<ul>, <ol>, <li>, <dl>, <dt> and <dd>

HTML Table Tags

table, tr, td, th, tbody, thead, tfoot, col, colgroup and caption

HTML Form Tags

form, input, textarea, select, option, optgroup, button, label, fieldset and legend

HTML Scripting Tags

script and noscript

HTML Attribute

- HTML attributes are special words which provide additional information about the elements or attributes are the modifier of the HTML element.
- Each element or tag can have attributes, which defines the behaviour of that element.
- Attributes should always be applied with start tag.
- The Attribute should always be applied with its name and value pair.
- The Attributes name and values are case sensitive, and it is recommended by W3C that it should be written in Lowercase only.
- You can add multiple attributes in one HTML element, but need to give space between two attributes.

Syntax

```
1. <element attribute_name="value">content</element>
```

Attributes Of Body Tag

The HTML <body> tag has several attributes that were previously used to control the appearance of the page, including background color, text color, and link colors. However, most of these attributes are now deprecated in favor of CSS, and are best avoided in modern HTML development.

Here's a breakdown of some key <body> attributes and their modern usage:

Deprecated Attributes:

- background:

Used to set a background image for the document. Deprecated. Use the background-image property in CSS instead.

- bgcolor:

Used to set the background color of the document. Deprecated. Use the background-color property in CSS instead.

- text:

Used to set the color of the text within the document. Deprecated. Use the color property in CSS instead.

- link:

Used to set the color of unvisited links. Deprecated. Use the color property for the a:link pseudo-class in CSS instead.

- alink:

Used to set the color of active links (when the link is being clicked). Deprecated. Use the color property for the a:active pseudo-class in CSS instead.

- vlink:

Used to set the color of visited links. Deprecated. Use the color property for the a:visited pseudo-class in CSS instead.

Example 1 bgcolor and text attributes using color names

```
 <!DOCTYPE html>
<html>
<head>
</head>
<body bgcolor=”cyan” text=”blue”>
   <h1> This is Style attribute</h1>
  <p style="height: 50px; color: blue">It will add style property in element</p>
```

```
    <p style="color: red">It will change the color of content</p>  </body>  </html>
```

Example 2 bgcolor and text attributes using color codes (htmlcolorcodes.com)

```
 <!DOCTYPE html>
<html>
<head>
</head>
<body bgcolor=" #59de32" text=" #053076">
    <h1> This is Style attribute</h1>
   <p style="height: 50px; color: blue">It will add style property in element</p>
    <p style="color: red">It will change the color of content</p>
</body>
</html>
```

Example 3 Applying Background Image

```
 <!DOCTYPE html>
<html>
<head>
</head>
<body background="d:\images\sardar1.jpg" text="blue">
    <h1> This is Style attribute</h1>
   <p style="height: 50px; color: blue">It will add style property in element</p>
    <p style="color: red">It will change the color of content</p>
</body>
</html>
```

Body Tag Modern Alternatives (CSS) Attributes

Instead of using the deprecated attributes, use CSS to style the <body> element and its children:

CSS is even better Choice these days to apply all attributes universally and at one attempt in HTML 5.0.

Example 4 Setting Up attributes Using CSS

```
 <!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: lightblue; /* Sets the background color */
  color: black; /* Sets the text color */
  background-image: url('your_image.jpg'); /* Sets the background image */
  a:link {
    color: blue; /* Sets the color of unvisited links */
  }
  a:visited {
    color: purple; /* Sets the color of visited links */
  }
  a:hover {
    color: red; /* Sets the color of links on hover */
  }
}
</style>
</head>
<body>

<h1>Hello world!</h1>
<p><a href="https://www.w3schools.com">Visit W3Schools.com!</a></p>
</body>
</html>
```

HTML <font> Tag

The HTML <font> Tag plays an important role in the web page to create an attractive and readable web page. The font tag is used to change the color, size, and style of a text and it was used in HTML4. The base font tag is used to set all the text to the same size, color, and face.

Note: Font tag is not supported in HTML5.

```
<!DOCTYPE html>
<html>
<body>
   <p>This is sample paragraph</p>
   <font face="Arial" size="4" color="green">
      This paragrap style by font tag
   </font>
</body>
</html>
```

Syntax

<font size="number" face="font_family" color="color_name|hex_number|rgb_number">

Font Attributes

Table of Content

Font Size

The Font size attribute is used to adjust the size of the text in the HTML document using a font tag with the size attribute. The range of size of the font in HTML is from 1 to 7 and the default size is 3.

<!DOCTYPE html>

```
<html>
<body>
   <font size="1">GeeksforGeeks!</font><br />
   <font size="2">GeeksforGeeks!</font><br />
   <font size="3">GeeksforGeeks!</font><br />
   <font size="4">GeeksforGeeks!</font><br />
   <font size="5">GeeksforGeeks!</font><br />
   <font size="6">GeeksforGeeks!</font><br />
   <font size="7">GeeksforGeeks!</font></body></html>
```

Font Type

The Font type can be set by using face attribute with font tag in HTML document. But the fonts used by the user need to be installed in the system first.

Example

```
<!DOCTYPE html>
<html>
<body>
   <font face="Times New Roman" size="6">
         GeeksforGeeks!!
   </font><br />
   <font face="Verdana" size="6">
         GeeksforGeeks!!
   </font><br />
   <font face="Comic sans MS" size=" 6">
         GeeksforGeeks!!
   </font><br />
   <font face="WildWest" size="6">
         GeeksforGeeks!!
   </font><br />
   <font face="Bedrock" size="6">
         GeeksforGeeks!!
   </font><br />
</body>
</html>
```

Font Color

The Font color is used to set the text color using a font tag with the color attribute in an HTML document. Color can be specified either with its name or with its hex code.

Example

```
<!DOCTYPE html>
<html>
<body>
   <font color="#009900">GeeksforGeeks</font><br />
   <font color="green">GeeksforGeeks</font>
</body>
</html>
```

HTML - Formatting

HTML formatting defines the way of content representation on the webpage to improve the readability, to give the semantic meaning, and to improve visual styling.

HTML formatting is done by using HTML physical and logical tags. In this chapter, we will learn about the text appearance with HTML formatting.

Let's understand what are physical and logical tags:

- Physical Tags: These tags are used to give the visual appearance to the textual content.
- Logical Tags: These tags are used to give logical and semantic meaning to the textual content. There are a few logical tags that are used for screen readers, but the impact of those tags is visible on the browsers.

Use of HTML Formatting

Without formatting, nothing looks good or soothing to our eyes. But HTML formatting is not only for soothing the eye or making textual content attractive. There are few reasons to do the HTML formatting.

HTML formatting is useful in many aspects:

- The appearance of any text provides a clear view of the content intent, such as highlighting the keywords, putting meaningful information in the quotations, underlining the main sentence, etc.
- Formatting helps search engines understand the content structure and is also helpful for search engine optimization.
- Formatting improves the visual layout and improves the readability of the content.

HTML Formatting Tags

The following table has a list of common HTML formatting tags that are used for text formatting:

Tag	Description	Category
<b>	This tag is used to make the text bold.	Physical Tag
<i>	This tag is used to make the text *italic*.	Physical Tag
<big>	This tag is used to make the text bigger. It is not supported in HTML5.	Physical Tag
<small>	This tag is used to make the text smaller.	Physical Tag
<u>	This tag is used to underline text.	Physical Tag
<strike>	This tag is used to strike through text. It is not supported in HTML5.	Physical Tag
<tt>	This tag is used to make text appear in teletype (monospace font). It is not supported in HTML5.	Physical Tag
<strong>	This tag is used to bold text and give it semantic importance.	Logical Tag
<em>	This tag is used to italicize text and give it *semantic emphasis*.	Logical Tag
<sup>	This tag is used to make superscript text (slightly above the normal line).	Other Tag

<sub>	This tag is used to make subscript text (slightly below the normal line).	Other Tag
<ins>	This tag is used to indicate that content has been added (typically underlined).	Other Tag
<del>	This tag is used to indicate that content has been deleted (typically struck through).	Other Tag
<mark>	This tag is used to highlight text with a yellow background.	Other Tag

HTML Formatting Tags with Examples

The detailed explanation of each formatting tag with their examples is as follows:

HTML <b> Tag

HTML <b> tag is used for making the text bold; there is no logical aspect of this tag; it is only used for visual impact.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Bold Text Example</title>
</head>
<body>
  <p>The following word uses a <b>bold</b> typeface.</p>
</body>
</html>
```

HTML <strong> Tag

HTML <strong> tag is used for making the text strong that has more importance, and the text inside it is typically displayed in the bold.

Note: The <b> tag makes the text bold for styling purposes only, while the <strong> tag makes the text bold and also adds importance to the text within its content.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Bold Text Example</title>
</head>
<body>
  <p>The following word uses a <strong>strong</strong> typeface.</p>
</body>
</html>
```

HTML <i> Tag

Any content that is enclosed within the <i>...</i> element is displayed in italicized.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Italic Text Example</title>
</head>
<body>
  <p>The following word uses a <i>italicized</i> typeface.</p>
</body>
</html>
```

HTML <em> Tag

HTML <em> tag gives semantic meaning to the text contained within it and renders it in italics on the browser.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Italic Text Example</title>
</head>
<body>
  <p>The following word uses a <em>emphasized</em> typeface.</p>
</body>
</html>
```

HTML <big> Tag

Any content that is enclosed within the <big>...</big> element is displayed one font size larger than the rest of the text surrounding it.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Larger Text Example</title>
</head>
<body>
  <p>Hello Welcome to <big>Sardar Azeem 03135879331</big>.</p>
</body>
</html>
```

HTML <small> Tag

The content, which is enclosed within the <small>...</small> element, is displayed one font size smaller than the rest of the text surrounding it.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Smaller Text Example</title>
</head>
```

```
<body>
  <p>Hello Welcome to <small>Sardar
Azeem 03135879331</small>.</p>
</body>
</html>
```

HTML <sup> Tag

Any content enclosed within the <sup>…</sup> element is written in superscript; the font size used is the same size as the characters surrounding it but is displayed at half the height of the surrounding characters, giving it a smaller and slightly raised appearance compared to the rest of the text.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Superscript Text Example</title>
</head>
<body>
  <p>The following word uses a <sup>superscript</sup> typeface. </p>
</body>
</html>
```

HTML <sub> Tag

Any content of a <sub>…</sub> element is written in subscript; the font size used is the same as the characters surrounding it and is displayed half a character's height beneath the other characters. It is typically used for writing things like chemical formulas, where certain characters need to be displayed below the regular text line.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Subscript Text Example</title>
</head>
<body>
  <p>The following word uses a <sub>subscript</sub> typeface. </p>
</body>
</html>
```

HTML <ins> Tag

Any content that is enclosed within the <ins>…</ins> element is displayed as inserted text.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Inserted Text Example</title>
</head>
<body>
  <p>I want to drink <del>cola</del> <ins>wine</ins></p>
</body>
</html>
```

HTML <del> Tag

Content that is enclosed within the <del>…</del> element is displayed as deleted text.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Deleted Text Example</title>
</head>
<body>
  <p>Hello welcome to <del>Madras</del> <ins>Chennai</ins></p>
</body>
</html>
```

HTML <u> Tag

Any content enclosed within the <u>…</u> element is displayed with an underline.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Underlined Text Example</title>
</head>
<body>
  <p>The following word uses a <u>underlined</u> typeface.</p>
</body>
</html>
```

HTML <strike> Tag

Content that is enclosed within the <strike>...</strike> element is displayed with strikethrough, which is a thin line through the text.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Strike Text Example</title>
</head>
<body>
  <p>The following word uses a <strike>strikethrough</strike> typeface.</p>
</body>
</html>
```

HTML <mark> Tag

HTML <mark> tag is used to mark or highlight text that is important for notation purposes.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Strike Text Example</title>
</head>
<body>
  <p>The following word uses a <mark>strikethrough</mark> typeface.</p>
</body>
</html>
```

HTML <tt> Tag

Any content enclosed within the <tt>...</tt> element is written in monospaced font. Most of the fonts are known as variable-width fonts because different letters are of different widths (for example, the letter 'm' is wider than the letter 'i'). In a monospaced font, however, each letter has the same width.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Monospaced Font Example</title>
</head>
<body>
  <p>The following word uses a <tt>monospaced</tt> typeface.</p>
</body>
</html>
```

HTML Paragraphs

HTML paragraphs are block-level elements that are used to structure and format text content on a webpage. A paragraph is basically a collection of words and punctuation together. It allows us to organize and present textual information in a coherent and readable manner. The HTML <p> tag is used to create a paragraph element.

Reason to Use Paragraphs

Paragraphs typically create space above and below the text, separating it from surrounding content. They can be styled using CSS to control aspects such as font size, color, alignment, and spacing. In web development, paragraphs play a crucial role in conveying information effectively, enabling clear communication, and enhancing the overall user experience on a website.

Creating a Paragraph

To create a paragraph in HTML, use the <p> tag. Place text inside <p> and </p> that you want to display as a paragraph on a webpage.

Syntax

<p>Text to display as a paragraph on the webpage</p>

Example of HTML Paragraph

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <p>Lorem ipsum odor amet, consectetuer adipiscing elit. Proin eros habitant
accumsan vulputate curae eu fusce vehicula.</p>
    <p>Laoreet sociosqu taciti iaculis cras leo nec litora. Nisi vehicula massa fusce justo
libero duis. Per condimentum vivamus nec elementum nullam sociosqu vel
scelerisque.</p>
  </body>
</html>
```

Line Breaks With Paragraphs

The <br> tags are used to insert line breaks within a paragraph to control the text layout.

Example

```
<!DOCTYPE html>
<html>
<head>
   <title>Line Breaks With
Paragraphs</title>
</head>
<body>
   <p>This is a paragraph with a <br> line
break. </p>
</body>
</html>
```

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Headings With Paragraphs</title>
</head>
<body>
  <h1>Main Heading</h1>
  <p> This is a paragraph beneath the main heading. </p>
</body>
</html>
```

Lists With Paragraphs

Lists can be incorporated within paragraphs for content organization.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Lists With Paragraphs</title>
</head>
<body>
  <p>This is a paragraph following an unordered list.</p>
  <ul>
    <li>Item 1</li>
    <li>Item 2</li>
  </ul>
</body>
</html>
```

Emphasis Within Paragraphs

Tags like <em> and <strong> allow you to emphasize text within paragraphs.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Emphasis Within Paragraphs</title>
</head>
<body>
  <p> This is a <em> paragraph </em> with <strong> emphasized </strong> text. </p>
</body>
</html>
```

Links within Paragraphs

You can insert links within paragraphs using the <a> tag.

Example

```
<html>
<head>
  <title>Links within Paragraphs</title>
</head>
<body>
  <p>Visit our website <a href="https://www.tutorialspoint.com">here </a>. </p>
</body>
</html>
```

Inline Styles Within Paragraphs

You can use the <span> tag with inline styles to apply specific formatting.

Example

```
<html>
<head>
  <title>Inline Styles Within Paragraphs</title>
</head>
<body>
  <p>This is a <span style="color: blue;">blue</span> text within a paragraph. </p>
</body>
</html>
```

Images Within Paragraphs

You can embed images within paragraphs using the <img> tag.

Example

```
<html>
<head>
    <title>Images Within Paragraphs</title>
</head>
<body>
  <p> Here's an image: <img src="\html\images\test.png" alt="Example Image"> </p>
</body>
</html>
```

Superscript and Subscript Within Paragraphs

Use <sup> and <sub> tags to create superscript and subscript text.

Example

```
<html>
<head>
  <title>Superscript and Subscript Within Paragraphs</title>
</head>
<body>
  <p> H<sub>2</sub>O is the chemical formula for water. 2<sup>3</sup> equals 8.</p>
</body>
</html>
```

Abbreviations Within Paragraphs

The <abbr> tag helps define abbreviations or acronyms.

Example

```
<html>
<head>
  <title>Abbreviations within Paragraphs</title>
</head>
<body>
  <p> <abbr title="Hypertext Markup Language">HTML</abbr> is used for web
development.</p>
</body>
</html>
```

Citations Within Paragraphs

The <cite> tag specifies citations and references within paragraphs.

Example

```
<html>
<head>
  <title>Citations Within Paragraphs</title>
</head>
<body>
  <p> The book <cite>War and Peace </cite> is a classic novel. </p>
</body>
</html>
```

Styling Paragraph with CSS

The following are the different ways to style HTML paragraphs:

1. Applying CSS Directly to Paragraphs

You can apply CSS styles directly to the paragraphs by writting inline CSS using the 'style' attribute with the <p> tag.

Example

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <p style="font-size: 24px; color: #f40;">This is the first paragraph.</p>
    <p>This is the second paragraph.</p>
    <p style="font-size: 24px; background-color: #f40; color: #fff;">This is the third
paragraph.</p>
  </body>
</html>
```

2. Applying CSS on 'p' Element

You can apply CSS styles to all paragraphs within the HTML document by writing CSS rules for the <p> tag.

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      p {
        font-size: 22px;
        color: #f40;
      }
    </style>
  </head>
  <body>
    <p>This is the first paragraph.</p>
    <p>This is the second paragraph.</p>
    <p>This is the third paragraph.</p>
  </body>
</html>
```

3. Using CSS Class with Different Paragraphs

You can apply CSS styles to specific paragraphs by creating a CSS class and using it with the different paragraphs. For this, use the 'class' attribute with the <p> tag.

```
<!DOCTYPE html>
<html>
 <head>
  <style>
   .special {
    font-size: 24px;
    color: #f40;
   }
  </style>
 </head>
 <body>
  <p class="special">This is the first paragraph.</p>
  <p>This is the second paragraph.</p>
  <p class="special">This is the third paragraph.</p>
 </body>
</html>
```

CSS provides extensive control over paragraph styles, allowing you to create visually appealing and well-formatted text on your web page.

HTML - Headings

HTML headings define the hierarchy (levels) and structure of content on a webpage. They create a visual hierarchy, with the highest-level heading, which is h1, indicating the most important content or the main heading, and lower-level headings like h2, h3, h4, etc. for subtopics.

Reason to use Headings

Headings are crucial for structuring content and providing a clear visual indication of the beginning of new sections or topics. Properly structured headings enhance readability and user experience on websites, ensuring that information is organized and easy to navigate.

- Heading Impact on SEO: The well-organized headings help search engines to understand the content structure and indexing.

- Highlighting Important Topics: The use of heading tags properly keeps the content readable.

HTML Heading Tags

The headings are defined with headings tags (<h1> to <h6>). It is important to use heading tags to show the content structure on a webpage. HTML has a different level of heading tags. The hierarchy determines the importance of content and aids in creating a clear information flow for both users and search engines.

Example

```
<!DOCTYPE html>
<html>
  <body>
    <h1>This is Heading 1 (H1 Tag)</h1>
    <h2>This is Heading 2 (H2 Tag)</h2>
    <h3>This is Heading 3 (H3 Tag)</h3>
    <h4>This is Heading 4 (H4 Tag)</h4>
    <h5>This is Heading 5 (H5 Tag)</h5>
    <h6>This is Heading 6 (H6 Tag)</h6>
  </body>
</html>
```

Hierarchical Structure of Heading Tags

Below is the list according to the hierarchy of the heading tags (most to least significant) −

- The <h1> Tag − The top-level heading denotes the main topic or title of the entire page.
- The <h2> Tag − Subheadings under <h1> represent major sections related to the main topic. They provide a more specific context to the content.
- The <h3> to <h6> Tags − These tags are used for further subsections or nested content within <h2> headings. They offer a deeper level of hierarchy for organizing content within specific sections.

Examples of HTML Headings

In these examples, you will see the usage of all the heading tags to create different types of headings and styling them using the CSS −

Headings Using <h1> to <h6> Tags

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Example of HTML Headings</title>
  </head>
  <body>
    <h1>Heading 1: Main Heading of Page</h1>
    <h2>Heading 2: Section</h2>
    <h3>Heading 3: Subsection</h3>
    <h4>Heading 4: Sub-subsection</h4>
    <h5>Heading 5: Lower-level heading</h5>
    <h6>Heading 6: Lowest-level heading</h6>
  </body>
</html>
```

Styling Headings With CSS

In the following example, we will apply the style such as font family, font color, font size, etc. to the headings −

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Example of HTML Headings</title>
    <style>
      h1, h2, h3, h4, h5, h6{
       font-family: Verdana;
      }
      h1{
```

```
color: Red;
 font-size: 32px;
      }
      h2{
       color: Green;
       font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>Heading 1: Main Heading of Page</h1>
    <h2>Heading 2: Section</h2>
    <h3>Heading 3: Subsection</h3>
    <h4>Heading 4: Sub-subsection</h4>
    <h5>Heading 5: Lower-level heading</h5>
    <h6>Heading 6: Lowest-level heading</h6>
  </body>
</html>
```

Using HTML Tags Within Heading Tags

HTML headings (h1 to h6) serve as the main titles and subheadings for content organization.

The <span> Tag

You can use the <span> tag to apply inline styles or classes to specific portions of the text within a heading. This allows for custom styling of text within the heading.

```
<!DOCTYPE html>
<html>
<head>
   <title>Using <span> Tag</title>
</head>
<body>
   <h2>This is a <span style="color: blue;">blue</span> word.</h2>
</body></html>
```

The <a>Tag for Links

To create a link within a heading, use the <a> tag. This is useful for headings that lead to other pages or sections of your website.

```
<!DOCTYPE html>
<html>
<head>
  <title>Using <a> Tag for Links</title>
</head>
<body>
  <h1><a href="https://www.tutorialspoint.com">Visit our website</a></h1>
</body>
</html>
```

The <em> and <strong> Tags

These tags are used for emphasizing text within headings. The <em> tag italicizes the text, while <strong> makes it bold.

```
<!DOCTYPE html>
<html>
<head>
  <title>Using <em> and <strong> Tags</title>
</head>
<body>
  <h3>This is <em>emphasized</em> and <strong>important</strong> text.</h3>
</body>
</html>
```

The <sup> and <sub> Tags

In heading, to include superscript or subscript text within a heading,
use <sup> and <sub>.

```
<!DOCTYPE html>
<html>
<head>
  <title>Using <sup> and <sub> Tags</title>
</head>
```

```
<body>
  <h4>The 10<sup>th</sup> floor is at the top.</h4>
  <h5>The chemical formula for water is H<sub>2</sub>O.</h5>
</body>
</html>
```

The <abbr> Tag for Abbreviations

When you need to include an abbreviation or acronym in a heading, use the <abbr> tag. It often provides a tooltip with the full meaning.

```
<!DOCTYPE html>
<html>
<head>
  <title>Using <abbr> Tag for Abbreviations</title>
</head>
<body>
  <h2>HTML stands for <abbr title="Hypertext Markup Language">HTML</abbr>.</h2>
</body>
</html>
```

The <br> Tag for Line Breaks

Sometimes, you might want to create line breaks within a heading for better formatting. The <br> tag serves this purpose.

```
<!DOCTYPE html>
<html>
<head>
  <title>Using <br> Tag for Line Breaks</title>
</head>
<body>
  <h3>This is the first line.<br>This is the second line.</h3>
</body>
</html>
```

The <mark> Tag

Use the <mark> tag to highlight specific text within a heading. It's often used to indicate search results or selected portions of text.

```
<!DOCTYPE html>
<html>
<head>
  <title>Using <mark> Tag</title>
</head>
<body>
  <h1>Search for "<mark>important</mark>" information here.</h1>
</body>
</html>
```

Mistakes to be Avoided

Make sure we avoid the following mistakes while using the heading tag −

- Skipping Levels − Always follow the proper hierarchy (h1, h2, h3, etc.). Don't skip levels.
- Overusing h1 − Reserve h1 for the main title; don't use it multiple times on a page.
- Styling Overload − Avoid excessive styling; CSS should handle the aesthetics, not headings.

HTML Phrase tag

The HTML phrase tags are special purpose tags, which defines the structural meaning of a block of text or semantics of text. Following is the list of phrase tags, some of which we have already discussed in HTML formatting.

- Abbreviation tag : <abbr>
- Acronym tag: <acronym> (not supported in HTML5)
- Marked tag: <mark>
- Strong tag: <strong>
- Emphasized tag : <em>
- Definition tag: <dfn>
- Quoting tag: <blockquote>
- Short quote tag : <q>
- Code tag: <code>
- Keyboard tag: <kbd>
- Address tag: <address>

1. Text Abbreviation tag

This tag is used to abbreviate a text. To abbreviate a text, write text between <abbr> and </abbr> tag.

Example

```
<p>An <abbr title = "Hypertext Markup language">HTML </abbr>language is used to c
reate web pages. </p>
```

2. Marked tag:

The content written between <mark> and </mark> tag will show as yellow mark on browser. This tag is used to highlight a particular text.

Example

```
<p>This tag will <mark>highlight</mark> the text.</p>
```

3. Strong text:

This tag is used to display the important text of the content. The text written between <strong> and </strong> will be displayed as important text.

Example

```
<p>In HTML it is recommended to use <strong>lower-
case</strong>, while writing a code. </p>
```

4. Emphasized text

This tag is used to emphasize the text, and displayed the text in italic form. The text written between <em> and </em> tag will italicized the text.

Example

```
<p>HTML is an <em>easy </em>to learn language.</p>
```

5. Definition tag:

When you use the <dfn> and </dfn> tags, it allow to specify the keyword of the content. Following is the example to show how to definition element.

Example

```
<p><dfn>HTML </dfn> is a markup language. </p>
```

6. Quoting text:

The HTML <blockquote> element shows that the enclosed content is quoted from another source. The Source URL can be given using the cite attribute, and text representation of source can display using <cite> ..... </cite>element.

Example

```
<blockquote cite="https://www.pictacademy.com/famous-
quotes/"><p>?The first step toward success is taken when you refuse to be a captive
 of the environment in which you first find yourself.?</p></blockquote>
 <cite>-Mark Caine</cite>
```

7. Short Quotations:

An HTML <q> .... </q> element defines a short quotation. If you will put any content between <q> .... </q>, then it will enclose the text in double quotes.

Example

```
<p>Steve Jobs said: <q>If You Are Working On Something That You Really Care About,
 You Don't Have To Be Pushed. The Vision Pulls You.</q>?</p>
```

8. Code tags

The HTML <code> </code> element is used to display the part of computer code. It will display the content in monospaced font.

Example

```
<p>First Java program</p>
    <p><code>class Simple{ public static void main(String args[]){
     System.out.println("Sardar Azeem"); }} </code>
   </p>
```

9. Keyboard Tag

In HTML the keyboard tag, <kbd>, indicates that a section of content is a user input from keyboard.

Example

```
<p>Please press <kbd>Ctrl</kbd> + <kbd>Shift</kbd> + t<kbd></kbd> to restore page
on chrome.</p>
```

10. Address tag

An HTML <address> tag defines the contact information about the author of the content. The content written between <address> and </address> tag, then it will be displayed in italic font.

Example

```
<address> You can ask your queries by contact us on <a href="">SardarAzeem@pict
academy.com</a>
  <br> You can also visit at: <br>58 S. Garfield Street. Villa Rica, GA 30187.
 </address>
```

HTML Anchor

HTML Hyperlinks

A hyperlink is a specific type of link that allows users to navigate from one web page or resource to another by clicking on it. You can create hyperlinks using text or images available on a webpage. A hyperlink is created using the HTML Anchor Tag (</a>).

The Anchor (<a>) Tag

An anchor tag, or <a> tag, is a basic element that creates hyperlinks between two pages. Anything which is written between the opening <a> and the closing </a> tags become clickable and when someone clicks on it, the linked page will be opened.

Syntax

Here is the syntax to create a hyperlinks in HTML:

<a href="URL" target="_target_type">Link Text</a>

Creating Hyperlinks (Linking Webpages/Documents)

You can link other webpages or documents by creating the hyperlinking to specific words, images, or any HTML element.

Syntax

<a href="URL" ... attributes-list>Link Text</a>

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Hyperlink Example</title>
</head>
<body>
  <p>Click following link</p>
  <a href="https://www.pictacademy.com/" target="_self">Tutorials Point</a>
</body>
</html>
```

The "target" Attribute

The target attribute specifies the location where linked document is opened. Following are the possible values of target attribute:

S.No.	Option & Description
1	_blank Opens the linked document in a new window or tab.
2	_self Opens the linked document in the same frame.
3	_parent Opens the linked document in the parent frame.
4	_top Opens the linked document in the full body of the window.
5	targetframe Opens the linked document in a named targetframe.

Example

```
<!DOCTYPE html>
<html>
<head>
   <title>Hyperlink Example</title>
   <base href="https://www.tutorialspoint.com/">
</head>
<body>
   <p>Click any of the following links</p>
   <a href="/html/index.htm" target="_blank">Opens in New</a> | <a
href="/html/index.htm" target="_self">Opens in Self</a> | <a href="/html/index.htm"
target="_parent">Opens in Parent</a> | <a href="/html/index.htm" target="_top">Opens
in Body</a>
</body>
</html>
```

Use of Base Path in Hyperlinks

When you link HTML documents related to the same website, it is not required to give a complete URL for every link. You can get rid of it if you use <base> tag in your HTML document header. This tag is used to give a base path for all the links. So your browser will concatenate given relative path to this base path and will make a complete URL.

Example

Following example makes use of <base> tag to specify base URL and later we can use relative path to all the links instead of giving complete URL for every link:

```
<!DOCTYPE html>
<html>
<head>
  <title>Hyperlink Example</title>
  <base href="https://www.tutorialspoint.com/">
</head>
<body>
  <p>Click following link</p>
  <a href="/html/index.htm" target="_blank">HTML Tutorial</a>
</body>
</html>
```

Linking to a Page Section

Linking to a section on the same page allows users to navigate directly to that section. You can create a link in the same to a specific section by using the href attribute with a #id value, where the #id targets an element on the page with a corresponding id attribute.

Example

In the below code, we demonstrate the usage of the href attribute to navigate to a different section within the same page. We provide #idofsection inside the href to navigate sections of our need:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <style>
      div {
         height: 900px;
      }
   </style>
</head>
```

```
<body>
  <h2>Sardar Azeem</h2>
  <div>
    <p>
      PICT Academy: Simply Easy Learning
    </p>
    <a href="#about">Know More</a>
  </div>
  <h2 id="about">Section 2</h2>
  <div>
  <p>
    Pict Academy is a learning platform
    providing Top Rated On Campus and Online Classes, paid premium courses,
    and eBooks. Learn the latest technologies and  programming languages SQL,
MySQL, Python, C, C++, Java, Python, PHP, Machine Learning, data
    science, AI, Prompt Engineering and more.
  </p>
  </div>
</body>


</html>
```

Styling Hyperlinks (Setting Link Color)

You can set colors of your links, active links and visited links using link, alink and vlink attributes of <body> tag.

Example

Save the following in test.htm and open it in any web browser to see how link, alink and vlink attributes work.

```
<html>
<head>
  <title>Hyperlink Example</title>
  <base href="https://www.pictacademy.com/">
</head>
<body alink="#54A250" link="#040404" vlink="#F40633">
```

```
  <p>Click following link</p>
  <a href="/html/index.htm" target="_blank">HTML Tutorial</a>
</body>
</html>
```

Downloadable Links

HTML allows you to create downloadable links where you can create links to make your PDF, DOC, or ZIP files downloadable. To create any link downloadable, you can use the download attribute with the <a> tag and specify the downloadable file path in the href attribute.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Downloadable Link Example</title>
</head>
<body>
  <a href="URL" download>Download File</a>
</body>
</html>
```

Custom File Name

You can also specify the filename for the downloaded file. To give a custom filename the file, you need to provide it to the download attribute.

Here is an example:

```
<a href="/html/src/sample.txt" download="custom-report.txt">Download File</a>
```

File Download Dialog Box

Appearance of HTML anchor tag

An unvisited link is displayed underlined and blue.

A visited link displayed underlined and purple.

An active link is underlined and red.

Creating Image Links

To create an HTML image link, we need an <img> tag and an anchor element. The image element is used to display the image on the web page, and the anchor element is used to specify the destination URL of the link.

Syntax

```
<a href=" destination URL">
  <img src="image URL" alt="alternative text">
</a>
```

Here are some example codes that explain the usage of image links in HTML:

- Create Hyperlink for an Image
- Image Link with Tooltip
- Mouse-Sensitive Images
    - Server-Side Image Maps

Client-Side Image Maps

Create Hyperlink for an Image

```
<!DOCTYPE html>
<html>
<head>
  <title>Image Hyperlink Example</title>
</head>
<body>
  <a href="https://pictacademy.com">
    <img src="/html/images/logo.png"
        alt="PICTACADEMY"
        border="0" />
  </a>
</body>
</html>
```

Image Link with Tooltip

You can also define a tooltip for an image link; when someone moves the mouse over the linked image, it will display a tooltip. To set the tooltip, you can set the title attribute of the <a> tag.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Image Hyperlink Example</title>
</head>
<body>
  <a href="https://www.pictacademy.com" title="Go to PICT">
    <img src="/html/images/logo.png"
        alt="Tutorials Point"
        border="0" />
  </a>
</body>
</html>
```

Mouse-Sensitive Images

The HTML and XHTML standards provide a feature that lets us embed several different links inside a single image. We can create different links on the single image based on different coordinates available on the image.

Once the links are attached to all coordinates, clicking on the different parts of the image redirects us to target documents. Such mouse-sensitive images are known as image maps.

There are two ways to create image maps:

- Server-side image maps: This is enabled by the ismap attribute of the <img> tag and requires access to a server and related image-map processing applications.
- Client-side image maps: This is created with the usemap attribute of the <img> tag, along with corresponding <map> and <area> tags.

Server-Side Image Maps

In the server-side image maps, we simply put the image inside a hyperlink and use the ismap attribute, which makes it a special image, and when the user clicks some place within the image, the browser passes the coordinates of the mouse pointer along with the URL specified in the <a> tag to the web server. The server uses the mouse pointer coordinates to determine which document to deliver back to the browser.

When *ismap* is used, the href attribute of the containing <a> tag must contain the URL of a server application like a CGI or PHP script to process the incoming request based on the passed coordinates.

The coordinates of the mouse position are screen pixels counted from the upper-left corner of the image, beginning with (0,0). The coordinates, preceded by a question mark, are added to the end of the URL.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>ISMAP Hyperlink Example</title>
</head>
<body>
  <p>
    Click on the Image to get its coordinates.
  </p>
  <a href="#" target="_self">
    <img src="/images/logo.png"
        alt="PICTATD"
        ismap/>
  </a>
</body>
</html>
```

Client-Side Image Maps

Client-side image maps are enabled by the usemap attribute of the <img /> tag and defined by special <map> and <area> extension tags. The <map> along with <area> tags define all the image coordinates and corresponding links. The <area> tag inside the map tag specifies the shape and the coordinates to define the boundaries of each clickable hotspot available on the image.

The image that is going to form the map is inserted into the page using the <img /> tag as a normal image, except it carries an extra attribute called usemap.

Example

```
<!DOCTYPE html>
<html lang="en">
<body>
<h1>Welcome to our interactive map!</h1>
<p> Click on a region to visit the respective language page: </p>
<img src="/html/images/lang.jpg" usemap="#langmap" alt="language Map" /> <map
name="langmap">
<area shape="rect" coords="0,0,180,165" alt="HTML" href="html/index.htm"
target="_blank" hreflang="en" />
<area shape="rect" coords="180,0,375,167" alt="JavaScript"
href="javascript/index.htm" target="_blank" hreflang="en" />
<area shape="rect" coords="0,166,180,338" alt="PHP" href="/php/index.htm"
target="_blank" hreflang="en" />
<area shape="rect" coords="180,165,375,338" alt="ReactJS" href="reactjs/index.htm"
target="_blank" hreflang="en" />
</map>
</body>
 </html>
```

Email Links (mailto)

HTML email links allow users to click on a link and automatically open their default email client with a new message composed to the specified email address.

This is done using the mailto: protocol in the href attribute of an <a> (anchor) tag.

You can also predefine the subject and body of the email using the mailto: protocol. This is done by appending ?subject= and &body= to the email address. Spaces and special characters in the subject and body should be URL-encoded. For example, spaces are encoded as %20.

Syntax

```
<a href= "mailto:name@email.com">name@email.com</a>
```

Examples HTML Email Links

Following are some examples that illustrate usage of HTML Email link:

Create Email link using href

The following HTML code illustrates how to create an email link using the href attribute of the <a> tag.

```
<!DOCTYPE html>
<html>

<body>
  <p>
    Creating an HTML Email Link
  </p>
  <a href= "mailto: name@email.com">
    Click to Send Mail
  </a>
</body>
</html>
```

Define Subject and Body in Email Link

HTML also allows you to specify a default email subject as well as an email body along with the email address to make it more specific.

```
<!DOCTYPE html>
<html>

<body>
  <p>
    Creating an HTML Email Link
  </p>
  <a
href="mailto:example@example.com?subject=Hello%20there&body=This%20is%20a%20p
redefined%20email%20body.">
    Click here to Send Mail
  </a>
</body>
</html>
```

Define cc and bcc in Email Link

We can also use the cc and bcc parameters to add carbon copy and blind carbon copy recipients, as shown in the below example:

```
<!DOCTYPE html>
<html>
<body>
  <p>
    Creating an HTML Email Link
  </p>
  <a href= "mailto: name@email.com ?cc=cc@example.com &bcc=bcc@example.com >
    Send email with cc and bcc
  </a>
</body>
</html>
```

Email Links for Multiple Recipients

It is also possible to add multiple recipients to the email link by separating them with commas, as illustrated in the below HTML code.

```
<!DOCTYPE html>
<html>
<body>
  <p>
    Creating an HTML Email Link
  </p>
  <a href="mailto:recipient1@example.com, recipient2@example.com">
    Send email to multiple recipients
  </a>
</body>
</html>
```

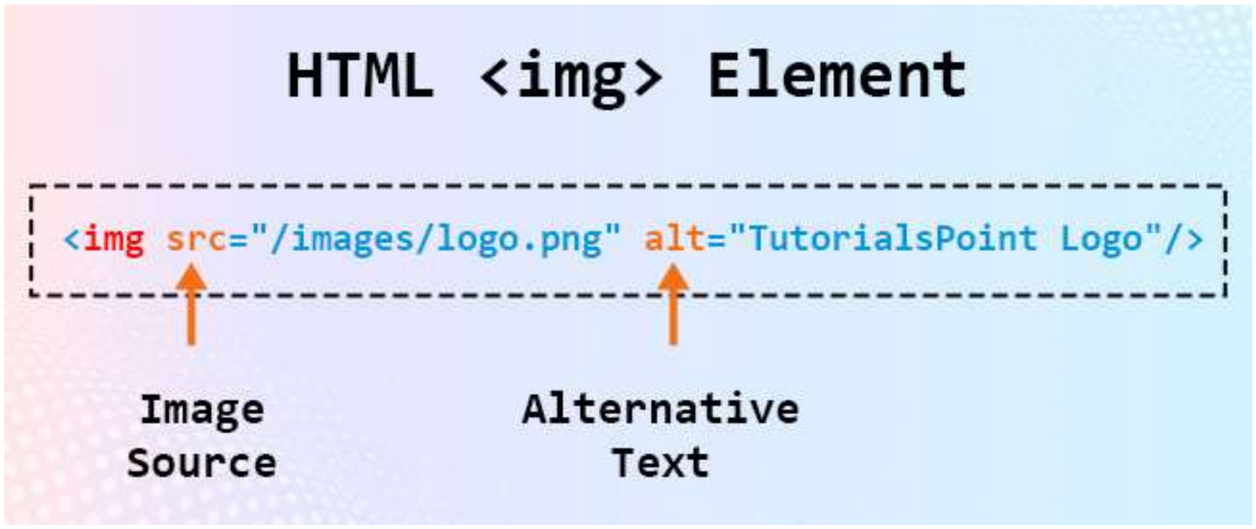
Security Concerns

Adding an HTML email link to your webpage is straightforward, but it can expose your email address to spam. Automated programs, known as email harvesters, can scan web pages for email addresses and add them to spam lists. This can result in a significant increase in unwanted emails

HTML - Images

HTML images provide visual content for web pages, enhancing user experiences and conveying information. They can be photographs, graphics, icons, or illustrations.

HTML offers various elements for embedding, manipulating, and controlling images, contributing to the aesthetics and functionality of websites. Understanding image tags, attributes, and responsive design principles is essential for effective web development.



```
<img src="/images/logo.png" alt="TutorialsPoint Logo"/>
```

HTML Image Syntax

<img src="image_path" alt="Alternate text for the image" width="200px" height="150px" />

Here,

- src: The src attribute defines the path of the image (image URL).
- alt: The alt attribute defines the alternate text; if there is a broken link to the image path, the alternate text displays on the webpage.
- width and height: The width and height attribute define the height and width for the image.

Insert Image

You can insert (embed) an image on the webpage using the <img> tag with the src attribute, which is a required attribute to define the image path.

Syntax

Use the following syntax to insert an image using the <img> tag:

<img src="Image URL" ... attributes-list/>

Example

```
<DOCTYPE html>
<html>
<head>
  <title>Example of HTML Image (Insert on the webpage)</title>
</head>
<body>
  <h1>Example of HTML Image (Insert on the webpage)</h1>
  <img src="/html/images/test.png" alt="Test Image" />
</body>
</html>
```

Set Image Location

Image location (path) must be clearly defined in the src attribute. You can follow the absolute path, which starts with root directory (/), then directory name (if any), and then image name with its extension.

Example

For example, if we have an image named "test.png" and it is stored in the "images" folder, which is in the "html" folder on the root directory. You can simply use an image path like "/html/images/test.png".

```
<!DOCTYPE html>
<html>
<head>
  <title>Using Image in Webpage</title>
</head>
<body>
  <img src="/html/images/test.png" alt="Test Image" />
</body>
</html>
```

Set Image Width and Height

You can set image width and height based on your requirements using width and height attributes. You can specify the width and height of the image in terms of either pixels or a percentage of its actual size.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Set Image Width and Height</title>
</head>
<body>
  <p>Setting image width and height</p>
  <img src="/html/images/test.png" alt="Test Image" width="450px" height="200px" />
</body>
</html>
```

Bordered Image

You can specify the border and its thickness in terms of pixels using the border attribute. A thickness of 0 means there is no border around the picture.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Set Image Border</title>
</head>
<body>
  <p>Setting image Border</p>
  <img src="/html/images/test.png" alt="Test Image" border="3" />
</body>
</html>
```

Image Alignment

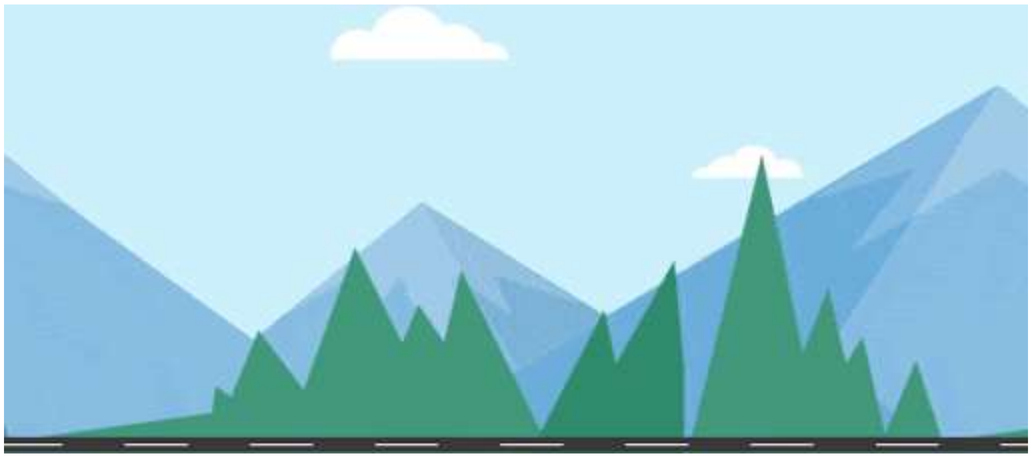
By default, the image will align at the left side of the page, but you can use the align attribute to set it in the center or right.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Set Image Alignment</title>
</head>
<body>
  <p>Setting image Alignment</p>
  <img src="/html/images/test.png" alt="Test Image" border="3" align="right" />
</body>
</html>
```

Animated Images

You can also use animated images (having .gif extensions) on the webpages. There is no specific attribute required to show animated images; you can simply set the path of the animated image (.gif) in the src attribute.



Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Using Animated Images in HTML</title>
</head>
<body>
  <h1>Using Animated Images in HTML</h1>
  <img src="/html/images/animated_image.gif" alt="Animated Images"  />
</body>
</html>
```

Responsive Images

You can also make the images responsive, which will automatically adjust their size based on the devices screen size and resolution. The following are the methods to make images responsive:

1. Using CSS

Using CSS, you can set the width of the image to 100%, which allows the image to scale proportionally to its parent container.

```
<img src="/html/images/test.png" alt="Responsive Image" style="width: 100%; height: auto;"/>
```

2. Using the <picture> Tag

You can also display different images in different sizes or resolutions by using the <picture> tag, which is useful when you want to display different images based on the device.

```
<picture>
  <source media="(min-width: 800px)" srcset="image_path_1">
  <source media="(max-width: 799px)" srcset="image_path_2">
  <img src="default_image_path.jpg" alt="Responsive Image">
</picture>
```

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Responsive Image Example</title>
    <style>
        img {
            width: 100%;
            height: auto;
        }
```

```
  </style>
</head>
<body>
  <h1>Responsive Image Example</h1>
  <img src="/html/images/test.png" alt="A responsive example image" />
</body>
</html>
```

Supported Image Formats

The following table shows the supported image formats in the HTML <img> tag:

Image Format	Image Format Name	Transparency Support	Animation Support	File Extensions
JPEG/JPG	Joint Photographic Experts Group	No	No	.jpg, .jpeg
PNG	Portable Network Graphics	Yes	No	.png
GIF	Graphics Interchange Format	Yes	Yes	.gif
SVG	Scalable Vector Graphics	Yes	No	.svg
WebP	Web Picture format	Yes	Yes	.webp
BMP	Bitmap Image File	No	No	.bmp
ICO	Icon File	Yes	No	.ico

HTML - Text Links

HTML Hyperlinks

A hyperlink is a specific type of link that allows users to navigate from one web page or resource to another by clicking on it. You can create hyperlinks using text or images available on a webpage. A hyperlink is created using the HTML Anchor Tag (</a>).

The Anchor (<a>) Tag

An anchor tag, or <a> tag, is a basic element that creates hyperlinks between two pages. Anything which is written between the opening <a> and the closing </a> tags become clickable and when someone clicks on it, the linked page will be opened.

Syntax

```
<a href="URL" target="_target_type">Link Text</a>
```

Creating Hyperlinks (Linking Webpages/Documents)

You can link other webpages or documents by creating the hyperlinking to specific words, images, or any HTML element.

Syntax

```
<a href="URL" ... attributes-list>Link Text</a>
```

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Hyperlink Example</title>
</head>
<body>
  <p>Click following link</p>
  <a href="https://www.tutorialspoint.com/" target="_self">Tutorials Point</a>
</body>
</html>
```

The "target" Attribute

The target attribute specifies the location where linked document is opened. Following are the possible values of target attribute:

S.No.	Option & Description
1	_blank Opens the linked document in a new window or tab.
2	_self Opens the linked document in the same frame.
3	_parent

	Opens the linked document in the parent frame.
4	_top Opens the linked document in the full body of the window.
5	targetframe Opens the linked document in a named targetframe.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Hyperlink Example</title>
  <base href="https://www.tutorialspoint.com/">
</head>
<body>
  <p>Click any of the following links</p>
  <a href="/html/index.htm" target="_blank">Opens in New</a> | <a href="/html/index.htm" target="_self">Opens in Self</a> | <a href="/html/index.htm" target="_parent">Opens in Parent</a> | <a href="/html/index.htm" target="_top">Opens in Body</a>
</body>
</html>
```

Use of Base Path in Hyperlinks

When you link HTML documents related to the same website, it is not required to give a complete URL for every link. You can get rid of it if you use <base> tag in your HTML document header. This tag is used to give a base path for all the links. So your browser will concatenate given relative path to this base path and will make a complete URL.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Hyperlink Example</title>
  <base href="https://www.tutorialspoint.com/">
</head>
<body>
```

```
  <p>Click following link</p>
  <a href="/html/index.htm" target="_blank">HTML Tutorial</a>
</body>
</html>
```

Linking to a Page Section

Linking to a section on the same page allows users to navigate directly to that section. You can create a link in the same to a specific section by using the href attribute with a #id value, where the #id targets an element on the page with a corresponding id attribute.

Example

```
<!DOCTYPE html>
<html lang="en">

<head>
   <style>
      div {
         height: 900px;
      }
   </style>
</head>

<body>
  <h2>Ed-Tech</h2>
  <div>
    <p>
      Tutorialspoint: Simply Easy Learning
    </p>
    <a href="#about">Know More</a>
  </div>
  <h2 id="about">Section 2</h2>
  <div>
  <p>
    Tutorials Point is an online learning platform
```

```
      providing free tutorials, paid premium courses,
      and eBooks. Learn the latest technologies and
      programming languages SQL, MySQL, Python, C,
      C++, Java, Python, PHP, Machine Learning, data
      science, AI, Prompt Engineering and more.
  </p>
  </div>
</body>

</html>
```

Styling Hyperlinks (Setting Link Color)

You can set colors of your links, active links and visited links using link, alink and vlink attributes of <body> tag.

Example

```
<html>
<head>
  <title>Hyperlink Example</title>
  <base href="https://www.tutorialspoint.com/">
</head>
<body alink="#54A250" link="#040404" vlink="#F40633">
  <p>Click following link</p>
  <a href="/html/index.htm" target="_blank">HTML Tutorial</a>
</body>
</html>
```

Downloadable Links

HTML allows you to create downloadable links where you can create links to make your PDF, DOC, or ZIP files downloadable. To create any link downloadable, you can use the download attribute with the <a> tag and specify the downloadable file path in the href attribute.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Downloadable Link Example</title>
</head>
<body>
  <a href="/html/src/sample.txt" download>Download File</a>
</body>
</html>
```

Custom File Name

You can also specify the filename for the downloaded file. To give a custom filename the file, you need to provide it to the download attribute.

Here is an example:

```
<a href="/html/src/sample.txt" download="custom-report.txt">Download File</a>
```

Creating Image Links

To create an HTML image link, we need an <img> tag and an anchor element. The image element is used to display the image on the web page, and the anchor element is used to specify the destination URL of the link.

Here, the href attribute of <a> element contains the destination link and src attribute of <img> tag contains the path of image.

Syntax

```
<a href=" destination URL">
  <img src="image URL" alt="alternative text">
</a>
```

Examples of HTML Image Links

- Create Hyperlink for an Image
- Image Link with Tooltip
- Mouse-Sensitive Images
    - Server-Side Image Maps
    - Client-Side Image Maps

Create Hyperlink for an Image

In the following example, we are using an image as a hyperlink. If you execute the below code, an image will be displayed, and if we click on it, the page will redirect to the home page of Tutorials Point.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Image Hyperlink Example</title>
</head>
<body>
  <a href="https://www.tutorialspoint.com">
    <img src="/html/images/logo.png"
        alt="Tutorials Point"
        border="0" />
  </a>
</body>
</html>
```

Image Link with Tooltip

You can also define a tooltip for an image link; when someone moves the mouse over the linked image, it will display a tooltip. To set the tooltip, you can set the title attribute of the <a> tag.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Image Hyperlink Example</title>
</head>
<body>
  <a href="https://www.tutorialspoint.com" title="Go to TutorialsPoint">
    <img src="/html/images/logo.png"
```

```
        alt="Tutorials Point"
        border="0" />
  </a>
</body>
</html>
```

Mouse-Sensitive Images

The HTML and XHTML standards provide a feature that lets us embed several different links inside a single image. We can create different links on the single image based on different coordinates available on the image.

Once the links are attached to all coordinates, clicking on the different parts of the image redirects us to target documents. Such mouse-sensitive images are known as image maps.

There are two ways to create image maps:

- Server-side image maps: This is enabled by the ismap attribute of the <img> tag and requires access to a server and related image-map processing applications.
- Client-side image maps: This is created with the usemap attribute of the <img> tag, along with corresponding <map> and <area> tags.

Server-Side Image Maps

In the server-side image maps, we simply put the image inside a hyperlink and use the ismap attribute, which makes it a special image, and when the user clicks some place within the image, the browser passes the coordinates of the mouse pointer along with the URL specified in the <a> tag to the web server. The server uses the mouse pointer coordinates to determine which document to deliver back to the browser.

When *ismap* is used, the href attribute of the containing <a> tag must contain the URL of a server application like a CGI or PHP script to process the incoming request based on the passed coordinates.

The coordinates of the mouse position are screen pixels counted from the upper-left corner of the image, beginning with (0,0). The coordinates, preceded by a question mark, are added to the end of the URL.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>ISMAP Hyperlink Example</title>
</head>
```

```
<body>
  <p>
    Click on the Image to get its coordinates.
  </p>
  <a href="#" target="_self">
    <img src="/images/logo.png"
        alt="Tutorials Point"
        ismap/>
  </a>
</body>
</html>
```

Client-Side Image Maps

Client-side image maps are enabled by the usemap attribute of the <img /> tag and defined by special <map> and <area> extension tags. The <map> along with <area> tags define all the image coordinates and corresponding links. The <area> tag inside the map tag specifies the shape and the coordinates to define the boundaries of each clickable hotspot available on the image.

The image that is going to form the map is inserted into the page using the <img /> tag as a normal image, except it carries an extra attribute called usemap.

On running the below code, an image with clickable areas will be displayed. If you click on one of the area, you will be redirected to the tutorial of that part.

To know how the value of the coords attribute is calculated, you can visit the explanation of coords attribute.

Example

```
<!DOCTYPE html>
<html lang="en">

<body>
  <h1>Welcome to our interactive map!</h1>
  <p>
    Click on a region to visit the
    respective language page:
  </p>
  <img src="/html/images/lang.jpg"
```

```
      usemap="#langmap"
      alt="language Map" />

   <map name="langmap">
     <area shape="rect"
          coords="0,0,180,165"
          alt="HTML"
          href="html/index.htm"
          target="_blank"
          hreflang="en" />
     <area shape="rect"
          coords="180,0,375,167"
          alt="JavaScript"
          href="javascript/index.htm"
          target="_blank"
          hreflang="en" />
     <area shape="rect"
          coords="0,166,180,338"
          alt="PHP"
          href="/php/index.htm"
          target="_blank" hreflang="en" />
     <area shape="rect"
          coords="180,165,375,338"
          alt="ReactJS"
          href="reactjs/index.htm"
          target="_blank"
          hreflang="en" />
   </map>
</body>
</html>
```

Coordinate System in HTML Images

Shape	Coordinates	Description
Rectangle	x_1 , y_1 , x_2 , y_2	Where x_1 and y_1 are the coordinates of the upper left corner of the rectangle; x_2 and y_2 are the coordinates of the lower right corner.
Circle	x_c , y_c , radius	Where x_c and y_c are the coordinates of the center of the circle, and radius is the circle's radius. A circle centred at 200,50 with a radius of 25 would have the attribute coords="*200,50,25*".
Polygon	x_1 , y_1 , x_2 , y_2 , x_3 , y_3 , … x_n , y_n	The various x-y pairs define vertices (points) of the polygon, with a "line" being drawn from one point to the next point. A diamond-shaped polygon with its top point at 20,20 and 40 pixels across at its widest points would have the attribute coords="*20,20,40,40,20,60,0,40*".

Email Links (mailto)

HTML email links allow users to click on a link and automatically open their default email client with a new message composed to the specified email address.

This is done using the mailto: protocol in the href attribute of an <a> (anchor) tag.

You can also predefine the subject and body of the email using the mailto: protocol. This is done by appending ?subject= and &body= to the email address. Spaces and special characters in the subject and body should be URL-encoded. For example, spaces are encoded as %20.

Syntax

```
<a href= "mailto:name@email.com">name@email.com</a>
```

Examples HTML Email Links

Create Email link using href

```
<!DOCTYPE html>
<html>


<body>
  <p>
    Creating an HTML Email Link
  </p>
  <a href= "mailto: name@email.com">
    Click to Send Mail
  </a>
</body>

</html>
```

Define Subject and Body in Email Link

HTML also allows you to specify a default email subject as well as an email body along with the email address to make it more specific.

```
<!DOCTYPE html>
<html>

<body>
  <p>
    Creating an HTML Email Link
  </p>
  <a
href="mailto:example@example.com?subject=Hello%20there&body=This%20is%20a%20p
redefined%20email%20body.">
    Click here to Send Mail
  </a>
</body>

</html>
```

Define cc and bcc in Email Link

We can also use the cc and bcc parameters to add carbon copy and blind carbon copy recipients, as shown in the below example:

```
<!DOCTYPE html>
<html>

<body>
  <p>
```

```
    Creating an HTML Email Link
  </p>
  <a href= "mailto: name@email.com ?cc=cc@example.com &bcc=bcc@example.com >
    Send email with cc and bcc
  </a>
</body>

</html>
```

Email Links for Multiple Recipients

It is also possible to add multiple recipients to the email link by separating them with commas, as illustrated in the below HTML code.

```
<!DOCTYPE html>
<html>

<body>
  <p>
    Creating an HTML Email Link
  </p>
  <a href="mailto:recipient1@example.com, recipient2@example.com">
    Send email to multiple recipients
  </a>
</body>

</html>
```

HTML - <marquee> Tag

HTML <marquee> tag is used to create auto scrolling or moveable element within webpage.

If we place any content inside this tag then that element will slide from right to left by default on it's own, but we can change the direction and axis through the attribute. This tag is now deprecated tag but supported by major browsers till now, we recommend you to use JavaScript and CSS to create this effect.

Syntax

```
<marquee>
 ...
</marquee>
```

Attribute

HTML marquee tag supports Global and Event attributes of HTML. Accept some specific attributes as well which is listed below.

Attribute	Value	Description
width	pixels	Specifies the width of the marquee. This can be a value like 10 or 20% etc.
height	pixels	Specifies the height of the marquee. This can be a value like 10 or 20% etc.(Deprecated)
direction	up down left right	Specifies the direction in which marquee should scroll.(Deprecated)
behavior	scroll slide alternate	Specifies the type of scrolling of the marquee.(Deprecated)
scrolldelay	value	Specifies how long to delay between each jump.(Deprecated)
scrollamount	value	Specifies the speed of marquee element.(Deprecated)
loop	number	Specifies how many times to loop. The default value is INFINITE, which means that the marquee loops endlessly.(Deprecated)
bgcolor	color_name or color_code	Specifies background color in terms of color name or color hex value.(Deprecated)
hspace	pixels	Specifies horizontal space around the marquee.(Deprecated)
vspace	pixels	Specifies vertical space around the marquee.(Deprecated)

Methods

Methods are used to perform some particular task on any element. Below listed method can be used on <marquee> element.

Method	Description
start()	This method is used to start the scrolling of the <marquee> element.
stop()	This method is used to stop the scrolling of the <marquee> element.

Event Handlers

Event Handlers are used to trigger or activate taske based on the behavior of the element. Below listed event handlers can be used on <marquee> element.

Event Handlers	Description
onbounce	This will trigger when the scrolling reaches to the end, but applicable only, when the behavior is set to alternate.
onfinish	This will trigger when the scrolling completed a loop, but applicable only, when the loop is set to a genuine number.
onstart	This will trigger when the scrolling start.

Examples of HTML marquee Tag

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML marquee Tag</title>
</head>
<body>
  <!-- Marquee Element  Default Scrollong from right to left -->
  <marquee>
     <h2>Tutorialspoint: Simply Easy Learning</h2>
  </marquee>
</body>
</html>
```

Implementing Horizontal Scrolling Effect

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML marquee Tag</title>
</head>
<body>
  <h3>From right to left Scrolling</h3>
  <marquee height="100"direction="left">Tutorialpoint</marquee>
  <h3>From left to right Scrolling</h3>
  <marquee height="100"direction="right">Tutorialpoint</marquee>
</body>
</html>
```

Implementing Vertical Scrolling Effect

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML marquee Tag</title>
</head>
<body>
  <h3>From down to up Scrolling</h3>
  <marquee height="100"direction="up">Tutorialpoint</marquee>
  <h3>From up to down Scrolling</h3>
  <marquee height="100"direction="down">Tutorialpoint</marquee>
</body>
</html>
```

Using all attributes on marquee Tag

```
<!DOCTYPE html>
<html>
<head>
<title>HTML marquee Tag</title>
</head>
<body>
<h2>Default Marquee Element</h2>
<marquee>
<p>Tutorialspoint: Simply Easy Learning</p>
 </marquee>
<h2>Setting width, Height and bgcolor on Marquee Element</h2>
 <marquee width="50%" height="25%" bgcolor="lightgray">
<p>Tutorialspoint: Simply Easy Learning</p>
</marquee>
<h2>Setting Behaviour Marquee Element</h2>
<marquee behavior="alternate">
<p>Tutorialspoint: Simply Easy Learning</p>
</marquee>
<h2>Setting Speed on Marquee Element</h2>
<marquee scrollamount="10">
<p>Tutorialspoint: Simply Easy Learning</p>
</marquee>
<h2>Setting delay time on Marquee Element</h2>
<marquee scrolldelay="600">
<p>Tutorialspoint: Simply Easy Learning</p>
</marquee>
</body> </html>
```

HTML - Block and Inline Elements

HTML block elements are used to create the logical and semantic layout of a web page. They help to organize the content into meaningful sections and make it easier for browsers, search engines, and site visitors to understand the structure and meaning of different parts of the web page. Inline elements are used to make useful block elements, like adding anchor links.

There are various tags that you can use to create blocks, such as <div>, <p>, <table>, and so on.

All the HTML elements can be categorized into two categories:

- Block-level Elements
- Inline Elements

HTML Block-level Elements

Block-level elements start on a new line, and anything that follows them appears on the next line. These elements may contain margins to add some space before and after. These elements take up the full width of their parent elements by default; you may set their width by using the CSS width property.

List of HTML Block-level Elements

HTML Block Elements				
<address>	<article>	<aside>	<blockquote>	<canvas>
<dd>	<div>	<dl>	<dt>	<fieldset>
<figcaption>	<figure>	<footer>	<form>	<h1> - <h6>
<header>	<hr>	<li>	<main>	<nav>
<noscript>	<ol>	<p>	<pre>	<section>
<table>	<tfoot>	<ul>	<video>	

Example of Block-level Elements

The following example demonstrates the block-level elements. Here, we are using one heading and two paragraphs separated by a horizontal line.

```
<!DOCTYPE html>
<html>
<head>
   <title>HTML Block Level Elements</title>
</head>
<body>
```

```
  <h3>HTML Block Level Elements</h3>
  <p>
    This line will appear in the next line
    after Heading.
  </p>
  <hr />
  <p>
    This line will appear after Horizontal
    Line.
  </p>
</body>
</html>
```

HTML Inline Elements

Inline elements can appear within the same line and do not start a new line on their own.

List of HTML Inline Elements

HTML Inline Elements				
<a>	<abbr>	<acronym>	<b>	<bdo>
<big>	<br>	<button>	<cite>	<code>
<dfn>	<em>	<i>	<img>	<input>
<kbd>	<label>	<map>	<object>	<output>
<q>	<samp>	<script>	<select>	<small>
<span>	<strong>	<sub >	<sup>	<textarea>
<time>	<tt>	<var>		

Example of Inline Elements

The following example demonstrates inline elements. Here, we are making the paragraph's text bold and italic using inline emelents <b> and <i> —

```
<!DOCTYPE html>
<html>
<head>
   <title>HTML inline Element</title>
</head>
<body>
```

```
    <h3>Inline Elements in HTML</h3>
    <!-- Using <b> inline element -->
    <p>This <b>paragraph</b> is bold. </p>
    <!-- Using <i> inline element  -->
    <p>This is an <i>italic</i> paragraph.</p>
</body>
</html>
```

Grouping Block and Inline Elements

Block-level and inline elements can be grouped using the <div> tag. The <div> tag is a block-level element that plays a big role in grouping various other HTML tags and applying CSS to groups of elements.

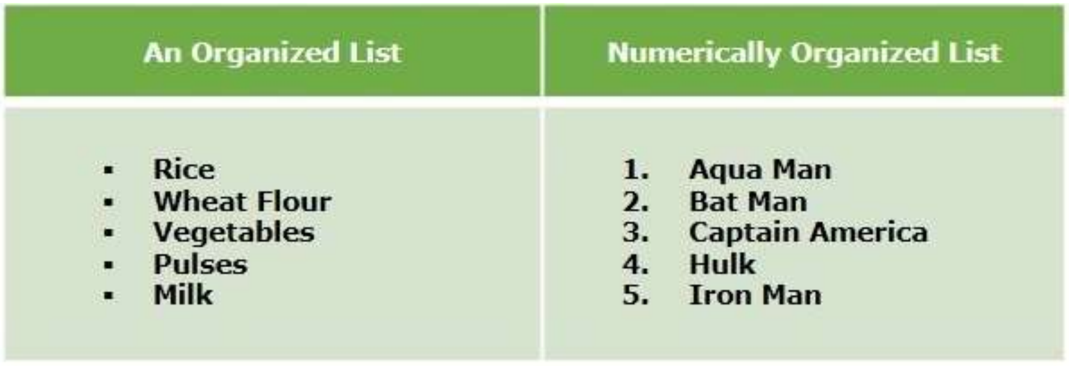
Example

```
<!DOCTYPE html>
<html>
<head> <title>HTML div Tag</title> </head>
<body>
<!-- First group of tags -->
<div style="background-color:yellow">
<h4>This is first group</h4>
<p>Following is a list of vegetables</p>
<ul>
<li>Beetroot</li>
<li>Ginger</li>
<li>Potato</li>
<li>Radish</li>
</ul>
</div>
<!-- Second group of tags -->
<div style="background-color:cyan">
<h4>This is second group</h4>
<p>Following is a list of fruits</p>
```

```
<ul>
<li>Apple</li>
<li>Banana</li>
<li>Mango</li>
<li>Strawberry</li>
</ul>
</div>
</body> </html>
```

HTML Lists

HTML lists are group or collection of items. These items can be both organized and unorganized depending on the requirement. They help in organizing, structuring, and presenting information to make it more user-friendly, readable, and accessible. Sample lists are shown below. –



An Organized List	Numerically Organized List
▪ Rice ▪ Wheat Flour ▪ Vegetables ▪ Pulses ▪ Milk	1. Aqua Man 2. Bat Man 3. Captain America 4. Hulk 5. Iron Man

Unordered lists

Unordered lists display lists of items that are not in a specific order. The unordered lists are marked with bullet points. To create an unordered list, the <ul> tag is used along with the <li> tag. Here, the <li> tag specifies the list items.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML List</title>
</head>
<body>
  <h2>Example of HTML List</h2>
  <ul>
```

```
    <li>HTML</li>
    <li>CSS</li>
    <li>JavaScript</li>
    <li>Java</li>
    <li>JavaFX</li>
  </ul>
</body>
</html>
```

Ordered Lists

Ordered lists are lists of items that are in a specific order. The ordered lists are marked with numbers by default; you can change the numbers into alphabets, roman numbers, etc. by using the type attribute or the CSS list-style-type property.

To create an ordered list, the <ol> tag is used along with the <li> tag, where <li> specifies the list items.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML List</title>
</head>
<body>
  <h2>Example of HTML List</h2>
  <ol>
    <li>HTML</li>
    <li>CSS</li>
    <li>JavaScript</li>
    <li>Java</li>
    <li>JavaFX</li>
  </ol>
</body>
</html>
```

Definition Lists

Definition lists are lists of items with their corresponding descriptions. The definition lists are created by using the <dl>, <dt>, and <dd> tags. Where the <dl> tag specifies the "definition list", the <dt> tag specifies the "definition term", and the <dd> tag specifies the "definition description".

Example

```
<!DOCTYPE html>
<html>
<head>
   <title>HTML List</title>
</head>
<body>
   <h2>Example of HTML List</h2>
   <dl>
      <dt>HTML</dt>
      <dd>HyperText markup languague</dd>
      <dt>CSS</dt>
      <dd>Cascading Style Sheet</dd>
      <dt>JS</dt>
      <dd>JavaScript</dd>
   </dl>
</body>
</html>
```

Nested Lists

A list created within another list is known as a nested list. HTML also allows nesting of lists within one another to generate complex document structures.

Example

In the following example, we are nesting an unordered list within an ordered list:

```
<!DOCTYPE html>
```

```
 <html>
<head>
<title>HTML List</title>
</head>
<body>
<h2>Example of HTML Nested List</h2>
<ol>
<li>Item One</li>
<li>Item Two <ul>
<li>Subitem A</li>
<li>Subitem B</li>
</ul>
</li>
<li>Item Three</li>
</ol>
</body>
</html>
```

Grouping Lists Inside <div> Tag

To enhance styling and layout, lists are often wrapped inside the <div> elements. This grouping aids in applying consistent styles and creating visually appealing list structures.

Example

```
<!DOCTYPE html>
<html> <head>
<title>HTML List</title>
</head>
<body>
<h2>Grouping of HTML List elements with div tag</h2>
 <div>
<p>First List</p>
<ol>
<li>Item One</li>
```

```
<li>Item Two</li>
</ol>
</div>
<div>
<p>Second List</p>
<ol>
<li>Item Three</li>
<li>Item Four</li>
</ol>
</div>
</body>
</html>
```

Styling Lists

Lists can be styled using CSS to enhance visual presentation. Custom styles can be applied to list items, creating unique and visually appealing designs. For this, we use the <style> tag, which is a way of specifying internal CSS.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>HTML List</title>
<style>
li { font-size: 16px; }
div { color: red; }
</style>
</head>
<body>
<h2>HTML List with CSS</h2>
<div>
<p>First List</p>
<ol>
<li>Item One</li>
```

```
<li>Item Two</li>
</ol>
</div>
<div>
<p>Second List</p>
<ol>
<li>Item Three</li>
<li>Item Four</li> </ol> </div> </body> </html>
```

HTML Lists Marker Types

CSS allows customization of marker types for list items. To do so, we use the CSS list-style-type property, which can be set to change markers to circles, squares, or other symbols.

Example

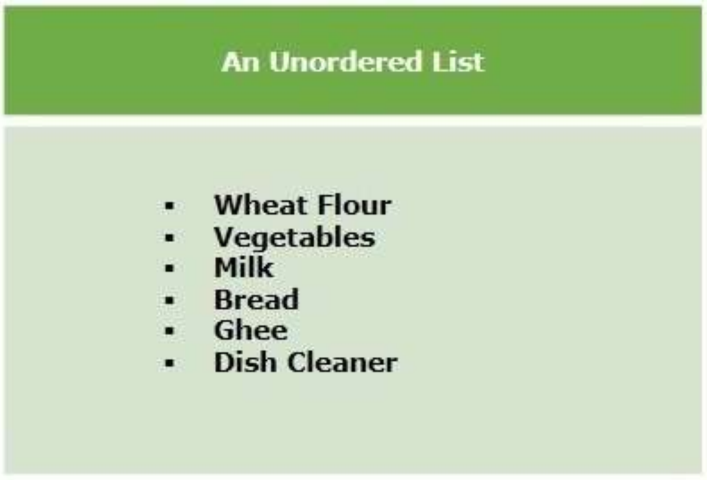
```
<!DOCTYPE html>
<html>
<head>
<title>HTML List</title>
<style>
li { font-size: 16px; list-style-type: square; }
</style>
</head>
<body>
<h2>HTML list-style-type Property</h2>
<div>
<p>First List</p>
<ol>
<li>Item One</li>
<li>Item Two</li>
</ol> <
/div>
<div>
<p>Second List</p>
```

```
 <ol>
<li>Item Three</li>
<li>Item Four</li>
</ol>
</div>
</body>
</html>
```

Unordered HTML Lists

An unordered list is a collection of list items that do not have a specific order or sequence and are marked with the bullets. An HTML unordered list is created by <ul> the tag, where each list item is defined by the <li> tag.

This type of list is used for describing a particular service or product, as it does not require any order to be followed.



Creating Unordered Lists

To create an unordered list in HTML, we use the <ul> tag and nest <li> tags inside it. Each <li> element represents one item in the list. By default, the browser will automatically display disc bullet points for each item. However, we can change this bullet style using the type attribute on the <ul> element.

Example

```
<!DOCTYPE html>
<html>
<head>
   <title>HTML Unordered List</title>
```

```
</head>
<body>
  <ul>
    <li>Beetroot</li>
    <li>Ginger</li>
    <li>Potato</li>
    <li>Radish</li>
  </ul></body></html>
```

HTML Unordered List - Specifying List Marker

The type attribute for the <ul> tag is used to specify the type of bullet for the unordered list in HTML. By default, disc bullet type is used. But we can change this with the help of the following options:

S.No	Value & Description
1	disc It is the default type of marker.
2	square List items will be displayed with a square marker.
3	circle It will set the marker to a hollow circle.

Disc Marker

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML Unordered List</title>
</head>
<body>
  <p>An unordered list with default disc marker:</p>
  <ul type="disc">
    <li>Apple</li>
    <li>Guava</li>
    <li>Carrot</li>
    <li>Orange</li>
  </ul>
</body>
</html>
```

Square Marker

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML Unordered List</title>
</head>
<body>
  <p>An unordered list with square marker:</p>
  <ul type="square">
    <li>Nuts</li>
    <li>Oats</li>
    <li>Eggs</li>
    <li>Dates</li>
  </ul>
</body>
</html>
```

Circle Marker

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML Unordered List</title>
</head>
<body>
```

```
  <p>An unordered list with hollow circle marker:</p>
  <ul type="circle">
    <li>Rice</li>
    <li>Pulses</li>
    <li>Flour</li>
    <li>Beans</li>
  </ul>
</body>
</html>
```

HTML Unordered List Without Bullets

You can also display the list items of an unordered list without showing the bullets. To display an unordered list without bullets, define the "none" to the list-style-type property.

Syntax

```
<ul style="list-style-type: none">
  <li>Item in list...</li>
  <li>Item in list...</li>
  <li>Item in list...</li>
</ul>
```

Example

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
  <ul style="list-style-type: none">
    <li>Abdul</li>
    <li>Jason</li>
    <li>Yadav</li>
  </ul>
</body>
/html>
```

Styling Unordered HTML Lists

Styling an unordered list (<ul>) using CSS allows for customization of the list's appearance, including modifying bullet points, spacing, and overall design.

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Styled Unordered List</title>
  <style>
    ul {
      list-style-type: square;
      /* Custom bullet type */
      padding: 0;
      /* Removes default padding */
    }
    li {
      margin-bottom: 8px;
      /* Adds spacing between list items */
      background-color: #f0f0f0;
      /* Background color */
      border: 1px solid #ccc;
      /* Border */
      padding: 8px;
      /* Padding inside each list item */
      transition: background-color 0.3s;
      /* Smooth transition effect */
    }
    li:hover {
      background-color: #e0e0e0;
      /* Change background on hover */
    }
  </style>
</head>
<body>
  <ul>
    <li>Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
  </ul>
</body>
</html>
```

Nested Unordered Lists

HTML allows nesting of lists; you can create nested unordered lists to display a list of items inside an item of the outer list element.

Example

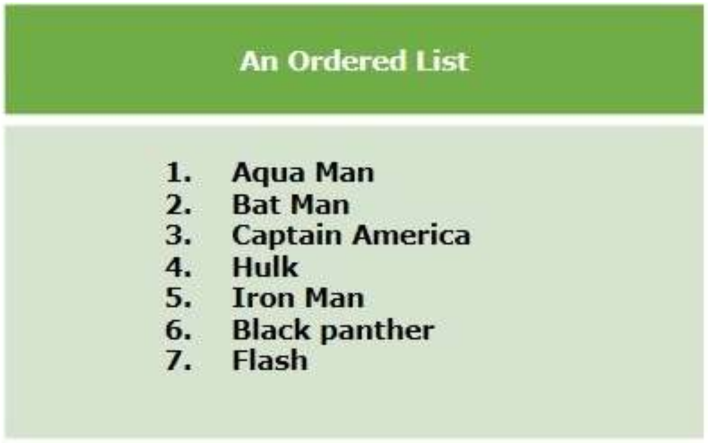
```
<!DOCTYPE html>
<html>
<head> <title>Nested Unordered Lists</title>
</head>
<body>
<h2>Example of Nested Unordered Lists</h2>
<ul>
<li>Fruits <ul>
<li>Apple</li>
<li>Banana</li>
<li>Orange</li>
```

```
</ul>
</li>
<li>Vegetables
<ul>
<li>Carrot</li>
<li>Broccoli</li>
<li>Spinach</li>
</ul>
</li>
<li>Dairy
<ul>
<li>Milk</li>
<li>Cheese</li>
<li>Yogurt</li>
</ul>
</li>
</ul>
</body>
</html>
```

Ordered HTML Lists

An ordered list is a collection of items that have a specific order or sequence. HTML ordered list is created by <ol> tag where each list item is defined by the <li> tag.

This type of ordered list is used to show the list items, where they are marked with an ordered numbered list, such as the steps of a recipe, the ranking of a leaderboard, or the chronological order of events as shown in the below figure:


An Ordered List

1. Aqua Man
2. Bat Man
3. Captain America
4. Hulk
5. Iron Man
6. Black panther
7. Flash

Creating Ordered Lists

To create an ordered list in HTML, we use the <ol> tag and nest <li> tags inside it. Each <li> element represents one item in the list. The numbering starts with 1 and is incremented by one for each successive ordered list element tagged with <li>. Like an unordered list, it also allows us to change the numbering style using the type attribute of the <ol> element.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML Ordered List</title>
</head>
<body>
  <ol>
    <li>Beetroot</li>
    <li>Ginger</li>
    <li>Potato</li>
    <li>Radish</li>
  </ol>
</body>
</html>
```

HTML Ordered List - The type Attribute

Thetype attribute for the<ol> tag is used to specify the type of marker for the ordered list in HTML. By default, the type of list numbering is numbers starting with 1, 2, 3, and so on. You can change the type of numbers by using any of the values given below:

S.No	Value & Description
1	1 It is the default type of marker.
2	I List items will be displayed with roman number marker.
3	A It will set the marker to upper case alphabets.
4	a It sets the marker to lower case alphabets.

Ordered List With Numbers

The following example demonstrates the ordered lists with numbers type marker:

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML Ordered List</title>
</head>
<body>
  <p>Ordered list with counting numbers:</p>
  <ol type="1">
    <li>Beetroot</li>
    <li>Ginger</li>
    <li>Potato</li>
    <li>Radish</li>
  </ol>
```

```
</body>
</html>
```

Ordered List With Uppercase Roman

The following example demonstrates the ordered lists with uppercase roman numbers type marker:

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML Ordered List</title>
</head>
<body>
  <p>Ordered list with uppercase roman numbers:</p>
  <ol type="I">
    <li>Aman</li>
    <li>Vivek</li>
    <li>Shrey</li>
    <li>Ansh</li> </ol></body></html>
```

Ordered List With Lowercase Roman

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML Ordered List</title>
</head>
<body>
  <p>Ordered list with lowercase roman numbers:</p>
  <ol type="i">
    <li>Aman</li>
    <li>Vivek</li>
    <li>Shrey</li>
    <li>Ansh</li>
  </ol>
</body>
```

```
</html>
```

Ordered List With Uppercase Alphabets

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML Ordered List</title>
</head>
<body>
  <p>Ordered list with uppercase alphabets:</p>
  <ol type="A">
    <li>Bus</li>
    <li>Train</li>
    <li>Plane</li>
    <li>Boat</li>
  </ol>
</body>
</html>
```

Ordered List With Lowercase Alphabets

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML Ordered List</title>
</head>
<body>
  <p>Ordered list with lowercase alphabets:</p>
  <ol type="a">
    <li>Bus</li>
    <li>Train</li>
    <li>Plane</li>
    <li>Boat</li>
  </ol>
</body>
```

```
</html>
```

HTML Ordered List - The start Attribute

By default, the numbering starts with 1, but you can change it by using the start attribute with the <ol> tag. The style attribute defines the starting numbers of the ordered list.

Syntax

The following are the different syntaxes (use cases) to define number types and the initial (starting) number for the ordered list:

<ol type="1" start="4"> - Numerals starts with 4.

<ol type="I" start="4"> - Numerals starts with IV.

<ol type="i" start="4"> - Numerals starts with iv.

<ol type="a" start="4"> - Letters starts with d.

<ol type="A" start="4"> - Letters starts with D.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML Ordered List</title>
</head>
<body>
  <ol type="i" start="4">
    <li>Beetroot</li>
    <li>Ginger</li>
    <li>Potato</li>
    <li>Radish</li>
  </ol>
</body>
</html>
```

Styling HTML Ordered List

Styling ordered lists with CSS allows customization of the appearance to match the design preferences of a website. The CSS styles can target both the list itself (<ol>) and the list items (<li>).

Example

```
<!DOCTYPE html>
<html lang="en">
<head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Styled Ordered List</title>
<style>
/* Basic Styling */
ol { color: navy; font-family: 'Arial', sans-serif; margin-left: 20px; }
li { font-size: 16px; margin-bottom: 8px; } /* Changing Numbering Style */
ol.roman { list-style-type: upper-roman; }
ol.letters { list-style-type: upper-alpha; } /* Adding Counters */
ol.counter-list { counter-reset: my-counter; }
ol.counter-list li { counter-increment: my-counter; }
ol.counter-list li::before { content: counter(my-counter) '. '; } /* Changing Text Color on Hover */
li.hover-effect:hover { color: #e44d26; }
</style>
</head>
<body>
<h1>Styled Ordered List Example</h1>
<h2>Basic Styling</h2>
<ol>
<li>Item 1</li>
<li>Item 2</li>
<li>Item 3</li>
</ol>
<h2>Changing Numbering Style</h2>
<ol class="roman">
```

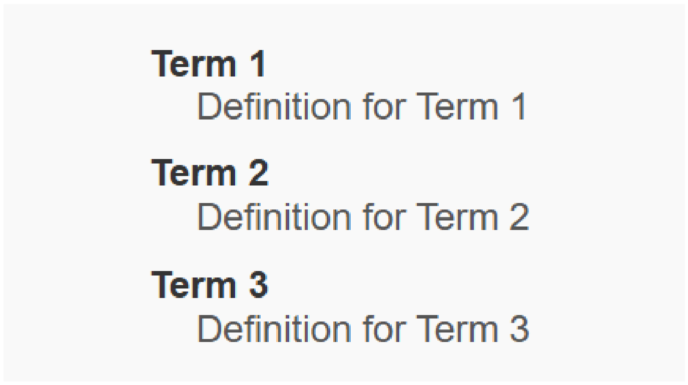
```
<li>Roman I</li>
<li>Roman II</li>
<li>Roman III</li>
</ol>
<ol class="letters">
<li>Letter A</li>
<li>Letter B</li>
<li>Letter C</li>
</ol>
<h2>Adding Counters</h2>
 <ol class="counter-list">
<li>Item</li>
<li>Item</li>
<li>Item</li>
</ol>
<h2>Changing Text Color on Hover</h2>
<ol>
<li class="hover-effect">Hover Effect 1</li>
<li class="hover-effect">Hover Effect 2</li>
<li class="hover-effect">Hover Effect 3</li>
</ol>
</body>
</html>
```

HTML - Definition Lists

A description list is defined by <dl> tag along with the <dt> and <dd> tags. Where <dt> tag defines the definition term, and <dd> tag defines the corresponding definition.

HTML Definition Lists

HTML definition lists define list items having the structure of terms and their corresponding definitions. These types of lists are used to define a listing structure where each list item (data term) contians its corresponding explanation (definition description).



Term 1
Definition for Term 1
Term 2
Definition for Term 2
Term 3
Definition for Term 3

The <dl> tag supports almost all browsers. It also supports the global attributes and event attributes. It consists of open and closing tags like <dl></dl>

Definition List Tags

The following are the HTML tags used for defining definition lists:

- <dl>: This tag defines the definition list.
- <dt>: This tag defines the description term.
- <dd>: This tag defines the corresponding description for the given definition term.

Creating Definition List

Where,

- <dl> is used as a container tag for the definition list.
- <dt> is used to define the terms that you want to define.
- <dd> is used to place the definitions for the corresponding terms.

Syntax

Below is the syntax (structure) of creating a definition list in HTML:

```
<dl>
  <dt>Term 1</dt>
  <dd>Definition for Term 1</dd>
  <dt>Term 2</dt>
  <dd>Definition for Term 2</dd>
  <dt>Term 3</dt>
  <dd>Definition for Term 3</dd>
</dl>
```

Example of Definition List

```
<!DOCTYPE html> <html> <body> <h2>Different Types Of Languages</h2> <dl>
<dt>English:</dt> <dd> English is the first world language. We can use English language
for communication in all areas like politics, media, entertainment, art etc. </dd>
<dt>Hindi:</dt> <dd> Hindi is an Indo-Aryan language spoken mostly in India. In India Hindi
is spoken as a first language by most of the people. </dd> <dt>Marathi:</dt> <dd> Marathi
is an Indo-Aryan language spoken by Marathi people of Maharashtra in India. It is a
official Language of Maharashtrian people </dd> <dt>French:</dt> <dd> French is the
official language in Canada, Central, African, Burkina, Faso, Burundi etc. </dd> </dl>
</body> </html>
```

Styling Definition Lists

```
<!DOCTYPE html>
<html>
<head>
<style>
body { font-family: Arial, sans-serif; margin: 20px; }
dl { background-color: #f9f9f9; border: 1px solid #ddd; padding: 20px; border-radius: 5px;
max-width: 400px; margin: 0 auto; }
dt { font-weight: bold; color: #333; margin-top: 10px; }
dd { margin: 0 0 10px 20px; color: #555; }
</style>
</head>
<body>
<dl>
<dt>Tutorialspoint</dt>
<dd>
Tutorialspoint provides access to a library of video courses on various prominent
technologies, aimed at helping individuals master those technologies and become
certified professionals.
</dd>
<dt>Tutorix</dt>
 <dd>
Tutorix is child company of tutorialspoint that covers NCERT-based syllabus for maths
and science. Also give a great foundation for IIT/JEE and NEET aspirants.
</dd>
```

```
</dl>
</body>
</html>
```

Nested Definition Lists

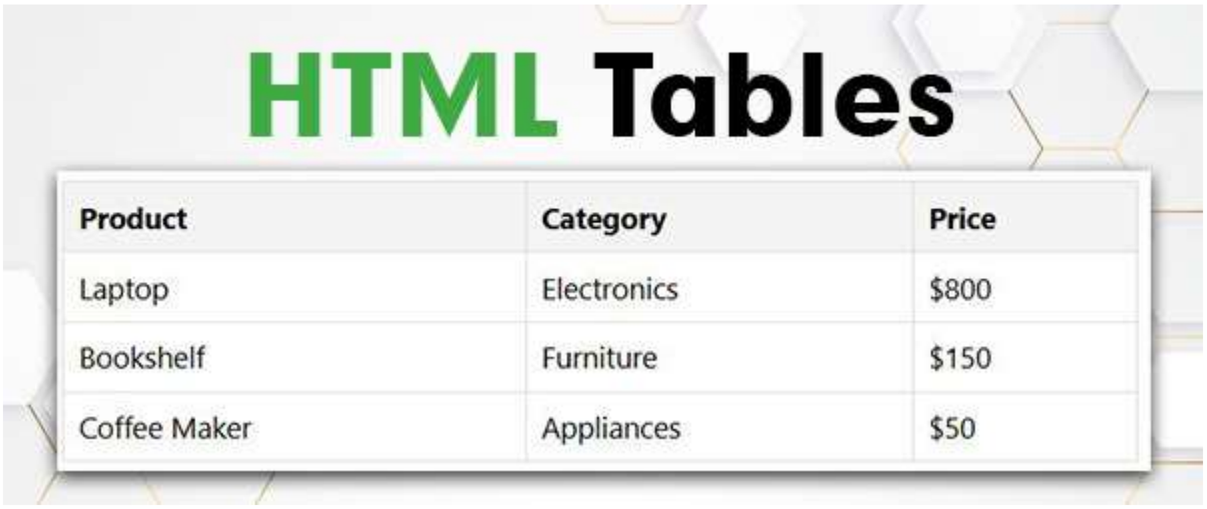
```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Nested Definition Lists Example</title>
</head>
<body>
<h2>Nested Definition Lists Example</h2>
 <dl>
<dt>Programming Languages</dt>
 <dd>
<dl>
<dt>Python</dt>
<dd>
A high-level, interpreted programming language.
</dd>
<dt>JavaScript</dt>
<dd>
A language used for web development.
</dd>
</dl>
</dd>
<dt>Web Technologies</dt>
 <dd>
<dl>
<dt>HTML</dt>
<dd>
The standard markup language for creating web pages.
</dd>
```

```
<dt>
CSS
</dt>
<dd>
Used for styling web pages.
</dd>
</dl>
</dd>
</dl>
</body>
</html>
```

HTML - Tables

HTML tables represent data, such as text, images, etc. in a structured format with rows and columns.

HTML tables offer a visual structure that aids in clarity and comprehension, making them a fundamental element in web development.



HTML Tables

Product	Category	Price
Laptop	Electronics	$800
Bookshelf	Furniture	$150
Coffee Maker	Appliances	$50

Why HTML Tables are Used?

HTML tables are used for various reasons, primarily centered around organizing and presenting data effectively. Some key purposes include –

- Structuring Data – Tables provide a coherent structure for organizing and displaying data, making it easier for users to interpret information.

- Comparative Presentation – When there is a need to compare different sets of data side by side like difference between two concepts, tables offer a clear and visually accessible format.
- Tabular Data Representation – Information that naturally fits into rows and columns, such as schedules, statistics, or pricing tables, can be well-represented using HTML tables.

Creating an HTML Table

You can create a table in HTML by using the <table> tag along with several tags that define the structure and content inside the table. The primary tags that are used with the <table> tag are <tr>, <td>, and <th>.

Creating tables in HTML involves several elements that define the structure and content. The primary tags used are <table>, <tr>, <td>, and <th>.

- HTML <table> Tag: This tag is used to create the table that wrap the rows and columns within it.
- HTML <tr> Tag: Stands for "table row" and is used to create a row within the table.
- HTML <td> Tag: Represents "table data" and is used to create standard cells within a row.
- HTML <th> Tag: Represents "table header" and is used to create header cells within a row.
-

HTML Table Structure - Rows and Columns

- Rows: Each row in an HTML table is defined using the `<tr>` tag. It contains a set of table cells (`<td>` or `<th>`), representing the individual elements within that row.
- Columns: The actual data or header information is contained within the table cells. Cells in the same position in different rows form a column.
- A table row is defined by the <tr> tag. To set table header, we use <th> tag. To insert data in table cell, use the <td> tag.
- A table in HTML consists of table cells inside rows and columns of the table.
- Table heading is defined by the <th>...</th>. Data inside the <th> are the headings of the column of a table.
- Each table cell is defined by a <td>...</td> tag. Data inside the <td> tag are the content of the table rows and columns.
- Each table row starts with a <tr> ....</tr> tag.
- We use style sheet to create border for the table.

Example

```
<!DOCTYPE html>
<html>
<body>
<table border="1">
<tr> <
th>Product</th>
<th>Category</th>
<th>Price</th>
</tr>
<tr>
<td>Laptop</td>
<td>Electronics</td>
<td>$800</td>
</tr>
<tr>
<td>Bookshelf</td>
<td>Furniture</td>
<td>$150</td>
</tr>
<tr>
<td>Coffee Maker</td>
 <td>Appliances</td>
<td>$50</td>
</tr>
</table>
</body>
</html>
```

Styling HTML Tables

You can also style an HTML table using CSS properties to give it a custom appearance. Either you can create classes to apply styles on a table, or you can simply write internal CSS properties to style the table.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
table { width: 100%; border-collapse: collapse; margin-bottom: 20px; }
th, td { border: 1px solid #ddd; padding: 8px; text-align: left; }
th { background-color: #f2f2f2; }
</style>
</head>
<body>
<h2>HTML Table</h2>
<p>This table is 3*3 cells including table header. </p>
<table>
<tr>
<th>Header 1</th>
<th>Header 2</th>
<th>Header 3</th>
 </tr>
<tr>
<td>Data 1</td>
<td>Data 2</td>
<td>Data 3</td>
</tr>
<tr>
<td>Data 4</td>
<td>Data 5</td>
<td>Data 6</td>
</tr>
</table>
</body>
</html>
```

Table Background Color and Image

You can set the background color and background image of an HTML table by using the CSS and attributes of the <table> tag.

Using Attributes

The following are the attributes that can be used with <table> tag to set the background color and/or image:

- bgcolor: The bgcolor attribute sets the table's background color.
- <table bgcolor="#f0f0f0">
- background: The background attribute sets a background image.
- <table background="image.jpg">

Using CSS Properties

Using table tag's attributes is an old (outdated) style. It is recommended that you should use CSS to style an HTML table. The background-color and background-image properties are used to set background color and image respectively.

```
table {
  background-color: #f0f0f0;
  background-image: url('image.jpg');
}
```

Example to set table's background color and image using attributes

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Table Background Color</title>
<style>
table { background-color: yellow; background-image: url('/images/test.png'); }
</style>
</head>
<body>
<table>
<tr>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
```

```
</tr>
<tr>
<td rowspan="2">Row 1 Cell 1</td> <td>Row 1 Cell 2</td>
<td>Row 1 Cell 3</td>
</tr>
<tr>
<td>Row 2 Cell 2</td>
<td>Row 2 Cell 3</td>
</tr>
<tr>
<td colspan="3">Row 3 Cell 1</td>
</tr>
</table>
</body>
</html>
```

Table Width and Height

The table's width and height can be set using either attributes or CSS properties. These values can be defined in pixels or percentages.

Using Attributes

The following attributes can set the width and height of a table:

- width: It defines the width of the table.

`<table width="80%">`

- height: It defines the height of the table.

`<table height="200">`

Using CSS

The following CSS properties can set the width and height of a table:

- width: It defines the width of the table.

`table { width: 80%; }`

- height: It defines the height of the table.

table { height: 400px; }

Example to set table's width and height using attributes

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Table Width/Height</title>
</head>
<body>
<table border="1" width="80%" height="400">
<tr>
<th>Header 1</th>
<th>Header 2</th>
</tr>
<tr>
<td>Row 1, Column 1</td>
<td>Row 1, Column 2</td>
</tr>
<tr>
<td>Row 2, Column 1</td>
<td>Row 2, Column 2</td>
</tr>
</table>
</body>
</html>
```

Example to set table's width and height using CSS

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Table Width/Height</title>
<style>
table{ width: 80%; height: 400px; }
```

```
</style>
</head>
<body>
<table border="1">
<tr>
<th>Header 1</th>
<th>Header 2</th>
</tr>
<tr>
<td>Row 1, Column 1</td>
<td>Row 1, Column 2</td>
</tr>
<tr>
<td>Row 2, Column 1</td>
<td>Row 2, Column 2</td>
</tr>
</table> </body> </html>
```

HTML Nested Tables

Nested HTML tables refer to create tables inside a table. You can create tables inside a table by using the <table> tab inside any <td> tag, it creates another table in the main table's cell.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Nested Tables</title>
</head>
<body>
<table border=1>
<tr>
<td> First Column of Outer Table </td>
```

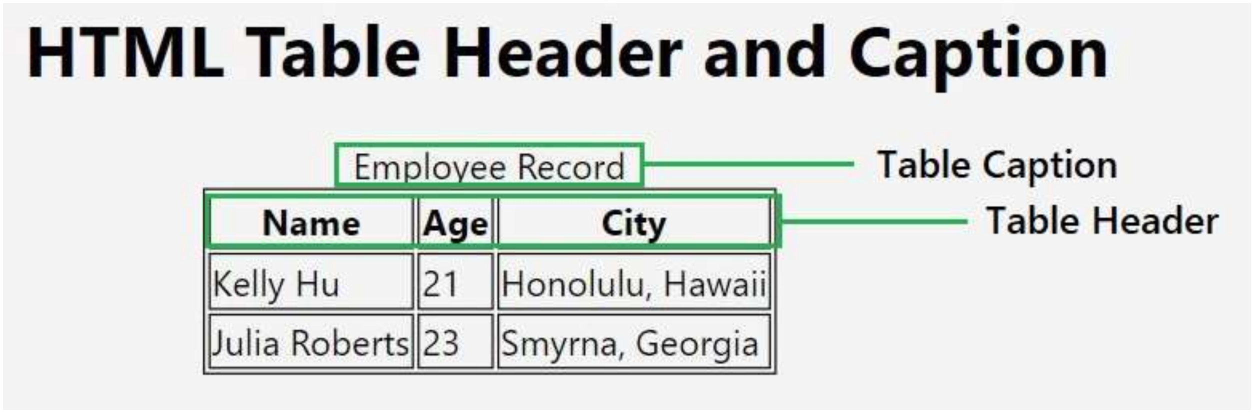
```
<td>
<table border=1> <tr>
<td> First row of Inner Table </td>
</tr>
<tr>
<td> Second row of Inner Table </td>
 </tr>
</table>
</td>
</tr>
 <tr>
<td> <table border=1> <tr>
<td> First row of Inner Table </td>
</tr>
<tr>
<td> Second row of Inner Table
</td> </tr> </table> </td> <td> First Column of Outer Table </td> </tr> </table> </body>
</html>
```

Table Headers and Captions

Headers and captions are used inside tables to organize and present data in a structured format.

The table heading is an essential part of a table, providing labels for columns. The <th> (table header) element is used to define table headings.

Captions are used in the tables to provide a title or explanation for the table. The <caption> element is placed immediately after the opening table tag.



Employee Record

Name	Age	City
Kelly Hu	21	Honolulu, Hawaii
Julia Roberts	23	Smyrna, Georgia

Syntax to Create Table's Header and Caption

```
<table>
<caption>Description of table</caption>
<tr>
  <th>heading 1</th>
  <th>heading 2</th>
  <th>heading 3</th>
</tr>
</table>
```

Define a Header Row for a Table

The <th> tag is used to represent table headings, and it is typically used within the <tr> (table row) element. Unlike the <td> (table data) tag used for regular cells, <th> is reserved for headers. In most cases, the first row of a table is designated as the header row.

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>HTML Table Header</title>
</head>
<body>
<table border="1">
<tr>
<th>Name</th>
<th>Salary</th>
</tr>
<tr>
<td>Ramesh Raman</td>
```

```
<td>5000</td>
</tr>
 <tr>
<td>Shabbir Hussein</td>
<td>7000</td>
</tr>
</table>
</body>
 </html>
```

Styling Table Headings

Styling table headings can enhance the visual appeal of a table. CSS can be applied to <th> elements to customize their appearance. In the following example, a simple CSS style is added to the <style> tag within the <head> section to modify the background color and text alignment of the table headings.

Example

```
<!DOCTYPE html>
<html lang="en">
 <head>
<title>Styled HTML Table Header</title>
<style>
th { background-color: #4CAF50; color: white; text-align: left; padding: 8px; }
</style>
</head>
<body>
<table border="1">
<tr>
<th>Name</th>
<th>Salary</th>
</tr> <tr>
```

```
<td>Ramesh Raman</td>
<td>5000</td> </tr>
<tr>
<td>Shabbir Hussein</td>
<td>7000</td>
</tr>
</table>
</body>
 </html>
```

Using Header Cells in Any Row

While it's common to use <th> in the first row of a table, you can utilize it in any row based on your requirements. This flexibility allows for the creation of complex tables with multiple header rows or headers interspersed within the table.

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Styled HTML Table Header</title>
<style>
th { background-color: #4CAF50; color: white; text-align: left; padding: 8px; }
</style>
</head>
<body>
<table border="1">
<tr> <th>Name</th> <th>Salary</th> </tr>
<tr> <td>Ramesh Raman</td> <td>5000</td> </tr>
<tr> <th>Additional Details</th> <th>Specialization</th> </tr>
 <tr> <td>Technical Lead</td> <td>Web Development</td> </tr>
</table>
```

```
 </body>
</html>
```

Table Header Using <thead> Element

The <thead> tag is used to group table header cells so that a combined CSS style can be applied to header cells.

The <thead> tag is typically placed within the <table> element and contains one or more <tr> elements, each of which, in turn, contains <th> elements representing column headers.

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>HTML Table Header</title>
</head>
<body>
<table border=1>
<thead>
<tr> <th>Column 1</th> <th>Column 2</th> <th>Column 3</th> </tr> </thead>
<!-- Table body goes here -->
</table>
</body> </html>
```

You can style HTML tables by using the CSS. Table styling helps you to customize the normal appearance of the elements like borders, cell padding, text alignment, background colors, and more to create a well-formatted table on a webpage.

The following are some of the table stylings in HTML:

- Collapsed Border Table
- Horizontal Zebra Striped Table
- Text Alignment in Table Cells
- Vertical Zebra Striped Table
- Table with Combined Vertical and Horizontal Zebra Stripes
- Table with Horizontal Dividers

- Hoverable Table Rows

Collapsed Border Table

You have the flexibility to manage the space between table borders by manipulating the 'border-collapse' property. This property determines how adjacent table cell borders interact, and adjusting it allows you to control the spacing or gap between the borders within your table.

By setting 'border-collapse' to "collapse", borders will merge, eliminating any spacing, while "separate" allows you to control the spacing using the 'border-spacing' property, providing a more customized appearance to your table layout.

Example

```
<!DOCTYPE html>
 <html>
<head>
<style>
#table1 { border-collapse: separate; }
#table2 { border-collapse: collapse; }
</style>
</head>
<body>
<table id="table1" border="1">
<tr> <th>Name</th> <th>Salary</th> </tr>
<tr> <td>Ramesh Raman</td> <td>5000</td> </tr>
<tr> <td>Shabbir Hussein</td> <td>7000</td> </tr>
</table>
<br />
<table id="table2" border="1">
<tr> <th>Name</th> <th>Salary</th> </tr>
 <tr> <td>Ramesh Raman</td> <td>5000</td> </tr>
<tr> <td>Shabbir Hussein</td> <td>7000</td> </tr>
 </table> </body> </html>
```

Horizontal Zebra Striped Table

The Zebra Stripes - Horizontal technique involves styling table rows with alternating colors, enhancing the visual appeal and readability of the table.

The :nth-child(even) selector targets every second row, and a background color is applied to create a striped effect.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
 tr:nth-child(even) { background-color: #8a83de; }
</style>
</head>
<body>
<table border="1">
<table border="1">
<tr> <th>Name</th> <th>Salary</th> </tr>
<tr> <td>Ramesh Raman</td> <td>5000</td> </tr>
<tr> <td>Shabbir Hussein</td> <td>7000</td> </tr>
</table> </table> </body> </html>
```

Text Alignment in Table Cells

You can align the text within your table cells horizontally and vertically using the text-align and vertical-align properties.

Example

```
<!DOCTYPE html>
 <html>
<head>
<style>
 td, th { text-align: center; vertical-align: middle; }
</style>
</head>
<body>
<table border="1">
<tr> <th>Name</th> <th>Salary</th> </tr> <tr> <td>Ramesh Raman</td> <td>5000</td> </tr>
<tr> <td>Shabbir Hussein</td> <td>7000</td> </tr> </table> </body> </html>
```

Vertical Zebra Striped Table

The Zebra Stripes - Vertical technique enhances table readability by applying alternating styles to every other column. This is achieved using the :nth-child(even) selector for both table data (td) and table header (th) elements.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
td:nth-child(even), th:nth-child(even) { background-color: #D6EEEE; }
</style>
</head>
<body>
<table border="1">
<tr> <th>Name</th> <th>Salary</th> </tr>
<tr> <td>Ramesh Raman</td> <td>5000</td> </tr>
<tr> <td>Shabbir Hussein</td> <td>7000</td> </tr>
</table> </body> </html>
```

Table with Combined Vertical & Horizontal Zebra Stripes

You can achieve a captivating visual effect by combining both horizontal and vertical zebra stripe patterns on your table. This involves applying alternating styles to both rows (:nth-child(even)) and columns (td:nth-child(even), th:nth-child(even)).

To enhance this effect, consider adjusting the color transparency using the rgba() function, creating an engaging and aesthetically pleasing overlap of stripes for a unique and visually interesting outcome.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
tr:nth-child(even) { background-color: rgba(150, 212, 212, 0.4); }
th:nth-child(even),
td:nth-child(even) { background-color: rgba(212, 150, 192, 0.4); }
```

```
</style>
</head>
<body>
<table border="1">
<tr> <th>Name</th> <th>Salary</th> </tr>
<tr> <td>Ramesh Raman</td> <td>5000</td> </tr>
<tr> <td>Shabbir Hussein</td> <td>7000</td> </tr>
</table>
</body>
</html>
```

Table with Horizontal Dividers

You can enhance the visual structure of your table by incorporating horizontal dividers. Achieve this effect by styling each '<tr>' element with a bottom border.

This CSS customization provides a clear separation between rows, contributing to improved table clarity and a more organized presentation of tabular data.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
table { border-collapse: collapse; }
tr { border-bottom: 5px solid #ddd; }
th, td { padding: 10px; /* Add padding for better visibility */ }
</style>
</head>
<body>
<table border="1">
<tr> <th>Name</th> <th>Salary</th> </tr>
 <tr> <td>Ramesh Raman</td> <td>5000</td> </tr>
<tr> <td>Shabbir Hussein</td> <td>7000</td> </tr>
```

```
</table>
</body>
 </html>
```

Hoverable Table Rows

You can improve user interaction by employing the ':hover' selector, which highlights table rows when users hover over them. This enhances the visual feedback, making the table more interactive and user-friendly.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
tr:hover { background-color: #D6EEEE; }
</style>
</head> <
body>
<table border="1">
<tr> <th>Name</th> <th>Salary</th> </tr>
<tr> <td>Ramesh Raman</td> <td>5000</td> </tr>
<tr> <td>Shabbir Hussein</td> <td>7000</td> </tr>
</table>
</body>
</html>
```

HTML - Frames

HTML frames are used to divide your browser window into multiple sections where each section can load a separate HTML document independently. A collection of frames in the browser window is known as a frameset. The window is divided into frames in a similar way the tables are organized: into rows and columns.

The <frame> tag is no longer recommended as it is not supported by HTML5. Instead of using this tag, we can use the <iframe> or <div> with CSS to achieve the similar effects.

Syntax

```
<frameset rows="50%,50%">
```

```
  <frame name="top" src="link/to/frame1" />
  <frame name="bottom" src="link/to/frame2" />
</frameset>
```

Where the rows attribute of frameset defines the division of the window into horizontal sections. In this case, the window is divided into two rows, each taking up 50% of the available height.

Examples of HTML Frames

To make frames on a page we use <frameset> tag instead of <body>
tag. The <frameset> tag defines how to divide the window into frames.
The rows attribute of <frameset> tag defines horizontal frames and cols attribute defines vertical frames. Each frame is indicated by <frame> tag and it defines which HTML document shall open into the frame.

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML Frames</title>
</head>
<frameset rows="10%,80%,10%">
  <frame name="top" src="/html/top_frame.htm" />
  <frame name="main" src="/html/main_frame.htm" />
  <frame name="bottom" src="/html/bottom_frame.htm" />
  <noframes>
    <body>
      Your browser does not support frames.
    </body>
  </noframes>
</frameset>
```

```
</html>
```

Creating vertical Frames

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML Frames</title>
</head>
<frameset cols="25%,50%,25%">
  <frame name="left" src="/html/top_frame.htm" />
  <frame name="center" src="/html/main_frame.htm" />
  <frame name="right" src="/html/bottom_frame.htm" />
  <noframes>
    <body>
      Your browser does not support frames.
    </body>
  </noframes>
</frameset></html>
```

Frame's name and target Attributes

One of the most popular uses of frames is to place navigation bars in one frame and then load main pages into a separate frame.

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML Target Frames</title>
</head>
<frameset cols="200, *">
  <frame src="/html/menu.htm" name="menu_page" />
  <frame src="/html/main.htm" name="main_page" />
  <noframes>
<body>
  Your browser does not support frames.
</body>
```

```
  </noframes>
</frameset>
</html>
```

Following is the content of menu.htm file

```
<!DOCTYPE html>
<html>
<body bgcolor="#4a7d49">
  <a href="https://www.google.com" target="main_page">
    Google
  </a>
  <br /><br />
  <a href="https://www.microsoft.com" target="main_page">
    Microsoft
  </a>
  <br /><br />
  <a href="https://news.bbc.co.uk" target="main_page">
    BBC News</a></body></html>
<!DOCTYPE html>
<html>
<body bgcolor="#b5dcb3">
  <h3>
    This is main page and content from any link
    will be displayed here.
  </h3>
  <p>
    So now click any link and see the result.
  </p>
</body>
</html>
```

Attributes of frameset Tag

Attributes	Description
cols	Specifies how many columns are contained in the frameset and the size of each column. You can specify the width of each column in one of four ways. • Absolute values in pixels. For example to create three vertical frames, use *cols="100, 500,100"*. • A percentage of the browser window. For example to create three vertical frames, use *cols="10%, 80%,10%"*. • Using a wildcard symbol. For example to create three vertical frames, use *cols="10%, *,10%"*. In this case wildcard takes remainder of the window. • As relative widths of the browser window. For example to create three vertical frames, use *cols="3*,2*,1*"*. This is an alternative to percentages. You can use relative widths of the browser window. Here the window is divided into sixths: the first column takes up half of the window, the second takes one third, and the third takes one sixth.
rows	This attribute works just like the cols attribute and takes the same values, but it is used to specify the rows in the frameset. For example to create two horizontal frames, use *rows="10%, 90%"*. You can specify the height of each row in the same way as explained above for columns.
border	This attribute specifies the width of the border of each frame in pixels. For example border="5". A value of zero means no border.
frameborder	This attribute specifies whether a three-dimensional border should be displayed between frames. This attribute takes value either 1 (yes) or 0 (no). For example frameborder="0" specifies no border.
framespacing	This attribute specifies the amount of space between frames in a frameset. This can take any integer value. For example framespacing="10" means there should be 10 pixels spacing between each frames.

HTML <frame> Tag Attributes

Attribute	Description
src	This attribute is used to give the file name that should be loaded in the frame. Its value can be any URL. For example,

	src="/html/top_frame.htm" will load an HTML file available in html directory.
name	This attribute allows you to give a name to a frame. It is used to indicate which frame a document should be loaded into. This is especially important when you want to create links in one frame that load pages into an another frame, in which case the second frame needs a name to identify itself as the target of the link.
frameborder	This attribute specifies whether or not the borders of that frame are shown; it overrides the value given in the frameborder attribute on the <frameset> tag if one is given, and this can take values either 1 (yes) or 0 (no).
marginwidth	This attribute allows you to specify the width of the space between the left and right of the frame's borders and the frame's content. The value is given in pixels. For example marginwidth="10".
marginheight	This attribute allows you to specify the height of the space between the top and bottom of the frame's borders and its contents. The value is given in pixels. For example marginheight="10".
noresize	By default you can resize any frame by clicking and dragging on the borders of a frame. The noresize attribute prevents a user from being able to resize the frame. For example noresize="noresize".
scrolling	This attribute controls the appearance of the scrollbars that appear on the frame. This takes values either "yes", "no" or "auto". For example scrolling="no" means it should not have scroll bars.
longdesc	This attribute allows you to provide a link to another page containing a long description of the contents of the frame. For example longdesc="framedescription.htm"

Disadvantages of Frames

There are few drawbacks with using frames, so it's never recommended to use frames in your webpages.

- Some smaller devices cannot cope with frames often because their screen is not big enough to be divided up.
- Sometimes your page will be displayed differently on different computers due to different screen resolution.
- The browser's *back button* might not work as the user hopes.
- There are still few browsers that do not support frame technology.

HTML iframes

HTML Iframe is used to display a nested webpage (a webpage within a webpage). The HTML <iframe> tag defines an inline frame, hence it is also called as an Inline frame.

An HTML iframe embeds another document within the current HTML document in the rectangular region.

The webpage content and iframe contents can interact with each other using JavaScript.

Iframe Syntax

An HTML iframe is defined with the <iframe> tag:

```
<iframe src="URL"></iframe>
```

Here, "src" attribute specifies the web address (URL) of the inline frame page.

Set Width and Height of iframe

You can set the width and height of iframe by using "width" and "height" attributes. By default, the attributes values are specified in pixels but you can also set them in percent. i.e. 50%, 60% etc.

Example: (Pixels)

```
<!DOCTYPE html>
<html>
<body>
<h2>HTML Iframes example</h2>
<p>Use the height and width attributes to specify the size of the iframe:</p>
<iframe src="https://www.SardarAzeem.com/" height="300" width="400"></iframe>

</body>
</html>
```

Example: (Percentage)

```
<!DOCTYPE html>
<html>
```

```
<body>
<h2>HTML Iframes</h2>
<p>You can use the height and width attributes to specify the size of the iframe:</p>

<iframe src="https://www.SardarAzeem.com/" height="50%" width="70%"></iframe>

</body>
</html>
```

You can also use CSS to set the height and width of the iframe.

Example:

```
<!DOCTYPE html>
<html>
<body>
<h2>HTML Iframes</h2>
<p>Use the CSS height and width properties to specify the size of the iframe:</p>
<iframe src="https://www.SardarAzeem.com/" style="height:300px;width:400px"></iframe>
</body>
</html>
```

Remove the border of iframe

By default, an iframe contains a border around it. You can remove the border by using <style> attribute and CSS border property.

Example:

```
<!DOCTYPE html>
<html>
<body>
<h2>Remove the Iframe Border</h2>
<p>This iframe example doesn't have any border</p>
<iframe src="https://www.SardarAzeem.com/" style="border:none;"></iframe>
</body>
</html>
```

You can also change the size, color, style of the iframe's border.

Example:

```
<!DOCTYPE html>

<html>

<body>

<h2>Custom Iframe Border</h2>

<iframe src="https://www.SardarAzeem.com/" style="border:2px solid tomato;"></ifr
ame>

</body>

</html>
```

Iframe Target for a link

You can set a target frame for a link by using iframe. Your specified target attribute of the link must refer to the name attribute of the iframe.

Example:

```
<!DOCTYPE html>

<html>

<body>


<h2>Iframe - Target for a Link</h2>

<iframe height="300px" width="100%" src="new.html" name="iframe_a"></iframe>

<p><a href="https://www.SardarAzeem.com" target="iframe_a">JavaSardarAzeem.co
m</a></p>

<p>The name of iframe and link target must have same value else link will not open
as a frame. </p>


</body>

</html>
```

```
<!DOCTYPE html>

<html>

<head>

<style>
```

```
p{ font-size: 50px;
color: red;}
</style>
</head>
<body style="background-color: #c7f15e;">
<p>This is a link below the ifarme click on link to open new iframe. </p>
</body>
</html>
```

Embed YouTube video using iframe

You can also add a YouTube video on your webpage using the <iframe> tag. The attached video will be played at your webpage and you can also set height, width, autoplay, and many more properties for the video.

Following are some steps to add YouTube video on your webpage:

- Goto YouTube video which you want to embed.
- Click on SHARE ➦ under the video.
- Click on Embed <> option.
- Copy HTML code.
- Paste the code in your HTML file
- Change height, width, and other properties (as per requirement).
-

Example

```
<iframe width="550" height="315" src="https://www.youtube.com/embed/JHq3pL4cd
y4" frameborder="0" allow="accelerometer; autoplay; encrypted-
media; gyroscope; picture-in-
picture" allowfullscreen style="padding:20px;"></iframe>
        <iframe width="550" height="315" src="https://www.youtube.com/embed/O5hSh
UO6wxs" frameborder="0" allow="accelerometer; autoplay; encrypted-
media; gyroscope; picture-in-picture" style="padding:20px;">></iframe>
```

Attributes of <iframe>

Attribute name	Value	Description
allowfullscreen		If true then that frame can be opened in full screen.

height	Pixels	It defines the height of the embedded iframe, and the default height is 150 px.
name	text	It gives the name to the iframe. The name attribute is important if you want to create a link in one frame.
frameborder	1 or 0	It defines whether iframe should have a border or not. (Not supported in HTML5).
Width	Pixels	It defines the width of embedded frame, and default width is 300 px.
src	URL	The src attribute is used to give the path name or file name which content to be loaded into iframe.
sandbox		This attribute is used to apply extra restrictions for the content of the frame
	allow-forms	It allows submission of the form if this keyword is not used then form submission is blocked.
	allow-popups	It will enable popups, and if not applied then no popup will open.
	allow-scripts	It will enable the script to run.
	allow-same-origin	If this keyword is used then the embedded resource will be treated as downloaded from the same source.
	srcdoc	The srcdoc attribute is used to show the HTML content in the inline iframe. It overrides the src attribute (if a browser supports).
scrolling		It indicates that browser should provide a scroll bar for the iframe or not. (Not supported in HTML5)
	auto	Scrollbar only shows if the content of iframe is larger than its dimensions.
	yes	Always shows scroll bar for the iframe.
	no	Never shows scrollbar for the iframe.

HTML - Meta Tags

HTML <meta> tag lets us specify metadata, which is additional important information about a document, in a variety of ways. The META elements can be used to include name and content pairs describing properties of the HTML document, such as author, expiry date, a list of keywords, document author, etc.

HTML <meta> tag can be used to provide extra information. It's a self-closing element, meaning it doesn't require a closing tag but carries information within its attributes. You can include one or more meta tags in your document based on what information you want to keep in your document, but in general, meta tags do not impact the physical appearance of the document, so from the appearance point of view, it does not matter if you include them or not.

Adding Metadata to Web Pages Using Meta Tags

The following metadata can be added using the <meta> tag:

- Specifying Keywords
- Document Revision Date
- Page Redirection
- Setting Author Name
- Document Description
- Document Refreshing
- Setting Cookies

Specify Character Set

You can use the <meta> tag to specify important keywords related to the document, and later these keywords are used by the search engines while indexing your webpage for searching purposes.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Meta Tags Example</title>
  <meta name="keywords" content="HTML, Meta Tags, Metadata" />
</head>
```

```
<body><p>Hello HTML5!</p></body></html>
```

Document Description

You can use the <meta> tag to give a short description about the document. This again can be used by various search engines while indexing your webpage for searching purposes.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Meta Tags Example</title>
  <meta name="keywords" content="HTML, Meta Tags, Metadata" />
  <meta name="description" content="Learning about Meta Tags." />
</head>
<body>
  <p>Hello HTML5!</p>
</body>
</html>
```

Document Revision Date

You can use the <meta> tag to give information about the last time the document was updated. This information can be used by various web browsers while refreshing your webpage.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Meta Tags Example</title>
  <meta name="keywords" content="HTML, Meta Tags, Metadata" />
  <meta name="description" content="Learning about Meta Tags." />
  <meta name="revised" content="Tutorialspoint, 3/7/2014" />
</head>
<body>
  <p>Hello HTML5!</p>
```

```
</body></html>
```

Document Refreshing

The <meta> tag can be used to specify a duration after which your web page will keep refreshing automatically.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Meta Tags Example</title>
  <meta name="keywords" content="HTML, Meta Tags, Metadata" />
  <meta name="description" content="Learning about Meta Tags." />
  <meta name="revised" content="Tutorialspoint, 3/7/2014" />
  <meta http-equiv="refresh" content="5" />
</head>
<body>
  <p>Hello HTML5!</p></body></html>
```

Page Redirection

You can use the <meta> tag to redirect your page to any other webpage. You can also specify a duration if you want to redirect the page after a certain number of seconds.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Meta Tags Example</title>
  <meta name="keywords" content="HTML, Meta Tags, Metadata" />
  <meta name="description" content="Learning about Meta Tags." />
  <meta name="revised" content="Tutorialspoint, 3/7/2014" />
  <meta http-equiv="refresh" content="5; url=http://www.tutorialspoint.com" />
</head>
<body>
  <p>Hello HTML5!</p>
</body></html>
```

Setting Cookies

Cookies are data stored in small text files on your computer, and it is exchanged between a web browser and a web server to keep track of various information based on your web application needs.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Meta Tags Example</title>
  <meta name="keywords" content="HTML, Meta Tags, Metadata" />
  <meta name="description" content="Learning about Meta Tags." />
  <meta name="revised" content="Tutorialspoint, 3/7/2014" />
  <meta http-equiv="cookie" content="userid=xyz; expires=Wednesday, 08-Aug-15
23:59:59 GMT;" />
</head><body> <p>Hello HTML5!</p></body></html>
```

Setting Author Name

You can set an author name on a web page using a <meta> tag. Author name be specified by assigning the "author" value to the "name" attribute.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Meta Tags Example</title>
  <meta name="keywords" content="HTML, Meta Tags, Metadata" />
  <meta name="description" content="Learning about Meta Tags." />
  <meta name="author" content="Mahnaz Mohtashim" />
</head>
<body>
  <p>Hello HTML5!</p>
</body>
</html>
```

Specify Character Set

You can use the <meta> tag to specify the character set used within the webpage. By default, Web servers and Web browsers use ISO-8859-1 (Latin1) encoding to process Web pages.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Meta Tags Example</title>
  <meta name="keywords" content="HTML, Meta Tags, Metadata" />
  <meta name="description" content="Learning about Meta Tags." />
  <meta name="author" content="Mahnaz Mohtashim" />
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
  <p>Hello HTML5!</p></body></html>
```

Example

To serve the static page with traditional Chinese characters, the webpage must contain a <meta> tag to set Big5 encoding:

```
<!DOCTYPE html>
<html>
<head>
  <title>Meta Tags Example</title>
  <meta name="keywords" content="HTML, Meta Tags, Metadata" />
  <meta name="description" content="Learning about Meta Tags." />
  <meta name="author" content="Mahnaz Mohtashim" />
  <meta http-equiv="Content-Type" content="text/html; charset=Big5" />
</head>
<body>
  <p>Hello HTML5!</p>
</body>
```

```
</html>
```

HTML - Forms

An HTML form is a webpage section usually used for collecting data from the users and then sent to a server for further processing.

HTML Forms

HTML forms are collections of interactive controls and various input types, such as text, numbers, email, password, radio buttons, checkboxes, buttons, etc., that collect user information. HTML forms are created by using the HTML <form> tag. All user input-related tags are placed inside the <form> tag.

Syntax

```
<form>
  <!-- Form elements-->
</form>
```

The following syntax contains all necessary elements:

```
<form action="url" method="method_type" target="target_value" enctype="enctype_value">
   <!-- Form elements-->
</form>
```

Why Use HTML Forms?

HTML forms are used to collect user information from the webpage and send it to the server. The common uses for HTML forms are:

- Creating registration forms so that users can sign up with their information and authenticate further to access the functionalities of the websites/web applications.
- Collect data through the different types of surveys, feedback, etc.
- Uploading the images, resumes, or any other type of files.

Creating an HTML Form

To create an HTML form, use the <form> element along with the other required elements based on the information you want to collect, such as input boxes, buttons, radio buttons, checkboxes, etc. These elements are known as form controls (form elements).

Example

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Form Example</title>
</head>
<body>
<h1>HTML Form Example</h1>
<form> <
!-- Text Input -->
<label for="name"><strong>Name:</strong></label> <input type="text" id="name" name="name" placeholder="Enter your name" required> <br/><br/>
<!-- Radio Buttons -->
<label><strong>Gender:</strong></label> <input type="radio" id="male" name="gender" value="male">
<label for="male">Male</label> <input type="radio" id="female" name="gender" value="female">
<label for="female">Female</label> <br/><br/> <!-- Checkboxes --> <label><strong>Hobbies:</strong></label> <input type="checkbox" id="reading" name="hobbies" value="reading">
<label for="reading">Reading</label> <input type="checkbox" id="traveling" name="hobbies" value="traveling">
<label for="traveling">Traveling</label> <input type="checkbox" id="sports" name="hobbies" value="sports">
<label for="sports">Sports</label> <br/><br/>
<!-- Submit Button -->
<button type="submit">Submit</button>
</form> </body> </html>
```

HTML Form with Redirection

In the previous example, we designed a form that accepts user input but doesn't process the data. In this example, users will be redirected to Tutorialspoint's HTML Tutorial upon form submission. The redirection only happens if both the first name and last name fields are filled out; otherwise, the form prompts the user to provide the required information.

Example

```
<!DOCTYPE html>
<html>
<head> <meta charset="utf-8">
<title>Sample HTML Form</title>
</head>
<body>
<!-- Start of the form element -->
<form action="" method="post">
<!-- Form controls -->
<label for="first_name">First Name:</label> <input type="text" id="first_name" name="first_name" required /> <br><br>
<label for="last_name">Last Name:</label> <input type="text" id="last_name" name="last_name" required /> <br><br>
<input type="submit" value="Submit">
</form>
</body>
</html>
```

Form Elements

There is a list of elements that can be used within the form element. All the elements are briefly described below:

1. The <form> Element

HTML <form> tag is used to create the <form> element. This element is the container for all other form elements. The form element does not create the form; it's the container that holds the other form elements.

Example

```
<form>.....</form>
```

2. The <input> Element

HTML <input> tag is an essential element of form control for gathering user input from websites. We can use this tag to create an input element.

Example

```
<input type = ".."/>
```

3. The <label> Element

HTML <label> tag is used to create a label element that represents a caption for an item in a UI (user interface), or to add labels to a form control like text, textarea, checkbox, radio button, etc.

Example

```
<label>.......</label>
```

4. The <legend> Element

HTML <legend> tag is the element's first child and specifies the caption or title for the <fieldset> tag.

Example

```
<legend>.......</legend>
```

5. HTML <select> Element

HTML <select> tag is used to create the dropdown in HTML forms. We can use this tag to create a dropdown anywhere we want.

Example

```
<select>....</select>
```

6. The <button> Element

HTML <button> tag is an interactive element used to create a button in HTML.

Example

```
<button>Button</button>
```

7. The <fieldset> Element

HTML <fieldset> tag is used to group several controls within a web form. By using the <fieldset> tag and <legend> tag, a form can be much easier for users to understand.

Example

```
  <fieldset>....</fieldset>
```

8. The <datalist> Element

HTML <datalist> tag contains a set of <option> elements that represent recommended options available to choose from among others.

Example

<datalist>....</datalist>

9. The <output> Element

HTML <output> tag is a flexible and underused component that enables programmers to dynamically show the outcomes of calculations or scripts inside the content.

Example

<output> Results... </output>

10. The <option> Element

HTML <option> tag defines either the elements of the data list for autocomplete, specified by the <datalist> tag, or the items of a drop-down list, defined by the <select> tag.

Example

<option>.....</option>

11. The <optgroup> Element

HTML <optgroup> tag is used in the <select> element to group together relevant <option> elements.

Example

```
<optgroup>
  <option>..</option>
    .
    .
</optgroup>
```

12. The <textarea> Element

HTML <textarea> tag is used to represent a multiline plain-text editing control.

Example

<textarea>.......</textarea>

Form Attributes

HTML form attributes provide specific functionalities, such as redirection to other web pages, auto-completion of text, etc.

The below table lists out some of the common form attributes:

Attribute	Description
action	It is used to specify a URL that processes the form submission.
method	It is used to define which HTTP method to use when submitting the form.
target	It is used to specify where to open the linked document.
autocomplete	It allows you to set whether the autocomplete for the form should be on or off.
enctype	It is used to specify how the form input data should be encoded before sending it to the server.
novalidate	It defines that while submitting the form, the form data should not be validated in an HTML document.

Styling HTML Forms

You can customize the appearance of HTML forms and their elements by using the CSS to match your website theme or to make it more appealing.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Form</title>
<style>
body { font-family: 'Segoe UI', Arial, sans-serif; }
form { width: 100%; max-width: 400px; background-color: #e8f5e9; padding: 20px;
border: 1px solid #ccc; border-radius: 5px; }
legend { font-size: 1.2rem; font-weight: bold; margin-bottom: 10px; }
label { display: block; margin-bottom: 5px; font-size: 0.9rem; }
input[type="text"], input[type="email"], input[type="password"], textarea { width: 100%;
padding: 8px; margin-bottom: 15px; border: 1px solid #ccc; border-radius: 4px; box-
sizing: border-box; }
textarea { resize: none; }
input[type="submit"] { width: 100%; padding: 10px; font-size: 1rem; color: #fff;
background-color: #04af2f; border: none; border-radius: 4px; cursor: pointer; }
input[type="submit"]:hover { background-color: #039325; }
</style>
</head>
```

```
<body>
<form>
<fieldset>
<legend>Registration Form</legend>
<label for="firstName">First Name</label> <input type="text" id="firstName"
name="FirstName" />
<label for="lastName">Last Name</label> <input type="text" id="lastName"
name="LastName" />
<label for="email">Email ID</label> <input type="email" id="email" name="email" />
<label for="password">Enter Your Password</label> <input type="password"
id="password" name="password" />
<label for="confirmPass">Confirm Your Password</label> <input type="password"
id="confirmPass" name="confirmPass" />
<label for="address">Address</label> <textarea id="address"
name="address"></textarea>
<input type="submit" value="Submit" /> </fieldset> </form> </body> </html>
```

Form Attributes

HTML form attributes provide different functionalities, such as redirection to other web pages, auto-completion of text, specifying data validation rules, controlling the behavior of form submissions, etc.

Following is a list of the most frequently used form attributes −

- action
- method
- target
- autocomplete
- enctype
- novalidate

The action Attribute

The action attribute of the <form> element transmits the user's input to a backend script for processing. A form is of no use unless it processes the information provided by the user. Therefore, it is important to pass the URL of a program to the action attribute. Note that the formaction attribute can override the value of the action attribute.

Example

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title> The action Attribute </title>
</head>
<body>
<!-- Start of the form element -->
<form action = "https://pictacademy.com">
<!-- to take input -->
Name: <input type = "text" name = "your_name" required/> <br><br>
Email: <input type = "email" name = "mail" required/> <br><br>
<!-- to submit the data -->
<input type = "submit"> </form> </body> </html>
```

The method Attribute

The method attribute determines which HTTP method should be used by the browser while uploading the form information. The most commonly used methods are as follows:

S.No	Values & Description
1	GET It is the default method for form submission, which means if we don't specify the method name explicitly, the form will use the GET method to send data.
2	POST It is used to send form data inside HTTP request body. It is safer than GET method.

It is not recommended to use the GET method while sending sensitive information like credit/debit card numbers and passwords because it exposes the submitted data in the URL.

Example

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title> The method Attribute </title>
</head>
<body>
<!-- Start of the form element -->
<form action = "https://pictacademy.com" method = "post">
<!-- to take input -->
Name: <input type = "text" name = "your_name" required/> <br><br>
Email: <input type = "email" name = "mail" required/> <br><br>
<!-- to submit the data -->
<input type = "submit">
</form> </body> </html>
```

The target Attribute

The target attribute determines the target window or frame where the result of the script will be displayed after submitting the form. The default target is the current window. The target attribute accepts the following values:

S.No.	Values & Description
1	_self It opens the response in the same frame as it was clicked.
2	_blank It opens the response in the new window or tab.
3	_parent It opens the response in the parent frame.
4	_top It opens the response in the full body of window.
5	framename It opens the response in the named iframe.

Example

```
<!DOCTYPE html>
<html>
<head> <meta charset="utf-8">
<title> The target Attribute </title>
</head>
<body>
<!-- Start of the form element -->
<form action = "https://pictacademy.com" target = "_self">
<!-- to take input -->
Name: <input type = "text" name = "your_name" required/> <br><br>
Email: <input type = "email" name = "mail" required/> <br><br>
<!-- to submit the data -->
<input type = "submit"> </form> </body> </html>
```

The novalidate Attribute

The novalidate is a Boolean attribute that indicates the form does not need any kind of validation. The term validation refers to the process of verifying the correctness of user input based on predefined conditions. This attribute, when applied, exempts the form from such checks, allowing user inputs to bypass these conditions.

If Boolean attributes like novalidate are present on an HTML element, it specifies true, and in the case of absence, false is assumed. They do not accept any values.

Example

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title> The novalidate Attribute </title>
</head>
<body>
<!-- Start of the form element -->
<form action = "https://pictacademy.com" target = "_blank" autocomplete="off" method =
"get" novalidate>
```

```
<!-- to take input -->
Name: <input type = "text" name = "your_name" required/> <br><br>
Email: <input type = "email" name = "mail" required/> <br><br>
<!-- to submit the data -->
<input type = "submit">
</form> </body> </html>
```

The autocomplete Attribute

The autocomplete attribute of HTML predicts and suggests the subsequent input based on the initial characters entered in the input field. This attribute primarily has two states, namely on and off.

S.No.	Values & Description
1	on By default, the autocomplete attribute is set to on, enabling the autocomplete functionality.
2	off The autocomplete attribute can be toggled to off to disable this feature as per the requirements of the web application.

The enctype Attribute

We use the enctype attribute to specify how the browser encodes the data before it sends it to the server. Its possible values are —

S.No.	Values & Description
1	application/x-www-form-urlencoded This is the standard method most forms use in simple scenarios.
2	mutlipart/form-data This is used when you want to upload binary data in the form of files like images, Word files etc.
3	text/plain It only encodes the spaces into + symbol.

HTML - Form Controls

HTML form controls (elements) are the elements used within the <form> element to collect the user information.

Form Controls (Elements)

The form elements create controls for the user interaction within the webpage; these elements are also termed as form controls. The form elements enable users to enter information for the server-side processing. The nature of interaction with the server can vary depending on the type of control used while creating the form. For example, radio buttons are typically used to accept gender information.

We have used several common form controls in previous discussions; we will now dive into a more detailed exploration of these elements.

There are different types of form controls that we can use to collect data using HTML form:

- Text Input Controls
- Checkboxes Control
- Radio Buttons Control
- Select Box Control
- File Select Box
- Button Control
- Hidden Form Control
- Datetime Controls
- Date Control
- Month Control
- Week Control
- Time Control
- Number Control
- Range Control
- Email Control
- URL Control

Text Input Controls

The text input controls are further divided into three main categories –

- Single-line Text Input Control
- Password Input Control
- Multi-line Text Input Control

Single-line Text Input Control

The single-line text input control is used for items that require only one line of user input, such as search boxes or names. They are created using the <input> tag.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Text Input Control</title>
</head>
<body>
  <form >
    First name: <input type = "text" name = "first_name" />
    <br><br>
    Last name: <input type = "text" name = "last_name" />
  </form>
</body>
</html>
```

Password Input Control

The password input control is also a single-line text input, but it masks the character as soon as a user enters it. They are also created using the HTML <input> tag, but the type attribute is set to password:

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Password Input Control</title>
</head>
<body>
  <form >
    User ID : <input type = "text" name = "user_id" />
    <br><br>
    Password: <input type = "password" name = "password" />   </form></body></html>
```

Multiple-line Text Input Control

The multiple-line text input control is used when the user is required to give details that may be longer than a single sentence. Multi-line input controls are created using the HTML <textarea> tag.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Multiple-Line Input Control</title>
</head>
<body>
  <form>
    Description : <br />
    <textarea rows = "5" cols = "50" name = "description">
      Enter description here...
    </textarea>
  </form>
</body>
</html>
```

Checkboxes Control

Checkboxes are used when more than one option is required to be selected. They are also created using the <input> tag, but the type attribute is set to checkbox.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Checkbox Control</title>
</head>
<body>
  <form>
    <input type = "checkbox" name = "maths" value = "on"> Maths
    <input type = "checkbox" name = "physics" value = "on"> Physics
```

```
</form>
</body>
</html>
```

Radio Buttons Control

Radio buttons are used when out of many options, just one option is required to be selected. They are also created using the <input> tag, but the type attribute is set to radio.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Radio Box Control</title>
</head>
<body>
  <form>
    <input type = "radio" name = "subject" value = "maths"> Maths
    <input type = "radio" name = "subject" value = "physics"> Physics
  </form>
</body>
</html>
```

Select Box Control

A select box provides features to list down various options in the form of drop-down list, from where a user can select one or more options.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Select Box Control</title>
</head>
<body>
  <form>
    <select name = "dropdown">
```

```
      <option value = "Maths" selected>Maths</option>
      <option value = "Physics">Physics</option>
      <option value = "Chemistry">Chemistry</option>
    </select>
  </form>
</body>
</html>
```

File Select Box

If we want to allow a user to upload a file to our website, we will need to use a file upload box, also known as a file select box. This is also created using the <input> element, but the type attribute is set to file.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>File Upload Box</title>
</head>
<body>
  <form>
    <input type = "file" name = "fileupload" accept = "image/*" />
  </form>
</body>
</html>
```

Button Control

There are various ways in HTML to create clickable buttons. We can create a clickable button using the <input> tag by setting its type attribute to button.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>File Upload Box</title>
</head>
```

```
<body>
  <form>
    <input type = "submit" name = "submit" value = "Submit" />
    <input type = "reset" name = "reset" value = "Reset" />
    <input type = "button" name = "ok" value = "OK" />
    <input type = "image" name = "imagebutton" src = "/html/images/logo.png" />
  </form>
</body>
</html>
```

Hidden Form Control

The hidden form controls are used to hide data inside the page, which later on can be pushed to the server. This control hides inside the code and does not appear on the actual page. For example, the following hidden form is being used to keep the current page number. When a user clicks next page, then the value of the hidden control will be sent to the web server, and there it will decide which page will be displayed next based on the passed current page.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>File Upload Box</title>
</head>
<body>
  <form>
    <p>This is page 10</p>
    <input type = "hidden" name = "pagename" value = "10" />
    <input type = "submit" name = "submit" value = "Submit" />
    <input type = "reset" name = "reset" value = "Reset" />
  </form>
</body>
</html>
```

Datetime Controls

In HTML, the datetime control represents date and time (year, month, day, hour, minute, second, and fractions of a second) together, encoded according to ISO 8601 with the time zone set to UTC. If we use the datetime-local, it will display date and time with no time zone information.

Example

```
<!DOCTYPE html>
<html>
<body>
  <form action = "/cgi-bin/html5.cgi" method = "get">
    Date and Time : <input type = "datetime" name = "newinput" />
    <input type = "submit" value = "submit" />
  </form>
</body>
</html>
```

Date Control

The HTML date control is used to specify a date (year, month, day) encoded according to ISO 8601.

Example

```
<!DOCTYPE html>
<html>
<body>
  <form action = "/cgi-bin/html5.cgi" method = "get">
    Date : <input type = "date" name = "newinput" />
    <input type = "submit" value = "submit" />
  </form>
</body>
</html>
```

Month Control

In HTML, the month control is used to display a date consisting of only a year and a month encoded according to ISO 8601.

Example

```
<!DOCTYPE html>
<html>
<body>
  <form action = "/cgi-bin/html5.cgi" method = "get">
    Month : <input type = "month" name = "newinput" />
    <input type = "submit" value = "submit" />
  </form>
</body>
</html>
```

Week Control

As the name suggests, the week control displays a date consisting of only a year and a week number encoded according to ISO 8601.

Example

```
<!DOCTYPE html>
<html>
<body>
  <form action = "/cgi-bin/html5.cgi" method = "get">
    Week : <input type = "week" name = "newinput" />
    <input type = "submit" value = "submit" />
  </form>
</body>
</html>
```

Time Control

The HTML time control specifies the hours, minutes, seconds, and fractional seconds encoded according to ISO 8601.

Example

```
<!DOCTYPE html>
<html>
<body>
  <form action = "/cgi-bin/html5.cgi" method = "get">
```

```
    Time : <input type = "time" name = "newinput" />
    <input type = "submit" value = "submit" />
  </form>
</body>
</html>
```

Number Control

The number control accepts only numerical values. The step attribute specifies the precision, and its default value is 1.

Example

```
<!DOCTYPE html>
<html>
<body>
  <form action = "/cgi-bin/html5.cgi" method = "get">
    Select Number : <input type = "number" min = "0" max = "10" step "1"
      value = "5" name = "newinput" />
    <input type = "submit" value = "submit" />
  </form>
</body>
</html>
```

Range Control

The range type is used for input fields that should contain a value from a range of numbers.

Example

```
<!DOCTYPE html>
<html>
<body>
  <form action = "/cgi-bin/html5.cgi" method = "get">
    Select Range : <input type = "range" min = "0" max = "10" step "1"
      value = "5" name = "newinput" />
    <input type = "submit" value = "submit" />
  </form></body></html>
```

Email Control

The email control accepts only email value. This type is used for input fields that should contain an e-mail address. If you try to submit a simple text, it forces you to enter only an email address in email@example.com format.

Example

```
<!DOCTYPE html>
<html>
<body>
  <form action = "/cgi-bin/html5.cgi" method = "get">
    Enter email : <input type = "email" name = "newinput" />
    <input type = "submit" value = "submit" />
  </form>
</body>
</html>
```

URL Control

The HTML URL control accepts only URL values. This type is used for input fields that should contain a URL address. If you try to submit a simple text, it forces you to enter only a URL address, either in the http://www.example.com format or in the http://example.com format.

Example

```
<!DOCTYPE html>
<html>
<body>
  <form action = "/cgi-bin/html5.cgi" method = "get">
    Enter URL : <input type = "url" name = "newinput" />
    <input type = "submit" value = "submit" />
  </form>
</body>
</html>
```

HTML - Input Attributes

The HTML input attributes define the characteristics and behavior of the <input> element. These input attributes are used with the different types of input fields, such as text, email, password, date, number, and so forth. Note that the input element is used to create interactive controls for the web-based forms so that it can accept data from the user.

The <input> element requires only an opening tag, and it will work only if we add it in between the <form> tags. In this tutorial, we are going to explore the attributes that are used with the <input> element.

The attributes of the <input> element are as follows —

- type and name
- value
- size
- maxlength
- readonly
- disabled
- min and max
- accept and multiple
- placeholder
- required
- autofocus
- list

The 'type' and 'name' Attributes

The type attribute indicates the type of input control, like text, password, email, and so on. The name attribute of an input element assigns an identifier to the form control that enables the server to recognize and retrieve the value.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>The type and name Attributes</title>
</head>
<body>
  <form >
```

```
    First name: <input type = "text" name = "first_name" />
    <br><br>
    Last name: <input type = "text" name = "last_name" />
  </form>
</body>
</html>
```

The 'value' Attribute

The value attribute is used to provide an initial value inside the input control.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>The value Attribute</title>
</head>
<body>
  <form >
    First name: <input type = "text" name = "first_name" value = "first name..." />
    <br><br>
    Last name: <input type = "text" name = "last_name" value = "last name..."/>
  </form>
</body>
</html>
```

The 'size' Attribute

The size attribute allows you to specify the width of the text-input control in terms of characters. The default size is 20 characters.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>The size Attribute</title>
</head>
```

```
<body>
  <form >
    First name: <input type = "text" name = "first_name" size = "40" />
    <br><br>
    Last name: <input type = "text" name = "last_name" size = "40"/>
  </form>
</body>
</html>
```

The 'maxlength' Attribute

The maxlength attribute allows you to specify the maximum number of characters a user can enter into the text box.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>The maxlength Attribute</title>
</head>
<body>
  <form >
    First name: <input type = "text" name = "first_name" />
    <br><br>
    Last name: <input type = "text" name = "last_name" />
    <br><br>
    Contact: <input type = "text" name = "phone" maxlength = "10"/>
  </form>
</body>
</html>
```

The 'readonly' Attribute

The readonly attribute of an input field indicates the field as read-only. Although the content of a read-only field cannot be altered, users can still select it and copy the text. Also, the value of a read-only field is included when the form is submitted.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>The readonly Attribute</title>
</head>
<body>
  <form >
    Emp. Name: <input type = "text" name = " your_name" value = "your name..."/>
    <br><br>
    Emp. Email: <input type = "text" name = "mail" value = "your email..."/>
    <br><br>
    Organization: <input type = "text" name = "organization" value = "Tutorialspoint"
readonly/>
  </form>
</body>
</html>
```

The 'disabled' Attribute

The disabled attribute of an input field indicates the field as disabled. Unlike readonly, the value of a disabled field will not be included when the form is submitted.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>The disabled Attribute</title>
</head>
<body>
  <form >
    Emp. Name: <input type = "text" name = "your_name" value = "your name..."/>
    <br><br>
    Emp. Email: <input type = "email" name = "mail" value = "your email..."/>
    <br><br>
    Organization: <input type = "text" name = "organization" value = "Tutorialspoint"
disabled/>
  </form>
</body>
</html>
```

The 'min' and 'max' Attributes

The min and max attributes determine the minimum and maximum values, respectively, of an input field like number, date, week, and so on. If we use them together, they will allow users to enter an input within a predefined range.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>The min and max Attribute</title>
</head>
<body>
<form >
Emp. Name: <input type = "text" name = "your_name" value = "your name..."/> <br><br>
Emp. Email: <input type = "email" name = "mail" value = "your email..."/> <br><br>
Organization: <input type = "text" name = "organization" value = "Tutorialspoint"
readonly/> <br><br> Working Hrs: <input type = "number" name = "working_hours"
min="3" max="8"/> </form> </body> </html>
```

The 'accept' and 'multiple' Attributes

The accept attribute specifies the types of files that the server will take in. If we use the multiple attribute, it will allow the users to upload more than one file.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>The accept and multiple Attributes</title>
</head>
<body>
  <form>
```

```
    <input type = "file" name = "fileupload" accept = "image/*" multiple />
  </form>
</body>
</html>
```

The 'placeholder' Attribute

The placeholder attribute of an input field, like text, search, and email, briefly outlines the desired value of the field. Its predefined value is displayed in the input field until the user begins to enter their own value.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>The placeholder Attribute</title>
</head>
<body>
  <form>
    Emp. Name: <input type = "text" name = "your_name"/>
    <br><br>
    Emp. Email: <input type = "email" name = "mail" placeholder =
"example@email.com"/>
  </form>
</body>
</html>
```

The 'required' Attribute

The required attribute in an input field like text, search, password, and email signifies that the field must contain some values for the form to be successfully submitted. In other words, it indicates the mandatory field.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>The required Attribute</title>
```

```
</head>
<body>
  <form >
    <p>The * Star represents mandatory field</p>
    Emp. Name: <input type = "text" name = "your_name" required/>*
    <br><br>
    Emp. Email: <input type = "email" name = "mail" placeholder = "example@email.com"
required/>*
    <br><br>
    <input type = "submit">
  </form>
</body>
</html>
```

The 'autofocus' Attribute

The autofocus attribute in an input field ensures that the field must be selected automatically once the webpage loads completely. It implies that the cursor will be positioned to the specified input field. In cases where multiple elements use the autofocus attribute, only the first element will acquire the focus.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>The autofocus Attribute</title>
</head>
<body>
  <form >
    Emp. Name: <input type = "text" name = "your_name" autofocus/>
    <br><br>
    Emp. Email: <input type = "email" name = "mail" placeholder = "example@email.com"
/>
    <br><br>
    <input type = "submit">
  </form>
```

```
</body>
</html>
```

The 'list' Attribute

The list attribute defines a set of predefined options for an <input> element, which are defined within a <datalist> element. The <input> element uses a specific string as an ID to create a link to the corresponding <datalist> element.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>The list Attribute</title>
</head>
<body>
  <form >
    Emp. Name: <input type = "text" name = "your_name" autofocus/>
    <br><br>
    Emp. Email: <input type = "email" name = "mail" placeholder = "example@email.com"
/>
    <br><br>
    Location —
    <input list="location" name="cities">
      <datalist id = "location">
        <option value="Banglore">
        <option value="Hyderabad">
        <option value="Patna">
        <option value="Delhi">
      </datalist>
    <input type = "submit">
  </form>
</body>
</html>
```

HTML Multimedia / HTML Video

The HTML <video> element embeds and shows a video on the webpage. You can embed any type of video content on the webpage by using the <video> element.

HTML <video> Element

The <video> element is used to enable video playback support within a web page. It works very similarly to the <img> element, as it also requires adding the path or URL of the video within the src attribute. The HTML supports only MP4, WebM, and Ogg video formats. The <video> element also supports audio; however, the <audio> element is more suitable for that purpose.

Embedding Videos in HTML

To embed a video inside a web page, you need to set the src attribute inside the <video> tag that specifies the path or URL for the video. A web page serves content to a wide variety of users with various types of browsers. Each browser supports different video formats (MP4, WebM, and Ogg) as mentioned earlier. Therefore, we can supply all the formats that HTML supports by including multiple <source> tags. Let the browser decide which format is more suitable for video playback.

Syntax

```
<video width="640" height="360" controls>
  <source src="video-file.mp4" type="video/mp4">
  <source src="video-file.ogg" type="video/ogg">
  Your browser does not support the video tag.
</video>
```

The controls Attribute

You can also enable the built-in controls for controlling audio and video playback for the users (if needed) with the help of the controls attribute. It provides an interface that enables users to manage video playback functions such as volume adjustment, video navigation (forward and backward), and play or pause operations.

Example to Embed a Video

```
<!DOCTYPE html>
<html>
<head>
   <title>HTML Video Element</title>
</head>
<body>
```

```
  <p>Playing video using video element</p>
  <p>The browser is responsible for determining the appropriate format to use.</p>
  <video width="450" height="250" controls>
    <source src="/html/media/video/sampleTP.webm" type="video/webm">
    <source src="/html/media/video/sampleTP.mp4" type="video/mp4">
    <source src="/html/media/video/sampleTP.ogv" type="video/ogg">
    <p>Sorry, video element is not supported!</p>
  </video>
</body>
</html>
```

Customizing Video Display Size

To set (adjust) the dimensions of the video display area, also known as the viewport, you can use the height and width attributes of the <video> element. These attributes define the height and width of the video viewport in pixels.

Note that the video will adjust its aspect ratio (e.g., 4:3 and 16:9) to align with the specified height and width. Therefore, it is advisable to match the dimensions of the viewport for a better user experience.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML Video Element</title>
</head>
<body>
  <p>Playing video using video element</p>
  <video width="450" height="250" controls>
    <source src="/html/media/video/sampleTP.mp4" type="video/mp4">
  </video>
</body>
</html>
```

HTML Video autoplay and loop Attributes

You can configure the video to play automatically in a loop by using the autoplay and loop attributes.

Remember, a few browsers like Chrome 70.0 do not support the autoplay feature unless the muted attribute is used. Therefore, it is recommended to use autoplay and muted attributes together.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML Video Element</title>
</head>
<body>
  <p>Playing video using video element</p>
  <video width="450" height="250" autoplay muted loop>
    <source src="/html/media/video/sampleTP.mp4" type="video/mp4">
  </video>
</body>
</html>
```

Setting a Video Thumbnail

You can pass a URL of an image that works as a thumbnail for the video within the poster attribute of the <video> element. It will display the specified image until the video starts playing.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML Video Element</title>
</head>
<body>
  <p>Playing video using video element</p>
  <video width="450" height="250" controls muted poster ="tutorials_point.jpg" >
    <source src="/html/media/video/sampleTP.mp4" type="video/mp4">
  </video>
</body>
</html>
```

HTML Multimedia / HTML Audio

The HTML <audio> element embeds an audio file to the webpage. You can add an audio player inside a webpage using the <audio> element.

The <audio> element is used to enable the support of audio files within a web page. We can include multiple sources of audio; however, the browser will choose the most appropriate file automatically. Most of the attributes of <video> element is also compatible with the <audio> element. The most frequently used attributes of the HTML audio element are controls, autoplay, loop, muted, and src.

Attributes of <audio> Elements

Attribute	Description
controls	This attribute adds built-in audio controls for play, pause, and volume.
autoplay	This attribute allows playing the audio automatically when the page is loaded.
loop	This attribute allows looping of the audio.
muted	This attribute mutes the audio by default when the page is loaded.
preload	This attribute specifies how the audio should be preloaded by the browser.
src	This attribute specifies the path to the audio file.

Embedding an Audio in HTML

You can embed an audio player using the <audio> tag by specifying the audio file path. The audio file path can be defined either by setting the src attribute or by including the <source> tag.

The current HTML5 draft specification does not specify which audio formats browsers should support in the audio tag. But the most commonly used audio formats are ogg, mp3, and wav. Therefore, it is also possible to supply all these formats by using multiple <source> tags within the <audio> element.

Syntax

```
<audio controls>
  <source src="file_path" type="audio/mpeg">
  Your browser does not support the audio element.
</audio>
```

- <audio>: The main element to embed an audio player.
- controls: Controls to add play, pause, and volume functionalities.
- <source>: It specifies the audio file name (along with its path) and the audio file's format.
- Fallback text: The text to be displayed if the browser doesn't support the <audio> element. In the above syntax, it will display: "Your browser does not support the audio element."

Example of HTML Audio Element

```
<!DOCTYPE html>
<html>
<body>
  <p>Working with audio element</p>
  <audio controls>
    <source src= "/html/media/audio/sample_3sec_audio.mp3" type = "audio/mp3" />
    <source src= "/html/media/audio/sample_3sec_audio.wav" type = "audio/wav" />
    <source src= "/html/media/audio/sample_3sec_audio.ogg" type = "audio/ogg" />
    Your browser does not support the <audio> element.
  </audio>
</body>
</html>
```

Using autoplay, muted, and loop Attributes in Audio Player

We can also configure the audio to play automatically in a loop by using the autoplay and loop attributes. Remember, the Chrome browser does not support the autoplay feature unless the muted attribute is used. Therefore, it is recommended to use autoplay and muted attributes together.

Example

```
<!DOCTYPE html>
<html>
<body>
  <p>Working with audio element</p>
  <audio controls autoplay muted loop>
    <source src= "/html/media/audio/sample_3sec_audio.mp3" type = "audio/mp3" />
    <source src= "/html/media/audio/sample_3sec_audio.wav" type = "audio/wav" />
    <source src= "/html/media/audio/sample_3sec_audio.ogg" type = "audio/ogg" />
    Your browser does not support the <audio> element.
  </audio>
</body>
</html>
```

HTML - Embed Multimedia

In the previous two chapters, we have used the <audio> and <video> elements to add music and videos into our web page. There are other alternative ways to add videos, sounds, images, or any other external content to the website by using HTML tags <embed>and<object>. These tags cause the browser itself to include controls for the multimedia automatically:

- HTML <embed> tag is used to embed external content such as images, videos, and web applications. It is often used for embedding content like Flash movies or audio/video players.
- HTML <object> tag is used to embed various types of external resources, including images, videos, PDFs, and other web resources. It can render multiple types of files.

Syntax

Embed tag:

```
<embed src = "url_of_resource">
```

Object tag:

```
<object data="url_of_resource" type="typeOfResource">
```

Attributes of <embed> Tag

Attribute	Description
width	Width attribute is used describe width of the embedded file in pixels.
height	Height of the embedded file in pixels.
title	It is used to label the content. The title should describe the content.
src	URL of the file to be embedded.
type	It indicates the type of input like mp4 and mp3.

Attributes of <object> Tag

Attributes	Description
data	The location or path of the resource is passed into data attribute.
type	It indicates the type of resource.
height	It signifies the height of the resource display area.
width	It signifies the width of the resource display area.
form	Its value is the id of a form. The form attribute shows which object is associated with the form.
name	It specify a unique name for the object.
usemap	Specifies a URL of a client-side image map to be used with the object.

Examples of HTML Multimedia Embedding

Here are a few examples that illustrate how to render multimedia content in a webpage using the <embed> and <object> tags:

- Embedding a Video Using <embed> Element
- Embedding an Audio Using <embed> Element
- Render a PDF Using <object> Element
- Render an HTML Document Inside Webpage

Embedding a Video Using <embed> Element

To embed an external video file inside the web page, we can pass the path of the video file into the src attribute of the <embed> element. The supported video formats are MP4, WebM, and Ogg. We dont need to use the controls attribute, as the <embed> tag provides the controls automatically depending on the type of media. The controls attribute allows users to manage the video playback functions.

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML embed Tag</title>
</head>

<body>
  <h1>Playing video using embed tag</h1>
  <embed src="/html/media/video/sampleTP.mp4"
       type="video/mp4"
       width="450"
       height="250">
  </embed>
</body>
</html>
```

Embedding an Audio Using <embed> Element

To add a soundtrack to the webpage, we can pass the path of the audio file into the src attribute by mentioning the type of audio. The supported audio formats are ogg, mp3, and wav.

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML embed Tag</title>
</head>

<body>
  <h1>Playing audio using embed tag</h1>
  <embed src = "/html/media/audio/sample_3sec_audio.mp3"
       type = "audio/mp3"
       width="450"
       height="250">
  </embed>
```

```
</body>
</html>
```

Render a PDF Using <object> Element

HTML 4 introduces the <object> element, which offers an all-purpose solution to generic object inclusion. The <object> element allows HTML authors to specify everything required by an object for its presentation by a user agent.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>PDF Embed Example</title>
</head>

<body>
    <h1>Embedding a PDF Document</h1>
    <p>Here is an embedded PDF document:</p>
    <object data="html/pdf/index.pdf"
          type="application/pdf"
          width="300"
          height="200">
    </object>
</body>
</html>
```

Render an HTML Document Inside Webpage

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>HTML Embed Example</title>
</head>
<body>
  <h1>Embedding an HTML Document</h1>
  <p>Here is an embedded HTML document:</p>
  <object data="html/index.htm"
```

```
        type="text/html"
        width="500"
        height="400">
    alt : <a href="html/index.htm">
      test.htm
    </a>
  </object>
</body>
</html>
```

HTML YouTube Videos

Converting videos to different formats can be difficult and time-consuming.

An easier solution is to let YouTube play the videos in your web page.

To play your video on a web page, do the following:

- Upload the video to YouTube
- Take a note of the video id
- Define an <iframe> element in your web page
- Let the src attribute point to the video URL
- Use the width and height attributes to specify the dimension of the player
- Add any other parameters to the URL

Example 1

```
<!DOCTYPE html>
<html>
<body>
<iframe width="420" height="345"src="https://www.youtube.com/embed/tgbNymZ7vqY">
</iframe>
</body>
</html>
```

Example 2

```
<!DOCTYPE html>
<html>
<body>
```

```
<iframe width="420" height="345"
src="https://www.youtube.com/embed/tgbNymZ7vqY?autoplay=1&mute=1">
</iframe>
</body>
</html>
```

YouTube Loop

Add playlist=videoID and loop=1 to let your video loop forever.

loop=0 (default) - The video will play only once.

loop=1 - The video will loop (forever).

```
<!DOCTYPE html>
<html>
<body>
<iframe width="420" height="345"
src="https://www.youtube.com/embed/tgbNymZ7vqY?playlist=tgbNymZ7vqY&loop=1">
</iframe>
</body>
</html>
```

YouTube Controls

Add controls=0 to NOT display controls in the video player.

controls=0 - Player controls does not display.

controls=1 (default) - Player controls is displayed.

```
<!DOCTYPE html>
<html>
<body>
<iframe width="420" height="345"
src="https://www.youtube.com/embed/tgbNymZ7vqY?controls=0">
</iframe>
</body>
</html>
```

HTML - Head Elements

HTML head elements define metadata like the title, character set, links to external stylesheets, and other details. This information does not display on the webpage but is helpful for the search engines and browsers. The head elements are placed inside the <head> tag.

The following are the commonly used head elements:

- <title> Element
- <meta> Element
- <base> Element
- <link> Element
- <style> Element
- <script> Element

HTML <title> Element

The HTML <title> tag is used for specifying the title of the HTML document. The title must describe the content of the web page, and its format should be text only. It appears in the title bar of the browser's tab.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML Title Tag Example</title>
</head>
<body>
  <p>Describing the use of title tag</p>
</body>
</html>
```

HTML <meta> Element

The HTML <meta> tag is used to provide metadata about an HTML document. The metadata is nothing but additional information about the web page, including page expiry, page author, list of keywords, page description, and so forth. This information is further used for the purpose of search engine optimization. Remember, the metadata specified by the <meta> tag is not displayed on the web page, but it is machine-readable. Its most commonly used attributes are name, content, charset, and http-equiv.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML Meta Tag Example</title>
  <!-- Provide list of keywords -->
  <meta name="keywords" content="C, C++, Java, PHP, Perl, Python">
  <!-- Provide description of the page -->
  <meta name="description" content="Simply Easy Learning by Tutorials Point">
  <!-- Author information -->
  <meta name="author" content="Tutorials Point">
  <!-- Page content type -->
  <meta http-equiv="content-type" content="text/html; charset=UTF-8">
  <!-- Page refreshing delay -->
  <meta http-equiv="refresh" content="30">
  <!-- Page expiry -->
  <meta http-equiv="expires" content="Wed, 21 June 2006 14:25:27 GMT">
  <!-- Tag to tell robots not to index the content of a page -->
  <meta name="robots" content="noindex, nofollow">
</head>
<body>
  <p>Describing the use of HTML meta tag</p>
</body>
</html>
```

HTML <base> Element

The HTML <base> tag is used for specifying the base URL for all relative URLs in a page, which means all the other URLs will be concatenated into the base URL while locating the given item. We are allowed to use only one base element in our HTML document. The most frequently used attributes of the tag are hrefandtarget.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML Base Tag Example</title>
  <base href = "index.htm/" />
</head>
<body>
  <img src="/images/logo.png" alt="Logo Image"/>
  <a href="/html/index.htm" title="HTML Tutorial"/>HTML Tutorial</a>
</body>
</html>
```

HTML <link> Element

In HTML, the <link> tag is used to specify relationships between the current webpage and another external resource. The source of external resources is placed inside thehref attribute. The other attributes of the tag are rel, type, and media. Its most common use is to embed stylesheets into the HTML document.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML link Tag Example</title>
  <link rel="stylesheet" type="text/css" href="/css/style.css">
</head>
<body>
 <p>It is an example of linking stysheet to the current HTML document.</p>
</body>
</html>
```

HTML <style> Element

The HTML <style> tag is used to specify styles either for the whole HTML document or for a particular element. Its most common attributes are title and media.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML style Tag Example</title>
  <base href="http://www.tutorialspoint.com/" />
  <style>
    .myclass{
      background-color: #aaa;
      padding: 10px;
    }
  </style>
</head>
<body>
  <p class="myclass">Hello, World!</p>
</body>
</html>
```

HTML <script> Element

The HTML <script> tag is used to include either an external script file or to define an internal script for the HTML document. The script is an executable code that performs some action.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML script Tag Example</title>
  <base href="http://www.tutorialspoint.com/" />
  <script type="text/JavaScript">
    function Hello(){
      alert("Hello, World");
    }
  </script>
```

```
</head>
<body>
  <input type="button" onclick="Hello();" name="ok" value="OK"  />
</body>
</html>
```

HTML - Favicon

HTML favicon stands for "favorite icon". It is a small-sized image that displays in the browser's tab just before the page title. Favicon is defined by using the <link> tag with the "rel=icon" attribute.

What is a HTML Favicon?

A favicon is a small image that represents your website and helps users identify it among multiple tabs, bookmarks, and search results. It can be in various formats, such as ICO, PNG, GIF, JPEG, or SVG, but ICO is the most widely supported format. If you have ever visited a website and noticed a small icon next to the page title in your browser's tab, you have seen a favicon.

How To Add a Favicon in HTML

You can add a favicon to a webpage by using the <link> tag with the rel attribute set to "icon". The <link> tag is a head element, so it must be placed within the <head> tag.

Syntax

```
<head>
  <link rel="icon" href="path_to_favicon.ico" type="image/x-icon">
</head>
```

Steps to Add Favicon on Webpage

To add a favicon, we need to follow these simple steps mentioned below −

Step 1 − Create or choose an image for your favicon. Its common size could be 16x16 pixels or 32x32 pixels. There are a few online tools available that help us in creating a favicon, such as "Favicon.io" and "ionos.com".

Step 2 − Save and upload the favicon image to the website directory. Make sure the image is in a format that browsers can recognize, such as PNG, GIF, or ICO.

Step 3 − Now use the <link> element, which tells the browser where to find the favicon image. Remember, the <link> tag comes inside the header part, i.e., <head> tag of the HTML document.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Tutorialspoint</title>
  <link rel = "icon" type = "image/png" href = "images/faviconTP.png">
</head>
<body>
  <h1>Adding Favivon</h1>
  <p>This is an example of including favicon to the web page.</p>
  <p> Favicon will be displayed in the browser tab to the left of the page title.</p>
</body>
</html>
```

Different Favicons for Different Pages on a Website?

Yes, different favicons can be added for different pages on a website. You need to define the favicon images for different pages using the <link> tag (as discussed above) inside the head.

Example

For Webpage 1:

```
<head>
 <title>Page Title 1</title>
 <link rel="icon" href="favicon1.ico" type="image/x-icon">
</head>
```

For Webpage 2:

```
<head>
 <title>Page Title 2</title>
 <link rel="icon" href="favicon2.ico" type="image/x-icon">
</head>
```

HTML Website Layout Methods

Creating Website Layouts

Creating a website layout is the activity of positioning the various elements that make a web page in a well-structured manner and give appealing look to the website.

You have seen most websites on the internet usually display their content in multiple rows and columns, formatted like a magazine or newspaper to provide the users a better reading and writing environment. This can be easily achieved by using the HTML tags, such as <table>, <div>, <header>, <footer>, <section>, etc. and adding some CSS styles to them.

HTML Table Based Layout

Table provides the simplest way for creating layouts in HTML. Generally, this involves the process of putting the contents such as text, images, and so on into rows and columns.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>HTML Table Layout</title>
</head>
<body style="margin:0px;">
<table style="width:100%; border-collapse:collapse; font:14px Arial,sans-serif;">
<tr> <td colspan="2" style="padding:10px 20px; background-color:#acb3b9;"> <h1 style="font-size:24px;">Tutorial Republic</h1> </td> </tr>
<tr style="height:170px;"> <td style="width:20%; padding:20px; background-color:#d4d7dc; vertical-align: top;"> <ul style="list-style:none; padding:0px; line-height:24px;"> <li><a href="#" style="color:#333;">Home</a></li> <li><a href="#" style="color:#333;">About</a></li> <li><a href="#" style="color:#333;">Contact</a></li> </ul> </td>
<td style="padding:20px; background-color:#f2f2f2; vertical-align:top;"> <h2>Welcome to our site</h2> <p>Here you will learn how to create websites...</p> </td>
 </tr>
<tr> <td colspan="2" style="padding:5px; background-color:#acb3b9; text-align:center;"> <p>copyright &copy; pictacademy.com</p> </td> </tr>
</table>
 </body> </html>
```

HTML Div Based Layout

Using the <div> elements is the most common method of creating layouts in HTML. The <div> (stands for division) element is used for marking out a block of content, or set of other elements inside an HTML document. It can contain further other div elements if required.

```
<!DOCTYPE html>
<html lang="en">
<head> <meta charset="utf-8">
<title>HTML Div Layout</title>
<style>
body { font: 14px Arial,sans-serif; margin: 0px; }
.header { padding: 10px 20px; background: #acb3b9; }
.header h1 { font-size: 24px; }
.container { width: 100%; background: #f2f2f2; }
.nav, .section { float: left; padding: 20px; min-height: 170px; box-sizing: border-box; }
.nav { width: 20%; background: #d4d7dc; }
.section { width: 80%; }
.nav ul { list-style: none; line-height: 24px; padding: 0px; }
.nav ul li a { color: #333; }
.clearfix:after { content: "."; display: block; height: 0; clear: both; visibility: hidden; }
.footer { background: #acb3b9; text-align: center; padding: 5px; }
</style>
</head>
<body>
<div class="container">
<div class="header"> <h1>Tutorial Republic</h1> </div>
<div class="wrapper clearfix">
<div class="nav"> <ul> <li><a href="#">Home</a></li> <li><a href="#">About</a></li> <li><a
href="#">Contact</a></li> </ul> </div>
<div class="section"> <h2>Welcome to our site</h2> <p>Here you will learn how to
create websites...</p> </div> </div>
<div class="footer">
```

```
<p>copyright &copy; tutorialrepublic.com</p>
</div> </div>
</body> </html>
```

Using the HTML5 Structural Elements

HTML5 has introduced the new structural elements such as <header>, <footer>, <nav>, <section>, etc. to define the different parts of a web page in a more semantic way.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>HTML5 Web Page Layout</title>
 <style>
body { font: 14px Arial,sans-serif; margin: 0px; }
header { padding: 10px 20px; background: #acb3b9; }
header h1 { font-size: 24px; }
.container { width: 100%; background: #f2f2f2; }
nav, section { float: left; padding: 20px; min-height: 170px; box-sizing: border-box; }
section { width: 80%; }
nav { width: 20%; background: #d4d7dc; }
nav ul { list-style: none; line-height: 24px; padding: 0px; }
nav ul li a { color: #333; }
.clearfix:after { content: "."; display: block; height: 0; clear: both; visibility: hidden; }
footer { background: #acb3b9; text-align: center; padding: 5px; }
</style>
</head>
<body>
<div class="container">
<header> <h1>Tutorial Republic</h1> </header>
<div class="wrapper clearfix">
<nav>
<ul>
<li>
```

```
<a href="#">Home</a>
</li>
<li>
<a href="#">About</a>
</li>
<li>
<a href="#">Contact</a>
</li>
</ul>
</nav>
<section> <h2>Welcome to our site</h2> <p>Here you will learn how to create websites...</p> </section>
</div>
<footer> <p>copyright &copy; tutorialrepublic.com</p> </footer>
</div> </body> </html>
```

"Programming is a skill best acquired by practice and example rather than from books."

By Sardar Azeem(+92-313-5879331)

Section 3

CSS 5.0 WITH EXAMPLES TUTORIAL

Lectured By : Sardar Azeem

What is CSS

CSS is the acronym for "Cascading Style Sheet". It's a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS helps the web developers to control the layout and other visual aspects of the web pages. CSS plays a crucial role in modern web development by providing the tools necessary to create visually appealing, accessible, and responsive websites.

CSS Versions

Since the inception of CSS, several versions have came into existence. Some of the notable versions include:

- CSS1 (Cascading Style Sheets Level1) - The initial version of CSS, released in December 1996. CSS1 provided basic styling capabilities for HTML documents, including properties for text, colors, backgrounds, margins, and borders.
- CSS2 (Cascading Style Sheets Level2) - Released in May 1998, CSS2 introduced new features such as positioning, z-index, media types, and more advanced selectors like attribute selectors and child selectors.
- CSS2.1 - The version 2.1, published as a W3C Recommendation in June 2011, clarified and refined CSS2, addressing inconsistencies and ambiguities in the specification. CSS2.1 focused on improving interoperability among web browsers.
- CSS3 (Cascading Style Sheets Level 3) - CSS3 is a collection of modules that extend the capabilities of CSS. It introduces numerous new features and enhancements, including advanced selectors, multiple column layouts, animations, transformations, gradients, shadows, and more.
- CSS4 (Cascading Style Sheets Level 4) - CSS4 is an ongoing effort to extend CSS3 with new features and enhancements.

Advantages of Using CSS

- Responsive design - CSS offers features like media queries that enable developers to create responsive layouts that adapt to different screen sizes and devices, ensuring a consistent user experience.
- Flexibility and Control - CSS provides precise control over the presentation of HTML elements, allowing developers to customize layout, typography, colors, and other visual properties.
- Consistency and Reusability - Developers can ensure consistency across the entire website, by defining styles in a central CSS file. Styles can be reused across multiple pages, reducing redundancy and making updates easier.
- Search Engine Optimization (SEO) - CSS can be used to structure content in a way that improves search engine visibility.

- Ease of Maintenance - Centralized CSS files make it easier to maintain and update styles across a website. Changes can be applied globally, ensuring uniformity and reducing the risk of inconsistencies.
- Faster Page Loading - External CSS files can be cached by browsers, resulting in faster page loading times for subsequent visits to a website. This caching mechanism reduces server load and bandwidth consumption.

Components of CSS

CSS works by associating rules with HTML elements. A CSS rule contains two main parts:

- a selector which specifies the HTML element(s) to style.
- a declaration block which contains one or more declarations separated by semicolons.

CSS Example

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: lightblue;
}

h1 {
  color: white;
  text-align: center;
}

p {
  font-family: verdana;
  font-size: 20px;
}
</style>
</head>
<body>
```

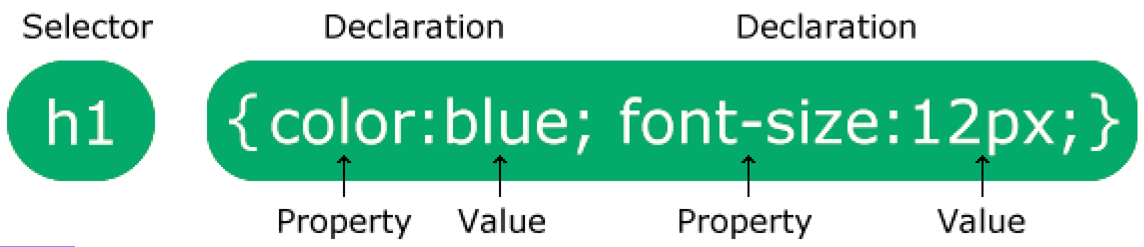
```
<h1>My First CSS Example</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

CSS Syntax

Following is the syntax of styling using CSS.

```
selector {
   property: value;
}
```

- Selector: CSS selectors are used to select the HTML element or groups of elements you want to style on a web page.
- Property: A CSS property is an aspect or characteristic of an HTML element that can be styled or modified using CSS, such as color, font-size, or margin.
- Value: Values are assigned to properties. For example, color property can have value like red, green etc.



Example

```
<!DOCTYPE html>
<html>
<head>
<style>
/* Style all the paragraphs */
p { background-color: black; color: white; padding: 5px; }
/* Style all elements with class 'special' */
.special { color: lightblue; /* Change text color */ }
</style>
</head>
```

```
<body>
<p> This a normal paragraph... </p>
<br>
<p class="special"> This is a paragraph with class special... </p>
<br>
<div class="special"> This is a div with class special... </div>
</body> </html>
```

CSS - Selectors

CSS Selectors are used to select the HTML elements you want to style on a web page. They allow you to target specific elements or groups of elements to apply styles like colors, fonts, margins, and more.

The element or elements that are selected by the selector are referred to as subject of the selector.

CSS Universal Selector

CSS universal selector is a special selector that selects all the elements in an HTML document. It is denoted by an asterisk mark (*).

Syntax

```
* {
  margin: 0;
  padding: 0;
}
```

Example

```
<html>
<head>
<style>
* { background-color: peachpuff; color: darkgreen; font-size: 25px; }
</style>
</head>
<body>
<h1>Universal selector (*)</h1>
<div>
Parent element
```

```
<p>Child paragraph 1</p>
<p>Child paragraph 2</p>
</div>
<p>Paragraph 3</p>
</body> </html>
```

CSS Element Selector

CSS element selector selects and styles specific HTML elements. The element selector is defined by simply using the element's name in the stylesheet.

Syntax

```
p {
  color: green;
}


h1 {
  text-decoration-line: underline;
}
```

Example

```
<html>
<head>
<style>
div { border: 5px inset gold; width: 300px; text-align: center; }
p { color: green; }
h1 { text-decoration-line: underline; }
</style>
</head>
<body>
<div>
<h1>Type selector</h1>
<p>div with border and text-aligned to center</p>
<p>paragraph with green color</p>
```

```
<p>h1 with an underline</p>
</div>
</body> </html>
```

CSS Class Selector

CSS class selector selects an element with a specific class attribute. The class selector is defined using a period (.) followed by the class name.

Syntax

```
.style-h1 {
   text-decoration-line: underline;
}

.style-p {
   color: green;
   font-size: 25px;
}
```

Example

```
<html>
<head>
<style>
.style-div { border: 5px inset gold; width: 300px; text-align: center; }
.style-p { color: green; font-size: 25px; }
.style-h1 { text-decoration-line: underline; }
</style>
</head>
<body>
<div class="style-div">
<h1 class="style-h1">class selector</h1>
<p class="style-p">class .style-p applied</p>
<p>No class applied on this p element</p>
</div>
</body> </html>
```

CSS ID Selector

CSS ID selector selects an element with a specific value for its id attribute. It is denoted by the "#" (hash) symbol.

Syntax

```
#style-p {
  color: green;
  font-size: 25px;
}

#style-h1 {
  text-decoration-line: underline;
  color: red;
}
```

Example

```
<html>
<head>
<style>
#style-div { border: 5px inset purple; width: 300px; text-align: center; background-color: lightgoldenrodyellow; }
#style-p { color: green; font-size: 25px; }
#style-h1 { text-decoration-line: underline; color: red; }
</style>
</head>
<body>
<div id="style-div">
<h1 id="style-h1">ID selector</h1>
<p id="style-p">id #style-p applied</p>
<p>No id applied on this p element</p>
</div>
</body> </html>
```

CSS Attribute Selector

CSS attribute selector selects an element based on a specific attribute or attribute values on an element.

Syntax

```
a[target] {
  background-color: peachpuff;
}
```

You can also specify the element with an attribute having a specific value.

```
a[href="https://www.pictacademy.com"] {
  background-color: peachpuff;
}
```

Example

```
<html>
<head>
<style>
a[target] { background-color: #04af2f; color: white; font-size: 2em; }
</style>
</head>
<body>
<h2>Attribute selector</h2>
<p>Styling applied to anchor element with target attribute:</p>
<a href="#">Tutorialspoint</a>
<a href="#" target="_blank">google</a>
<a href="#" target="_self">wikipedia</a>
 </body> </html>
```

CSS Group Selector

CSS group selector allow us to apply same style to multiple elements at a time. Name of elements are comma-separated. The group selector keep CSS concise and avoid redundancy.

Syntax

```
h1, h2 {
  background-color: grey;
}
```

Example

```
<html>
<head>
<style>
/* This applies to both <h1> and <h2> elements */
h1, h2 { background-color: grey; padding: 4px; }
/*Applies to all paragraphs, elements with class*/ /*'highlight', and element with ID 'hightlightSpan'*/ p,
.highlight, #hightlightSpan { background-color: yellow; padding: 10px; }
</style>
</head>
<body>
<h1>CSS Selectors</h1>
<h2>Group Selectors</h2>
<p>This is a paragraph.</p>
<div class="highlight"> This is div </div> <br>
<span id="hightlightSpan"> This is span </span>
</body> </html>
```

CSS Pseudo-class Selector

CSS pseudo-class selector styles a specific state of an element, such as :hover is used to style an element when hovered.

Syntax

```
a :hover {
  background-color: peachpuff;
  color: green;
  font-size: 2em;}
```

Example

```
<html>
<head>
<style>
  a:hover {
    background-color: peachpuff;
    color: green;
    font-size: 2em;
  }
</style>
</head>
<body>
  <h2>Pseudo-class selector</h2>
  <p>Styling applied to anchor element with a pseudo-class:</p>
  <a href="#">Tutorialspoint</a>
</body>
</html>
```

CSS Pseudo-element Selector

CSS pseudo-element selector is used to style a specific part of an element rather than the element itself.

Syntax

```
a::before {
  content: url();
}
```

Example

```
<html>
<head>
<style>
/* Add and style contents before paragraph */
p::before { content: "Note: "; font-weight: bold; color: red; }
 /* Add and style contents after paragraph */
```

```
p::after { content: " [Read more]"; font-style: italic; color: blue; }
</style>
</head>
<body>
 <h2>Pseudo-element selector</h2>
<p>This is a paragraph.</p>
</body>
 </html>
```

CSS Descendant Selector

CSS descendant selector styles all the tags which are child of a particular specified tag. To mention as descendant, a single space between parent and child element is used.

Syntax

```
div p {
   color: blue;
}
```

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
<style>
div{ border: 2px solid; }
div p { color: blue; }
</style>
</head>
<body>
<div>
<p> This paragraph is inside a div and will be blue. </p>
<section>
<p> This paragraph is inside a section which is inside a div and will also be blue. </p>
</section>
</div>
 <p> This paragraph is outside the div and will not be blue. </p> </body> </html>
```

CSS Child Selector

CSS child selector selects all the direct child of a particular element. This is denoted by '>' (Greater than) symbol.

Syntax

```
div > p {
   color: blue;
}
```

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
<style>
 div{ border: 2px solid; }
div > p { color: blue; }
</style>
</head>
<body>
<div>
<p> This paragraph is inside a div and will be blue. </p>
<section>
<p> This paragraph is inside a section which is inside a div and will not be blue as this is
not direct child </p>
</section>
</div>
 <p> This paragraph is outside the div and will not be blue. </p>
</body> </html>
```

CSS Adjacent Sibling Selectors

CSS adjacent sibling selector selects an element that is immediately preceded by a specified element. A plus symbol ("+") is used to denote adjacent sibling.

Syntax

```
h1 + p {
  margin-top: 0;
}
```

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
<style>
div{ border: 4px solid; }
div + p { color: blue; }
</style>
</head>
<body>
<p> This paragraph is above the div and will not be blue </p>
<div> <p> This paragraph is inside a div and will not be blue. </p> </div>
 <p> This paragraph 1 after the div and will be blue. </p>
<p>This paragraph 2 after the div and will not be blue. </p>
</body> </html>
```

CSS General Sibling Selector

general sibling selector targets all the element that is preceded by a specified element. The general sibling selector is denoted by tilde symbol ("~").

Syntax

```
h1 ~ p {
  color: gray;
}
```

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
<style>
div{ border: 4px solid; }
div ~ p { color: blue; }
</style>
</head>
<body>
<p> This paragraph is above the div and will not be blue </p>
<div> <p> This paragraph is inside a div and will not be blue. </p> </div>
<p> This paragraph 1 after the div and will be blue. </p>
<p>This paragraph 2 after the div and will be blue. </p>
</body> </html>
```

CSS Nested Selectors

CSS nesting allows to nest one style rule inside another rule, with the selector of the child rule relative to the selector of the parent rule.

Characteristics

The nesting selector shows the relationship between the parent and child rules.

- When the nested selectors are parsed by the browser, it automatically adds a whitespace between the selectors, thus creating a new CSS selector rule.
- In situations where the nested rule needs to be attached to the parent rule (without any whitespace), like while using the pseudo-class or compound selectors, the & nesting selector must be prepended immediately to achieve the desired result.
- In order to reverse the context of rules, the & nesting selector can be appended.
- There can be multiple instances of & nesting selector.

Syntax

```
nav {
  & ul {
    list-style: none;
    & li {
    display: inline-block;
    & a {
      text-decoration: none;
      color: blue;
      &:hover {
        color: red;
      }
    }
    }
  }
}
```

Example

```
<html>
<head>
<style>
#sample { font-family: Verdana, Geneva, Tahoma, sans-serif; font-size: 1.5rem;
& a { color: crimson;
&:hover, &:focus { color: green; background-color: yellow;
}
}
}
</style>
</head>
<body>
<h1>& nesting selector</h1>
```

```
<p id="sample"> Hover <a href="#">over the link</a>. </p>
 </body> </html>
```

How To Add CSS

There are three ways of inserting a style sheet:

- External CSS
- Internal CSS
- Inline CSS

External CSS

With an external style sheet, you can change the look of an entire website by changing just one file!

Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.

Main.html

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="mystyle.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Mystyle.css

```
body {
  background-color: lightblue;
}

h1 {
  color: navy;
  margin-left: 20px;
}
```

Internal CSS

An internal style sheet may be used if one single HTML page has a unique style.

The internal style is defined inside the <style> element, inside the head section.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: linen;
}
```

```
h1 {
  color: maroon;
  margin-left: 40px;
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Inline CSS

An inline style may be used to apply a unique style for a single element.

To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

Example

```
<!DOCTYPE html>
<html>
<body>

<h1 style="color:blue;text-align:center;">This is a heading</h1>
<p style="color:red;">This is a paragraph.</p>

</body>
</html>
```

CSS - Colors

CSS uses color values to specify a color. Typically, these are used to set a color either for the foreground of an element (i.e. its text) or else for the background of the element. They can also be used to affect the color of borders and other decorative effects.

You can specify your color values in various formats. Following table lists all the possible formats.

Format	Syntax	Description	Example
Keyword	<property>: <colorname>	CSS has a set of predefined color names that you can use directly.	red, blue, green, yellow, black, white, etc.
Hexadecimal Code	#RRGGBB	Starts with a hash (#) followed by six hexadecimal digits.	#FF0000 - red
Short Hexadecimal Code	#RGB	Shorter version of hexadecimal format where each of the RGB components is represented by a single digit, and the value is duplicated.	#F00 - red
RGB	rgb(red,green,blue)	Colors can be defined using the rgb() function, which takes three parameters representing the red, green, and blue values.	rgb(0, 0, 255) - blue
RGBA	rgba()	Similar to RGB, with an additional parameter for the alpha (transparency) value. 0 (fully transparent) and 1 (fully opaque)	rgba(0,0,255,0.5) - translucent blue
HSL	hsl()	Colors can be defined using the rgb() function which stands for Hue (0 to 360 degree), Saturation (%), and Lightness (%).	hsl(120, 100%, 50%) - pure green
HSLA	hsla()	Similar to HSL, with an additional parameter for the alpha (transparency) value.	hsl(120, 100%, 50%, 0.5) - translucent green
currentcolor Keyword	currentcolor	It refers to the value of the color property of the element.	color: red; /* Red text color */ border: 10px solid currentcolor; /* Red border color */
System color	as per OS or browser	CSS allows usage of system colors defined by the user's OS or browser.	ButtonText, Window, WindowText

CSS Color Names

In CSS, a color can be specified by using a predefined color name:

Example

```
<!DOCTYPE html>
<html>
<body>

<h1 style="background-color:Tomato;">Tomato</h1>
<h1 style="background-color:Orange;">Orange</h1>
<h1 style="background-color:DodgerBlue;">DodgerBlue</h1>
<h1 style="background-color:MediumSeaGreen;">MediumSeaGreen</h1>
<h1 style="background-color:Gray;">Gray</h1>
<h1 style="background-color:SlateBlue;">SlateBlue</h1>
<h1 style="background-color:Violet;">Violet</h1>
<h1 style="background-color:LightGray;">LightGray</h1>

</body>
</html>
```

CSS RGB Colors

In CSS, a color can be specified as an RGB value, using this formula:

rgb(*red, green*, *blue*)

Each parameter (red, green, and blue) defines the intensity of the color between 0 and 255.

For example, rgb(255, 0, 0) is displayed as red, because red is set to its highest value (255) and the others are set to 0.

To display black, set all color parameters to 0, like this: rgb(0, 0, 0).

To display white, set all color parameters to 255, like this: rgb(255, 255, 255).

Example

```
<!DOCTYPE html>
<html>
```

```
<body>

<h1>Specify colors using RGB values</h1>

<h2 style="background-color:rgb(255, 0, 0);">rgb(255, 0, 0)</h2>
<h2 style="background-color:rgb(0, 0, 255);">rgb(0, 0, 255)</h2>
<h2 style="background-color:rgb(60, 179, 113);">rgb(60, 179, 113)</h2>
<h2 style="background-color:rgb(238, 130, 238);">rgb(238, 130, 238)</h2>
<h2 style="background-color:rgb(255, 165, 0);">rgb(255, 165, 0)</h2>
<h2 style="background-color:rgb(106, 90, 205);">rgb(106, 90, 205)</h2>

</body>
</html>
```

RGBA Value

RGBA color values are an extension of RGB color values with an alpha channel - which specifies the opacity for a color.

An RGBA color value is specified with:

rgba(*red, green*, *blue, alpha*)

The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all):

Example

```
<!DOCTYPE html>
<html>
<body>

<h1>Make transparent colors with RGBA</h1>

<h2 style="background-color:rgba(255, 99, 71, 0);">rgba(255, 99, 71, 0)</h2>
<h2 style="background-color:rgba(255, 99, 71, 0.2);">rgba(255, 99, 71, 0.2)</h2>
<h2 style="background-color:rgba(255, 99, 71, 0.4);">rgba(255, 99, 71, 0.4)</h2>
<h2 style="background-color:rgba(255, 99, 71, 0.6);">rgba(255, 99, 71, 0.6)</h2>
<h2 style="background-color:rgba(255, 99, 71, 0.8);">rgba(255, 99, 71, 0.8)</h2>
```

```
<h2 style="background-color:rgba(255, 99, 71, 1);">rgba(255, 99, 71, 1)</h2>


</body>
</html>
```

HEX Value

In CSS, a color can be specified using a hexadecimal value in the form:

#rrggbb

Where rr (red), gg (green) and bb (blue) are hexadecimal values between 00 and ff (same as decimal 0-255).

For example, #ff0000 is displayed as red, because red is set to its highest value (ff) and the others are set to the lowest value (00).

To display black, set all values to 00, like this: #000000.

To display white, set all values to ff, like this: #ffffff.

Example

```
<!DOCTYPE html>
<html>
<body>

<h1>Specify colors using HEX values</h1>

<h2 style="background-color:#ff0000;">#ff0000</h2>
<h2 style="background-color:#0000ff;">#0000ff</h2>
<h2 style="background-color:#3cb371;">#3cb371</h2>
<h2 style="background-color:#ee82ee;">#ee82ee</h2>
<h2 style="background-color:#ffa500;">#ffa500</h2>
<h2 style="background-color:#6a5acd;">#6a5acd</h2>

</body>
</html>
```

HSL Value

In CSS, a color can be specified using hue, saturation, and lightness (HSL) in the form:

hsl(*hue*, *saturation*, *lightness*)

Hue is a degree on the color wheel from 0 to 360. 0 is red, 120 is green, and 240 is blue.

Saturation is a percentage value. 0% means a shade of gray, and 100% is the full color.

Lightness is also a percentage. 0% is black, 50% is neither light or dark, 100% is white

Example

```
<!DOCTYPE html>
<html>
<body>

<h1>Specify colors using HSL values</h1>

<h2 style="background-color:hsl(0, 100%, 50%);">hsl(0, 100%, 50%)</h2>
<h2 style="background-color:hsl(240, 100%, 50%);">hsl(240, 100%, 50%)</h2>
<h2 style="background-color:hsl(147, 50%, 47%);">hsl(147, 50%, 47%)</h2>
<h2 style="background-color:hsl(300, 76%, 72%);">hsl(300, 76%, 72%)</h2>
<h2 style="background-color:hsl(39, 100%, 50%);">hsl(39, 100%, 50%)</h2>
<h2 style="background-color:hsl(248, 53%, 58%);">hsl(248, 53%, 58%)</h2>

</body>
</html>
```

HSLA Value

HSLA color values are an extension of HSL color values with an alpha channel - which specifies the opacity for a color.

An HSLA color value is specified with:

hsla(*hue, saturation*, *lightness, alpha*)

The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all):

Example

```
<!DOCTYPE html>
<html>
<body>
<h1>Make transparent colors with HSLA</h1>
<h2 style="background-color:hsla(9, 100%, 64%, 0);">hsla(9, 100%, 64%, 0)</h2>
<h2 style="background-color:hsla(9, 100%, 64%, 0.2);">hsla(9, 100%, 64%, 0.2)</h2>
<h2 style="background-color:hsla(9, 100%, 64%, 0.4);">hsla(9, 100%, 64%, 0.4)</h2>
<h2 style="background-color:hsla(9, 100%, 64%, 0.6);">hsla(9, 100%, 64%, 0.6)</h2>
<h2 style="background-color:hsla(9, 100%, 64%, 0.8);">hsla(9, 100%, 64%, 0.8)</h2>
<h2 style="background-color:hsla(9, 100%, 64%, 1);">hsla(9, 100%, 64%, 1)</h2>
</body>
</html>
```

CSS Backgrounds

CSS backgrounds define the background colors and images of HTML elements. They allow you to use different colors, gradients, or images behind the content. In this chapter, we will learn about various CSS background properties, including how to set background colors, apply images, adjust their size and position, control repetition, and more.

CSS Backgrounds Examples.

CSS Background Shorthand Property

The background shorthand property allows you to specify all background properties in a single declaration.The correct order of properties when using the shorthand background property is as follows:

- background-color
- background-image
- background-position
- background-size (must be used with /)
- background-repeat
- background-origin
- background-attachment
- background-clip

Syntax

background: bg-color bg-image bg-position bg-size bg-repeat bg-origin bg-clip bg-attachment | initial | inherit;

/* Example */

background: green url('image.jpg') top/20% no-repeat border-box content-box fixed;

Note: If background-size is to be added, it must be included immediately after the background-position, separated with '/'. For example: "left/50%".

Setting Background Color

You can set the background color for elements like div, span, body, paragraph, etc using the background-color property.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
body { background-color: lightgray; }
div{ padding: 25px; }
.firstDiv{ background-color: rgb(255, 215, 0); }
.secondDiv{ background-color: #f0f0f0; }
.thirdDiv{ background-color: hsl(120, 100%, 75%); }
</style>
</head>
<body>
<h2>CSS Background Colors</h2>
 Body Color: lightgray; <br> <br>
 <div class="firstDiv"> Color: rgb(255, 215, 0) </div> <br>
<div class="secondDiv"> Color: #f0f0f0</div> <br>
<div class="thirdDiv"> Color: hsl(120, 100%, 75%)</div>
</body> </html>
```

Setting Images in Background

To set an image as background for another element such as div, span, body, paragraph, etc., you can use the background-image property. It can be used to set one or more than one image as the background. To set multiple images as background, we separate the images using commas.

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
<style>
div{
background-color: rgba(255, 255, 255);
opacity: 70%;
padding: 20px;
}
body {
background-image: url(/css/images/logo.png);
height: 350px;
 }
</style>
</head>
<body>
<div>
 <h1>Welcome to My Website</h1>
 <p> This is an example of setting a background image using CSS </p>
 </div>
 </body> </html>
```

Define Background Position

The background-position property sets the initial position of the element's background image. The position of the image is relative to the value set by the background-origin property.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
.position-right { background-image: url('/css/images/logo.png'); background-position: right; background-repeat: no-repeat; width: 100%; height: 300px; border: 3px solid black; position: relative; }
</style>
</head>
<body>
<div class="position-right"></div>
</body> </html>
```

Setting Background Size

To set the size of the background image of an element, you can use the background-size property. The background image can either be stretched, constrained, or left to its normal size.

Example

```
<!DOCTYPE html>
<html> <head>
<style>
.size-contain { background-image: url('/css/images/pink-flower.jpg');
background-size: contain; width: 300px; height: 300px; }
</style>
</head>
<body>
<h2>CSS background-size property</h2>
<div class="size-contain"></div>
</body>
</html>
```

Repeating Background Image

You can control the repetition of the background images using the background-repeat property. The image can be repeated along the horizontal and vertical axes, or not repeated.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
.repeat { background-image: url('/css/images/logo.png');
background-repeat: repeat; width: 800px; height: 400px; position: relative;
}
</style>
</head>
<body>
<h2> CSS background-repeat property </h2>
<div class="repeat"></div>
</body> </html>
```

Defining Background Origin

CSS background-origin property is used to set the origin of the background, which could be from the start of the border, inside the border, or inside the padding.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
div { border: 10px rgb(13, 7, 190); border-style: dashed; margin: 5px; padding: 1cm; font:
700 1em sans-serif; color: aliceblue; display: inline-block; background-image:
url('/css/images/yellow-flower.jpg'); height: 200px; width: 200px; background-size:
contain; } .content-box { background-origin: content-box; }
</style>
</head>
<body>
<div class="content-box"> </div>
<p> This image background start from content box of div element. </p>
</body>
```

```
</html>
```

Controlling Background Scrolling

You can use the background-attachment property to determine whether the position of the background image is fixed within the viewport or scrolls within its container.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
.fixed { background-image: url('images/logo.png'); background-repeat: no-repeat; background-attachment: fixed; background-position: left top; background-color: lightblue; background-size: 40% 30%; padding: 5rem; width: 250px; height: 500px; }
</style>
</head>
<body>
<h2>CSS background-attachment Property</h2>
<div class="fixed">
<p> Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularized in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.
</p>
</div>
</body>
</html>
```

Controlling Background Display

You can use CSS background-clip property to specify how the background image or color should be displayed within an element's padding box, border box, or content box. It determines the area of an element to which the background will be applied.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
p { border: 10px dotted black; padding: 15px; background: green; color: white; } .border-
area { background-clip: border-box; }
.padding-area { background-clip: padding-box; }
</style>
</head>
<body>
<h2>CSS background-clip property</h2>
<p class="border-area"> Background applied to the entire element. </p>
<p class="padding-area"> Background applied to the content & padding area. </p>
</body> </html>
```

CSS Borders

The CSS border properties allow you to specify the style, width, and color of an element's border.

CSS Border Style

The border-style property specifies what kind of border to display.

- dotted - Defines a dotted border
- dashed - Defines a dashed border
- solid - Defines a solid border
- double - Defines a double border
- groove - Defines a 3D grooved border. The effect depends on the border-color value
- ridge - Defines a 3D ridged border. The effect depends on the border-color value
- inset - Defines a 3D inset border. The effect depends on the border-color value
- outset - Defines a 3D outset border. The effect depends on the border-color value
- none - Defines no border
- hidden - Defines a hidden border

The border-style property can have from one to four values (for the top border, right border, bottom border, and the left border).

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
p.dotted {border-style: dotted;}
p.dashed {border-style: dashed;}
p.solid {border-style: solid;}
p.double {border-style: double;}
p.groove {border-style: groove;}
p.ridge {border-style: ridge;}
p.inset {border-style: inset;}
p.outset {border-style: outset;}
p.none {border-style: none;}
p.hidden {border-style: hidden;}
p.mix {border-style: dotted dashed solid double;}
</style>
</head>
<body>

<h2>The border-style Property</h2>
<p>This property specifies what kind of border to display:</p>

<p class="dotted">A dotted border.</p>
<p class="dashed">A dashed border.</p>
<p class="solid">A solid border.</p>
<p class="double">A double border.</p>
<p class="groove">A groove border.</p>
<p class="ridge">A ridge border.</p>
<p class="inset">An inset border.</p>
```

```
<p class="outset">An outset border.</p>
<p class="none">No border.</p>
<p class="hidden">A hidden border.</p>
<p class="mix">A mixed border.</p>

</body>
</html>
```

CSS Border Width

The border-width property specifies the width of the four borders.

The width can be set as a specific size (in px, pt, cm, em, etc) or by using one of the three pre-defined values: thin, medium, or thick:

```
<!DOCTYPE html>
<html>
<head>
<style>
p.one {
  border-style: solid;
  border-width: 5px;
}

p.two {
  border-style: solid;
  border-width: medium;
}

p.three {
  border-style: dotted;
  border-width: 2px;
}

p.four {
  border-style: dotted;
  border-width: thick;
}

p.five {
  border-style: double;
  border-width: 15px;
}

p.six {
  border-style: double;
  border-width: thick;
}
</style>
</head>
<body>
<p class="one">Some text.</p>
<p class="two">Some text.</p>
<p class="three">Some text.</p>
<p class="four">Some text.</p>
<p class="five">Some text.</p>
<p class="six">Some text.</p>
</body>
</html>
```

CSS Border Color

The border-color property is used to set the color of the four borders.

The color can be set by:

- name - specify a color name, like "red"
- HEX - specify a HEX value, like "#ff0000"
- RGB - specify a RGB value, like "rgb(255,0,0)"
- HSL - specify a HSL value, like "hsl(0, 100%, 50%)"
- Transparent

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
p.one {
  border-style: solid;
  border-color: red;
}

p.two {
  border-style: solid;
  border-color: green;
}

p.three {
  border-style: dotted;
  border-color: blue;
}
</style>
</head>
<body>
<p class="one">A solid red border</p>
<p class="two">A solid green border</p>
<p class="three">A dotted blue border</p>
</body>
</html>
```

CSS Border - Individual Sides

From the examples on the previous pages, you have seen that it is possible to specify a different border for each side.

In CSS, there are also properties for specifying each of the borders (top, right, bottom, and left):

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  border-top-style: dotted;
  border-right-style: solid;
  border-bottom-style: dotted;
  border-left-style: solid;
}
</style>
</head>
<body>
<h2>Individual Border Sides</h2>
<p>2 different border styles.</p>
</body>
</html>
```

CSS Border - Shorthand Property

To shorten the code, it is also possible to specify all the individual border properties in one property.

- border-width
- border-style (required)
- border-color

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  border: 5px solid red;
}
</style>
```

```
    </head>
    <body>
    <h2>The border Property</h2>
    <p>This property is a shorthand property for border-width, border-style, and
    border-color.</p>
    </body>
    </html>
```

CSS Rounded Borders

The border-radius property is used to add rounded borders to an element:

```
<!DOCTYPE html>
<html>
<head>
<style>
p.normal {
  border: 2px solid red;
  padding: 5px;
}

p.round1 {
  border: 2px solid red;
  border-radius: 5px;
  padding: 5px;
}

p.round2 {
  border: 2px solid red;
  border-radius: 8px;
  padding: 5px;
}

p.round3 {
  border: 2px solid red;
  border-radius: 12px;
  padding: 5px;
}
</style>
</head>
<body>
<p class="normal">Normal border</p>
<p class="round1">Round border</p>
<p class="round2">Rounder border</p>
<p class="round3">Roundest border</p>
</body>
</html>
```

CSS Margins

The CSS margin properties are used to create space around elements, outside of any defined borders.

With CSS, you have full control over the margins. There are properties for setting the margin for each side of an element (top, right, bottom, and left).

Margin - Individual Sides

CSS has properties for specifying the margin for each side of an element:

- margin-top
- margin-right
- margin-bottom
- margin-left

All the margin properties can have the following values:

- auto - the browser calculates the margin
- *length* - specifies a margin in px, pt, cm, etc.
- *%* - specifies a margin in % of the width of the containing element
- inherit - specifies that the margin should be inherited from the parent element

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  border: 1px solid black;
  margin-top: 100px;
  margin-bottom: 100px;
  margin-right: 150px;
  margin-left: 80px;
  background-color: lightblue;
}
</style>
</head>
<body>
<h2>Using individual margin properties</h2>
<div>This div element has a top margin of 100px, a right margin of 150px, a bottom margin
of 100px, and a left margin of 80px.</div>
</body>
</html>
```

CSS Padding

Padding is used to create space around an element's content, inside of any defined borders.

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  padding: 70px;
  border: 1px solid #4CAF50;
}
```

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  border: 1px solid black;
  background-color: lightblue;
  padding-top: 50px;
```

</style> </head> <body> <h2>CSS Padding</h2> <div>This element has a padding of 70px.</div> </body> </html>	padding-right: 30px; padding-bottom: 50px; padding-left: 80px; } </style> </head> <body> <h2>Using individual padding properties</h2> <div>This div element has a top padding of 50px, a right padding of 30px, a bottom padding of 50px, and a left padding of 80px.</div> </body> </html>

CSS Height, Width and Max-width

The height and width properties are used to set the height and width of an element.

The height and width properties do not include padding, borders, or margins. It sets the height/width of the area inside the padding, border, and margin of the element.

CSS height and width Values

The height and width properties may have the following values:

- auto - This is default. The browser calculates the height and width
- length - Defines the height/width in px, cm, etc.
- % - Defines the height/width in percent of the containing block
- initial - Sets the height/width to its default value
- inherit - The height/width will be inherited from its parent value

<!DOCTYPE html> <html> <head> <style> div { height: 50px; width: 100%;	<!DOCTYPE html> <html> <head> <style> div { height: 200px; width: 50%;

```
  border: 1px solid #4CAF50;
}
</style>
</head>
<body>

<h2>CSS height and width properties</h2>

<div>This div element has a height of 50 pixels and a width of 100%.</div>

</body>
</html>
```

```
  background-color: powderblue;
}
</style>
</head>
<body>

<h2>Set the height and width of an element</h2>

<div>This div element has a height of 200px and a width of 50%.</div>

</body>
</html>
```

Setting max-width

The max-width property is used to set the maximum width of an element.

The max-width can be specified in *length values*, like px, cm, etc., or in percent (%) of the containing block, or set to none (this is default. Means that there is no maximum width).

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  max-width: 500px;
  height: 100px;
  background-color: powderblue;
}
</style>
</head>
<body>

<h2>Set the max-width of an element</h2>
```

```
<div>This div element has a height of 100px and a max-width of 500px.</div>

<p>Resize the browser window to see the effect.</p>

</body>
</html>
```

CSS Outline

An outline is a line that is drawn around elements, OUTSIDE the borders, to make the element "stand out".

CSS has the following outline properties:

- outline-style
- outline-color
- outline-width
- outline-offset
- outline

CSS Outline Style

The outline-style property specifies the style of the outline, and can have one of the following values:

- dotted - Defines a dotted outline
- dashed - Defines a dashed outline
- solid - Defines a solid outline
- double - Defines a double outline
- groove - Defines a 3D grooved outline
- ridge - Defines a 3D ridged outline
- inset - Defines a 3D inset outline
- outset - Defines a 3D outset outline
- none - Defines no outline
- hidden - Defines a hidden outline

```html
<!DOCTYPE html>
<html>
<head>
<style>
p {
  border: 2px solid black;
  outline: #4CAF50 solid 10px;
  margin: auto;
  padding: 20px;
  text-align: center;
}
</style>
</head>
<body>

<h2>CSS Outline</h2>
<p>This element has a 2px black border and a green outline with a width of 10px.</p>

</body>
</html>
```

```html
<!DOCTYPE html>
<html>
<head>
<style>
p {outline-color:red;}

p.dotted {outline-style: dotted;}
p.dashed {outline-style: dashed;}
p.solid {outline-style: solid;}
p.double {outline-style: double;}
p.groove {outline-style: groove;}
p.ridge {outline-style: ridge;}
p.inset {outline-style: inset;}
p.outset {outline-style: outset;}
</style>
</head>
<body>

<h2>The outline-style Property</h2>

<p class="dotted">A dotted outline</p>
<p class="dashed">A dashed outline</p>
<p class="solid">A solid outline</p>
<p class="double">A double outline</p>
<p class="groove">A groove outline. The effect depends on the outline-color value.</p>
<p class="ridge">A ridge outline. The effect depends on the outline-color value.</p>
<p class="inset">An inset outline. The effect depends on the outline-color value.</p>
```

	<p class="outset">An outset outline. The effect depends on the outline-color value.</p> </body> </html>

CSS Outline Width

The outline-width property specifies the width of the outline, and can have one of the following values:

- thin (typically 1px)
- medium (typically 3px)
- thick (typically 5px)
- A specific size (in px, pt, cm, em, etc)

```
<!DOCTYPE html>
<html>
<head>
<style>
p.ex1 {
  border: 1px solid black;
  outline-style: solid;
  outline-color: red;
  outline-width: thin;
}

p.ex2 {
  border: 1px solid black;
  outline-style: solid;
  outline-color: red;
  outline-width: medium;
}

p.ex3 {
  border: 1px solid black;
  outline-style: solid;
  outline-color: red;
  outline-width: thick;
}

p.ex4 {
  border: 1px solid black;
  outline-style: solid;
  outline-color: red;
  outline-width: 4px;
}
</style>
</head>
<body>

<h2>The outline-width Property</h2>

<p class="ex1">A thin outline.</p>
<p class="ex2">A medium outline.</p>
<p class="ex3">A thick outline.</p>
<p class="ex4">A 4px thick outline.</p>

</body>
</html>
```

CSS Outline Color

The outline-color property is used to set the color of the outline.

The color can be set by:

- name - specify a color name, like "red"
- HEX - specify a hex value, like "#ff0000"
- RGB - specify a RGB value, like "rgb(255,0,0)"
- HSL - specify a HSL value, like "hsl(0, 100%, 50%)"

- invert - performs a color inversion (which ensures that the outline is visible, regardless of color background)

```
<!DOCTYPE html>
<html>
<head>
<style>
p.ex1 {
  border: 2px solid black;
  outline-style: solid;
  outline-color: red;
}

p.ex2 {
  border: 2px solid black;
  outline-style: dotted;
  outline-color: blue;
}

p.ex3 {
  border: 2px solid black;
  outline-style: outset;
  outline-color: grey;
}
</style>
</head>
<body>

<h2>The outline-color Property</h2>
<p>The outline-color property is used to set the color of the outline.</p>

<p class="ex1">A solid red outline.</p>
<p class="ex2">A dotted blue outline.</p>
```

```
<p class="ex3">An outset grey outline.</p>


</body>
</html>
```

CSS Outline - Shorthand property

The outline property is a shorthand property for setting the following individual outline properties:

- outline-width
- outline-style (required)
- outline-color

```
<!DOCTYPE html>
<html>
<head>
<style>
p.ex1 {outline: dashed;}
p.ex2 {outline: dotted red;}
p.ex3 {outline: 5px solid yellow;}
p.ex4 {outline: thick ridge pink;}
</style>
</head>
<body>

<h2>The outline Property</h2>

<p class="ex1">A dashed outline.</p>
<p class="ex2">A dotted red outline.</p>
<p class="ex3">A 5px solid yellow outline.</p>
<p class="ex4">A thick ridge pink outline.</p>

</body>
</html>
```

CSS Outline Offset

The outline-offset property adds space between an outline and the edge/border of an element. The space between an element and its outline is transparent.

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  margin: 30px;
  border: 1px solid black;
  outline: 1px solid red;
  outline-offset: 15px;
}
</style>
</head>
<body>

<h2>The outline-offset Property</h2>

<p>This paragraph has an outline 15px outside the border edge.</p>

</body>
</html>
```

CSS Text

CSS has a lot of properties for formatting text.

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  border: 1px solid gray;
  padding: 8px;
}

h1 {
  text-align: center;
  text-transform: uppercase;
  color: #4CAF50;
}

p {
  text-indent: 50px;
  text-align: justify;
  letter-spacing: 3px;
}

a {
  text-decoration: none;
  color: #008CBA;
}
</style>
</head>
<body>

<div>
  <h1>text formatting</h1>
  <p>This text is styled with some of the text formatting properties. The heading uses
the text-align, text-transform, and color properties.
  The paragraph is indented, aligned, and the space between characters is specified.
The underline is removed from this colored
  <a target="_blank" href="tryit.asp?filename=trycss_text">"Try it Yourself"</a> link.</p>
</div>
</body>
</html>
```

Text Color

The color property is used to set the color of the text. The color is specified by:

- a color name - like "red"
- a HEX value - like "#ff0000"
- an RGB value - like "rgb(255,0,0)"

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  color: blue;
}

h1 {
  color: green;
}
</style>
</head>
<body>

<h1>This is heading 1</h1>
<p>This is an ordinary paragraph. Notice that this text is blue. The default text color for a page is defined in the body selector.</p>
<p>Another paragraph.</p>

</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: lightgrey;
  color: blue;
}

h1 {
  background-color: black;
  color: white;
}

div {
  background-color: blue;
  color: white;
}
</style>
</head>
<body>
<h1>This is a Heading</h1>
<p>This page has a grey background color and a blue text.</p>
<div>This is a div.</div>
</body>
</html>
```

Text Alignment and Text Direction

In this chapter you will learn about the following properties:

- text-align
- text-align-last
- direction
- unicode-bidi
- vertical-align

Text Direction

The `direction` and `unicode-bidi` properties can be used to change the text direction of an element:

Vertical Alignment

The vertical-align property sets the vertical alignment of an element.

<!DOCTYPE html>	<!DOCTYPE html>
<html>	<html>
<head>	<head>
<style>	<style>
h1 {	img.a {
text-align: center;	vertical-align: baseline;
}	}
h2 {	img.b {
text-align: left;	vertical-align: text-top;
}	}
h3 {	img.c {
text-align: right;	vertical-align: text-bottom;
}	}
</style>	
</head>	img.d {
<body>	vertical-align: sub;

```

<h1>Heading 1 (center)</h1>
<h2>Heading 2 (left)</h2>
<h3>Heading 3 (right)</h3>

<p>The three headings above are aligned
center, left and right.</p>

</body>
</html>

<!DOCTYPE html>
<html>
<head>
<style>
p.a {
  text-align-last: right;
}

p.b {
  text-align-last: center;
}

p.c {
  text-align-last: justify;
}
</style>
</head>
<body>

<h1>The text-align-last Property</h1>
```

```
}

img.e {
  vertical-align: super;
}
</style>
</head>
<body>

<h1>The vertical-align Property</h1>

<h2>vertical-align: baseline
(default):</h2>
<p>An <img class="a" src="sqpurple.gif"
width="9" height="9"> image with a default
alignment.</p>

<h2>vertical-align: text-top:</h2>
<p>An <img class="b" src="sqpurple.gif"
width="9" height="9"> image with a text-
top alignment.</p>

<h2>vertical-align: text-bottom:</h2>
<p>An <img class="c" src="sqpurple.gif"
width="9" height="9"> image with a text-
bottom alignment.</p>

<h2>vertical-align: sub:</h2>
<p>An <img class="d" src="sqpurple.gif"
width="9" height="9"> image with a sub
alignment.</p>

<h2>vertical-align: sup:</h2>
```

```
<h2>text-align-last: right:</h2>

<p class="a">Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.</p>


<h2>text-align-last: center:</h2>

<p class="b">Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.</p>


<h2>text-align-last: justify:</h2>

<p class="c">Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.</p>


</body>

</html>
```

```
<p>An <img class="e" src="sqpurple.gif" width="9" height="9"> image with a super alignment.</p>


</body>

</html>
```

CSS Text Decoration

- text-decoration-line
- text-decoration-color
- text-decoration-style
- text-decoration-thickness
- text-decoration

Add a Decoration Line to Text

The text-decoration-line property is used to add a decoration line to text.

Specify a Color for the Decoration Line

The text-decoration-color property is used to set the color of the decoration line.

Specify a Style for the Decoration Line

The text-decoration-style property is used to set the style of the decoration line.

Specify the Thickness for the Decoration Line

The text-decoration-thickness property is used to set the thickness of the decoration line.

The Shorthand Property

The text-decoration property is a shorthand property for:

- text-decoration-line (required)
- text-decoration-color (optional)
- text-decoration-style (optional)
- text-decoration-thickness (optional)

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  text-decoration: overline;
}

h2 {
  text-decoration: line-through;
}

h3 {
  text-decoration: underline;
}

p.ex {
  text-decoration: overline underline;
}
</style>
</head>
<body>
```

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  text-decoration-line: overline;
  text-decoration-color: red;
}

h2 {
  text-decoration-line: line-through;
  text-decoration-color: blue;
}

h3 {
  text-decoration-line: underline;
  text-decoration-color: green;
}

p {
  text-decoration-line: overline underline;
  text-decoration-color: purple;
```

```
    <h1>Overline text decoration</h1>
    <h2>Line-through text decoration</h2>
    <h3>Underline text decoration</h3>
    <p class="ex">Overline and underline
    text decoration.</p>

    <p><strong>Note:</strong> It is not
    recommended to underline text that is
    not a link, as this often confuses
    the reader.</p>

    </body>
    </html>
```

```
}
</style>
</head>
<body>

<h1>Overline text decoration</h1>
<h2>Line-through text decoration</h2>
<h3>Underline text decoration</h3>
<p>Overline and underline text
decoration.</p>

</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  text-decoration-line: underline;
  text-decoration-style: solid; /* this is
default */
}

h2 {
  text-decoration-line: underline;
  text-decoration-style: double;
}

h3 {
  text-decoration-line: underline;
  text-decoration-style: dotted;
}
```

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  text-decoration-line: underline;
  text-decoration-thickness: auto;  /* this
is default */
}

h2 {
  text-decoration-line: underline;
  text-decoration-thickness: 5px;
}

h3 {
  text-decoration-line: underline;
  text-decoration-thickness: 25%;
}
```

```
p.ex1 {
  text-decoration-line: underline;
  text-decoration-style: dashed;
}

p.ex2 {
  text-decoration-line: underline;
  text-decoration-style: wavy;
}

p.ex3 {
  text-decoration-line: underline;
  text-decoration-color: red;
  text-decoration-style: wavy;
}
</style>
</head>
<body>

<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<p class="ex1">A paragraph.</p>
<p class="ex2">Another paragraph.</p>
<p class="ex3">Another paragraph.</p>

</body>
</html>
```

```
p {
  text-decoration-line: underline;
  text-decoration-color: red;
  text-decoration-style: double;
  text-decoration-thickness: 5px;
}
</style>
</head>
<body>
```

```
<!DOCTYPE html>
<html>
<head>
```

```
<style>
h1 {
  text-decoration: underline;
}

h2 {
  text-decoration: underline red;
}

h3 {
  text-decoration: underline red double;
}

p {
  text-decoration: underline red double 5px;
}
</style>
</head>
<body>

<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<p>A paragraph.</p>

</body>
</html>
```

Text Transformation

The text-transform property is used to specify uppercase and lowercase letters in a text.

```
<!DOCTYPE html>
<html>
<head>
```

```
<style>
p.uppercase {
  text-transform: uppercase;
}

p.lowercase {
  text-transform: lowercase;
}

p.capitalize {
  text-transform: capitalize;
}
</style>
</head>
<body>

<h1>Using the text-transform property</h1>

<p class="uppercase">This text is transformed to uppercase.</p>
<p class="lowercase">This text is transformed to lowercase.</p>
<p class="capitalize">This text is capitalized.</p>

</body>
</html>
```

CSS Text Spacing

Text Indentation

The text-indent property is used to specify the indentation of the first line of a text:

Letter Spacing

The letter-spacing property is used to specify the space between the characters in a text.

Line Height

The line-height property is used to specify the space between lines:

Word Spacing

The word-spacing property is used to specify the space between the words in a text.

White Space

The white-space property specifies how white-space inside an element is handled.

Text Shadow

The text-shadow property adds shadow to text.

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  text-indent: 50px;
}
</style>
</head>
<body>

<h1>Using text-indent</h1>

<p>In my younger and more vulnerable years my father gave me some advice that I've been turning over in my mind ever since. 'Whenever you feel like criticizing anyone,' he told me, 'just remember that all the people in this world haven't had the advantages that you've had.'</p>

</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<style>
h2 {
  letter-spacing: 5px;
}

h3 {
  letter-spacing: -2px;
}
</style>
</head>
<body>

<h1>Using letter-spacing</h1>

<h2>This is heading 1</h2>
<h3>This is heading 2</h3>

</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<style>
```

```
<!DOCTYPE html>
<html>
<head>
<style>
```

```
p.small {
  line-height: 0.7;
}

p.big {
  line-height: 1.8;
}
</style>
</head>
<body>

<h1>Using line-height</h1>

<p>
This is a paragraph with a standard line-height.<br>
The default line height in most browsers is about 110% to 120%.<br>
</p>

<p class="small">
This is a paragraph with a smaller line-height.<br>
This is a paragraph with a smaller line-height.<br>
</p>

<p class="big">
This is a paragraph with a bigger line-height.<br>
This is a paragraph with a bigger line-height.<br>
</p>
```

```
p.one {
  word-spacing: 10px;
}

p.two {
  word-spacing: -2px;
}
</style>
</head>
<body>
<h1>Using word-spacing</h1>
<p>This is a paragraph with normal word spacing.</p>
<p class="one">This is a paragraph with larger word spacing.</p>
<p class="two">This is a paragraph with smaller word spacing.</p>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  text-shadow: 2px 2px 5px red;
}
</style>
</head>
<body>

<h1>Text-shadow effect!</h1>
```

```
</body>
</html>
```

```
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  white-space: nowrap;
}
</style>
</head>
<body>
<h1>Using white-space</h1>
<p>
This is some text that will not wrap.
This is some text that will not wrap.
This is some text that will not wrap.
This is some text that will not wrap.
This is some text that will not wrap.
This is some text that will not wrap.
This is some text that will not wrap.
This is some text that will not wrap.
This is some text that will not wrap.
</p>

<p>Try to remove the white-space property to see the difference!</p>

</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  text-shadow: 2px 2px;
}
</style>
</head>
<body>
<h1>Text-shadow effect!</h1>
</body>
</html>

<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  text-shadow: 2px 2px red;
}
</style>
</head>
<body>

<h1>Text-shadow effect!</h1>

</body>
</html>
```

CSS Icons

How To Add Icons

The simplest way to add an icon to your HTML page, is with an icon library, such as Font Awesome.

Add the name of the specified icon class to any inline HTML element (like <i> or <span>).

All the icons in the icon libraries below, are scalable vectors that can be customized with CSS (size, color, shadow, etc.)

Font Awesome Icons

To use the Font Awesome icons, go to fontawesome.com, sign in, and get a code to add in the <head> section of your HTML page:

<script src="https://kit.fontawesome.com/*yourcode*.js" crossorigin="anonymous"></script>

Bootstrap Icons

To use the Bootstrap glyphicons, add the following line inside the <head> section of your HTML page:

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

Google Icons

To use the Google icons, add the following line inside the <head> section of your HTML page:

<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">

<!DOCTYPE html> <html> <head> <script src="https://kit.fontawesome.com/a076d05399.js" crossorigin="anonymous"></script> </head> <body> <i class="fas fa-cloud"></i> <i class="fas fa-heart"></i> <i class="fas fa-car"></i> <i class="fas fa-file"></i> <i class="fas fa-bars"></i>	<!DOCTYPE html> <html> <head> <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"> </head> <body> <i class="glyphicon glyphicon-cloud"></i> <i class="glyphicon glyphicon-remove"></i> <i class="glyphicon glyphicon-user"></i> <i class="glyphicon glyphicon-envelope"></i> <i class="glyphicon glyphicon-thumbs-up"></i>

</body> </html>	 </body> </html>
<!DOCTYPE html> <html> <head> <link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons" > </head> <body> <i class="material-icons">cloud</i> <i class="material-icons">favorite</i> <i class="material-icons">attachment</i> <i class="material-icons">computer</i> <i class="material-icons">traffic</i> </body> </html>	

CSS Links

Styling Links

Links can be styled with any CSS property (e.g. color, font-family, background, etc.).

The four links states are:

- a:link - a normal, unvisited link
- a:visited - a link the user has visited
- a:hover - a link when the user mouses over it
- a:active - a link the moment it is clicked

Text Decoration

The `text-decoration` property is mostly used to remove underlines from links:

Background Color

The background-color property can be used to specify a background color for links:

Link Buttons

This example demonstrates a more advanced example where we combine several CSS properties to display links as boxes/buttons:

```html
<!DOCTYPE html>
<html>
<head>
<style>
a {
  color: hotpink;
}
</style>
</head>
<body>

<h2>Style a link with a color</h2>

<p><b><a href="default.asp" target="_blank">This is a link</a></b></p>

</body>
</html>
```

```html
<!DOCTYPE html>
<html>
<head>
<style>
/* unvisited link */
a:link {
  color: red;
}

/* visited link */
a:visited {
  color: green;
}

/* mouse over link */
a:hover {
  color: hotpink;
}

/* selected link */
a:active {
  color: blue;
}
</style>
</head>
<body>

<h2>Styling a link depending on state</h2>

<p><b><a href="default.asp" target="_blank">This is a link</a></b></p>
```

	<p><b>Note:</b> a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective.</p> <p><b>Note:</b> a:active MUST come after a:hover in the CSS definition in order to be effective.</p> </body> </html>

<!DOCTYPE html>
<html>
<head>
<style>
a:link {
  text-decoration: none;
}
 	
a:visited {
  text-decoration: none;
}
 	
a:hover {
  text-decoration: underline;
}
 	
a:active {
  text-decoration: underline;
}
</style>
</head>
<body>
	<!DOCTYPE html>
	<html>
	<head>
	<style>
	a:link {
	  background-color: yellow;
	}
	
	a:visited {
	  background-color: cyan;
	}
	
	a:hover {
	  background-color: lightgreen;
	}
	
	a:active {
	  background-color: hotpink;
	}
	</style>
	</head>
	<body>

```
<h2>Styling a link with text-decoration property</h2>


<p><b><a href="default.asp" target="_blank">This is a link</a></b></p>
<p><b>Note:</b> a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective.</p>
<p><b>Note:</b> a:active MUST come after a:hover in the CSS definition in order to be effective.</p>


</body>
</html>
```

```
<h2>Styling a link with background-color property</h2>


<p><b><a href="default.asp" target="_blank">This is a link</a></b></p>
<p><b>Note:</b> a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective.</p>
<p><b>Note:</b> a:active MUST come after a:hover in the CSS definition in order to be effective.</p>


</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<style>
a:link, a:visited {
  background-color: #f44336;
  color: white;
  padding: 14px 25px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
}

a:hover, a:active {
  background-color: red;
}
</style>
</head>
<body>
```

```

<h2>Link Button</h2>


<p>A link styled as a button:</p>
<a href="default.asp" target="_blank">This is a link</a>


</body>
</html>
```

CSS Lists

HTML Lists and CSS List Properties

In HTML, there are two main types of lists:

- unordered lists (<ul>) - the list items are marked with bullets
- ordered lists (<ol>) - the list items are marked with numbers or letters

The CSS list properties allow you to:

- Set different list item markers for ordered lists
- Set different list item markers for unordered lists
- Set an image as the list item marker
- Add background colors to lists and list items

Different List Item Markers

The `list-style-type` property specifies the type of list item marker.

An Image as The List Item Marker

The list-style-image property specifies an image as the list item marker:

Position The List Item Markers

The list-style-position property specifies the position of the list-item markers (bullet points).

Remove Default Settings

The list-style-type:none property can also be used to remove the markers/bullets. Note that the list also has default margin and padding. To remove this, add margin:0 and padding:0 to <ul> or <ol>:

List - Shorthand property

The list-style property is a shorthand property. It is used to set all the list properties in one declaration:

Styling List With Colors

We can also style lists with colors, to make them look a little more interesting.

```
<!DOCTYPE html>
<html>
<head>
<style>
ul.a {
  list-style-type: circle;
}

ul.b {
  list-style-type: square;
}

ol.c {
  list-style-type: upper-roman;
}

ol.d {
  list-style-type: lower-alpha;
}
</style>
</head>
<body>

<h2>The list-style-type Property</h2>

<p>Example of unordered lists:</p>
<ul class="a">
  <li>Coffee</li>
```

```
<!DOCTYPE html>
<html>
<head>
<style>
ul {
  list-style-image: url('sqpurple.gif');
}
</style>
</head>
<body>

<h2>The list-style-image Property</h2>

<p>The list-style-image property specifies an image as the list item marker:</p>

<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>

</body>
</html>
```

```
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>

<ul class="b">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>

<p>Example of ordered lists:</p>
<ol class="c">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>

<ol class="d">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>

</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<style>
ul.a {
  list-style-position: outside;
}
```

```
<!DOCTYPE html>
<html>
<head>
<style>
ul.demo {
  list-style-type: none;
  margin: 0;
```

ul.b {
  list-style-position: inside;
}
</style>
</head>
<body>
 	
<h1>The list-style-position Property</h1>
 	
<h2>list-style-position: outside (default):</h2>
<ul class="a">
  <li>Coffee - A brewed drink prepared from roasted coffee beans, which are the seeds of berries from the Coffea plant</li>
  <li>Tea - An aromatic beverage commonly prepared by pouring hot or boiling water over cured leaves of the Camellia sinensis, an evergreen shrub (bush) native to Asia</li>
  <li>Coca Cola - A carbonated soft drink produced by The Coca-Cola Company. The drink's name refers to two of its original ingredients, which were kola nuts (a source of caffeine) and coca leaves</li>
</ul>
 	
<h2>list-style-position: inside:</h2>
<ul class="b">
  <li>Coffee - A brewed drink prepared from roasted coffee beans, which are the seeds of berries from the Coffea plant</li>
  <li>Tea - An aromatic beverage commonly prepared by pouring hot or boiling water over cured leaves of the
	padding: 0;
	}
	</style>
	</head>
	<body>
	
	<p>Default list:</p>
	<ul>
	  <li>Coffee</li>
	  <li>Tea</li>
	  <li>Coca Cola</li>
	</ul>
	
	<p>Remove bullets, margin and padding from list:</p>
	<ul class="demo">
	  <li>Coffee</li>
	  <li>Tea</li>
	  <li>Coca Cola</li>
	</ul>
	
	</body>
	</html>

```
Camellia sinensis, an evergreen shrub (bush) native to Asia</li>

  <li>Coca Cola - A carbonated soft drink produced by The Coca-Cola Company. The drink's name refers to two of its original ingredients, which were kola nuts (a source of caffeine) and coca leaves</li>

</ul>


</body>

</html>
```

```
<!DOCTYPE html>

<html>

<head>

<style>

ul {

  list-style: square inside url("sqpurple.gif");

}

</style>

</head>

<body>


<h2>The list-style Property</h2>


<p>The list-style property is a shorthand property, which is used to set all the list properties in one declaration.</p>


<ul>

  <li>Coffee</li>

  <li>Tea</li>

  <li>Coca Cola</li>
```

```
ol {
  background: #ff9999;
  padding: 20px;
}

ul {
  background: #3399ff;
  padding: 20px;
}

ol li {
  background: #ffe5e5;
  color: darkred;
  padding: 5px;
  margin-left: 35px;
}

ul li {
  background: #cce5ff;
  color: darkblue;
  margin: 5px;
}
```

```
</ul>


</body>
</html>
```

CSS Tables

The look of an HTML table can be greatly improved with CSS.

```
<!DOCTYPE html>
<html>
<head>
<style>
#customers {
  font-family: Arial, Helvetica, sans-serif;
  border-collapse: collapse;
  width: 100%;
}

#customers td, #customers th {
  border: 1px solid #ddd;
  padding: 8px;
}

#customers tr:nth-child(even){background-color: #f2f2f2;}

#customers tr:hover {background-color: #ddd;}

#customers th {
  padding-top: 12px;
  padding-bottom: 12px;
  text-align: left;
  background-color: #04AA6D;
  color: white;
}
</style>
</head>
<body>
<h1>A Fancy Table</h1>
<table id="customers">
  <tr>
    <th>Company</th>
    <th>Contact</th>
    <th>Country</th>
  </tr>
  <tr>
    <td>Alfreds Futterkiste</td>
    <td>Maria Anders</td>
    <td>Germany</td>
  </tr>
  <tr>
    <td>Berglunds snabbköp</td>
    <td>Christina Berglund</td>
    <td>Sweden</td>
  </tr>
  <tr>
    <td>Centro comercial Moctezuma</td>
    <td>Francisco Chang</td>
    <td>Mexico</td>
  </tr>
  <tr>
    <td>Ernst Handel</td>
    <td>Roland Mendel</td>
    <td>Austria</td>
  </tr>
  <tr>
```

```
    <td>Island Trading</td>
    <td>Helen Bennett</td>
    <td>UK</td>
  </tr>
  <tr>
    <td>Königlich Essen</td>
    <td>Philip Cramer</td>
    <td>Germany</td>
  </tr>
  <tr>
    <td>Laughing Bacchus Winecellars</td>
    <td>Yoshi Tannamuri</td>
    <td>Canada</td>
  </tr>
  <tr>
    <td>Magazzini Alimentari Riuniti</td>
    <td>Giovanni Rovelli</td>
    <td>Italy</td>
  </tr>
  <tr>
    <td>North/South</td>
    <td>Simon Crowther</td>
    <td>UK</td>
  </tr>
  <tr>
    <td>Paris spécialités</td>
    <td>Marie Bertrand</td>
    <td>France</td>
  </tr>
</table>

</body>
```

```
</html>
```

Table Borders

To specify table borders in CSS, use the border property.

Full-Width Table

The table above might seem small in some cases. If you need a table that should span the entire screen (full-width), add width: 100% to the <table> element:

Collapse Table Borders

The border-collapse property sets whether the table borders should be collapsed into a single border:

```
<!DOCTYPE html>
<html>
<head>
<style>
table, th, td {
  border: 1px solid;
}
</style>
</head>
<body>

<h2>Add a border to a table:</h2>

<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
  </tr>
  <tr>
    <td>Peter</td>
    <td>Griffin</td>
  </tr>
  <tr>
```

```
<!DOCTYPE html>
<html>
<head>
<style>
table, th, td {
  border: 1px solid;
}

table {
  width: 100%;
}
</style>
</head>
<body>

<h2>Full-width Table</h2>

<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
  </tr>
  <tr>
```

```html
    <td>Lois</td>
    <td>Griffin</td>
  </tr>
</table>

</body>
</html>
```

```html
    <td>Peter</td>
    <td>Griffin</td>
  </tr>
  <tr>
    <td>Lois</td>
    <td>Griffin</td>
  </tr>
</table>

</body>
</html>
```

```html
<!DOCTYPE html>
<html>
<head>
<style>
table, td, th {
  border: 1px solid;
}

table {
  width: 100%;
  border-collapse: collapse;
}
</style>
</head>
<body>

<h2>Let the table borders collapse</h2>

<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
  </tr>
  <tr>
    <td>Peter</td>
    <td>Griffin</td>
  </tr>
  <tr>
    <td>Lois</td>
    <td>Griffin</td>
  </tr>
</table>

</body>
</html>
```

Responsive Table

A responsive table will display a horizontal scroll bar if the screen is too small to display the full content:

```
<!DOCTYPE html>
<html>
<head>
<style>
table {
  border-collapse: collapse;
  width: 100%;
}

th, td {
  text-align: left;
  padding: 8px;
}

tr:nth-child(even) {background-color: #f2f2f2;}
```

```
</style>
</head>
<body>

<h2>Responsive Table</h2>
<p>A responsive table will display a horizontal scroll bar if the screen is too
small to display the full content. Resize the browser window to see the effect:</p>
<p>To create a responsive table, add a container element (like div) with
<strong>overflow-x:auto</strong> around the table element:</p>

<div style="overflow-x: auto;">
  <table>
    <tr>
      <th>First Name</th>
      <th>Last Name</th>
      <th>Points</th>
      <th>Points</th>
      <th>Points</th>
      <th>Points</th>
      <th>Points</th>
      <th>Points</th>
      <th>Points</th>
      <th>Points</th>
      <th>Points</th>
      <th>Points</th>
    </tr>
    <tr>
      <td>Jill</td>
      <td>Smith</td>
      <td>50</td>
      <td>50</td>
      <td>50</td>
      <td>50</td>
      <td>50</td>
      <td>50</td>
      <td>50</td>
      <td>50</td>
      <td>50</td>
      <td>50</td>
    </tr>
    <tr>
      <td>Eve</td>
      <td>Jackson</td>
      <td>94</td>
      <td>94</td>
      <td>94</td>
      <td>94</td>
      <td>94</td>
      <td>94</td>
      <td>94</td>
      <td>94</td>
      <td>94</td>
      <td>94</td>
    </tr>
    <tr>
      <td>Adam</td>
      <td>Johnson</td>
      <td>67</td>
      <td>67</td>
      <td>67</td>
      <td>67</td>
      <td>67</td>
      <td>67</td>
      <td>67</td>
      <td>67</td>
      <td>67</td>
      <td>67</td>
     </tr>
    </table>
  </div>

  </body>
  </html>
```

CSS Layout - The z-index Property

The z-index Property

When elements are positioned, they can overlap other elements.

The z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others).

An element can have a positive or negative stack order:

```
<!DOCTYPE html>
<html>
<head>
<style>
img {
  position: absolute;
  left: 0px;
  top: 0px;
  z-index: -1;
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<img src="img_tree.png">
<p>Because the image has a z-index of -1, it will be placed behind the text.</p>
```

```
</body>
</html>
```

CSS Layout - Overflow

The overflow property specifies whether to clip the content or to add scrollbars when the content of an element is too big to fit in the specified area.

The overflow property has the following values:

- visible - Default. The overflow is not clipped. The content renders outside the element's box
- hidden - The overflow is clipped, and the rest of the content will be invisible
- scroll - The overflow is clipped, and a scrollbar is added to see the rest of the content
- auto - Similar to scroll, but it adds scrollbars only when necessary

overflow: visible

By default, the overflow is `visible`, meaning that it is not clipped and it renders outside the element's box:

overflow: hidden

With the hidden value, the overflow is clipped, and the rest of the content is hidden:

overflow: scroll

Setting the value to scroll, the overflow is clipped and a scrollbar is added to scroll inside the box. Note that this will add a scrollbar both horizontally and vertically (even if you do not need it):

overflow: auto

The auto value is similar to scroll, but it adds scrollbars only when necessary:

overflow-x and overflow-y

The overflow-x and overflow-y properties specifies whether to change the overflow of content just horizontally or vertically (or both):

<!DOCTYPE html> <html> <head> <style> div {	<!DOCTYPE html> <html> <head> <style> div {

```
  background-color: coral;
  width: 200px;
  height: 65px;
  border: 1px solid;
  overflow: visible;
}
</style>
</head>
<body>

<h2>Overflow: visible</h2>

<p>By default, the overflow is visible, meaning that it is not clipped and it renders outside the element's box:</p>

<div>You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.</div>

</body>
</html>
```

```
  background-color: coral;
  width: 200px;
  height: 100px;
  border: 1px solid black;
  overflow: scroll;
}
</style>
</head>
<body>

<h2>Overflow: scroll</h2>

<p>Setting the overflow value to scroll, the overflow is clipped and a scrollbar is added to scroll inside the box. Note that this will add a scrollbar both horizontally and vertically (even if you do not need it):</p>

<div>You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.</div>

</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  background-color: coral;
  width: 200px;
```

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  background-color: coral;
  width: 200px;
  height: 65px;
  border: 1px solid black;
  overflow: auto;
}
</style>
</head>
<body>

<h2>Overflow: auto</h2>

<p>The auto value is similar to scroll, only it add scrollbars when necessary:</p>

<div>You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.</div>

</body>
</html>
```

```
  height: 65px;
  border: 1px solid black;
  overflow-x: hidden;
  overflow-y: scroll;
}
</style>
</head>
<body>

<h2>Overflow-x and overflow-y</h2>

<p>You can also change the overflow of content horizontally or vertically.</p>
<p>overflow-x specifies what to do with the left/right edges of the content.</p>
<p>overflow-y specifies what to do with the top/bottom edges of the content.</p>

<div>You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.</div>

</body>
</html>
```

CSS Layout - float and clear

The float property is used for positioning and formatting content e.g. let an image float left to the text in a container.

The float property can have one of the following values:

- left - The element floats to the left of its container
- right - The element floats to the right of its container
- none - The element does not float (will be displayed just where it occurs in the text). This is default

- inherit - The element inherits the float value of its parent

```
<!DOCTYPE html>
<html>
<head>
<style>
img {
  float: right;
}
</style>
</head>
<body>

<h2>Float Right</h2>

<p>In this example, the image will float to the right in the paragraph, and the text in the paragraph will wrap around the image.</p>

<p><img src="pineapple.jpg" alt="Pineapple" style="width:170px;height:170px;margin-left:15px;">
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed
```

```
<!DOCTYPE html>
<html>
<head>
<style>
img {
  float: left;
}
</style>
</head>
<body>

<h2>Float Left</h2>

<p>In this example, the image will float to the left in the paragraph, and the text in the paragraph will wrap around the image.</p>

<p><img src="pineapple.jpg" alt="Pineapple" style="width:170px;height:170px;margin-right:15px;">
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed ornare turpis.
```

ornare turpis. Donec vitae dui eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.</p> </body> </html>	Donec vitae dui eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.</p> </body> </html>

<!DOCTYPE html>
<html>
<head>
<style>
img {
  float: none;
}
</style>
</head>
<body>
 	
<h2>Float None</h2>
 	
<p>In this example, the image will be displayed just where it occurs in the text (float: none;).</p>
 	
<p><img src="pineapple.jpg" alt="Pineapple" style="width:170px;height:170px;">
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris
	<!DOCTYPE html>
	<html>
	<head>
	<style>
	div {
	  float: left;
	  padding: 15px;
	}
	
	.div1 {
	  background: red;
	}
	
	.div2 {
	  background: yellow;
	}
	
	.div3 {
	  background: green;
	}
	</style>
	</head>
	<body>

```
ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.</p>


</body>
</html>
```

```
<h2>Float Next To Each Other</h2>


<p>In this example, the three divs will float next to each other.</p>


<div class="div1">Div 1</div>
<div class="div2">Div 2</div>
<div class="div3">Div 3</div>


</body>
</html>
```

The clear Property

When we use the float property, and we want the next element below (not on right or left), we will have to use the clear property.

The clear property specifies what should happen with the element that is next to a floating element.

The clear property can have one of the following values:

- none - The element is not pushed below left or right floated elements. This is default
- left - The element is pushed below left floated elements
- right - The element is pushed below right floated elements
- both - The element is pushed below both left and right floated elements
- inherit - The element inherits the clear value from its parent

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  border: 3px solid #4CAF50;
  padding: 5px;
}
```

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  border: 3px solid #4CAF50;
  padding: 5px;
}
```

```
.img1 {
  float: right;
}

.img2 {
  float: right;
}

.clearfix {
  overflow: auto;
}
</style>
</head>
<body>

<h2>Without Clearfix</h2>

<p>This image is floated to the right. It is also taller than the element containing it, so it overflows outside of its container:</p>

<div>
  <img class="img1" src="pineapple.jpg" alt="Pineapple" width="170" height="170">
  Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet...
</div>

<h2 style="clear:right">With Clearfix</h2>
```

```
.img1 {
  float: right;
}

.img2 {
  float: right;
}

.clearfix::after {
  content: "";
  clear: both;
  display: table;
}
</style>
</head>
<body>

<h2>Without Clearfix</h2>

<p>This image is floated to the right. It is also taller than the element containing it, so it overflows outside of its container:</p>

<div>
  <img class="img1" src="pineapple.jpg" alt="Pineapple" width="170" height="170">
  Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet...
</div>

<h2 style="clear:right">With New Modern Clearfix</h2>
```

<p>We can fix this by adding a clearfix class with overflow: auto; to the containing element:</p> <div class="clearfix"> <img class="img2" src="pineapple.jpg" alt="Pineapple" width="170" height="170"> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet... </div> </body> </html>	<p>Add the clearfix hack to the containing element, to fix this problem:</p> <div class="clearfix"> <img class="img2" src="pineapple.jpg" alt="Pineapple" width="170" height="170"> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet... </div> </body> </html>

CSS Navigation Bar

Navigation Bar = List of Links

A navigation bar needs standard HTML as a base.

```
<!DOCTYPE html>
<html>
<body>
<ul>
  <li><a href="#home">Home</a></li>
  <li><a href="#news">News</a></li>
  <li><a href="#contact">Contact</a></li>
  <li><a href="#about">About</a></li>
</ul>
<p>Note: We use href="#" for test links. In a real web site this would be URLs.</p>
</body>
</html>
```

Vertical Navigation Bar

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  margin: 0;
}

ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  width: 25%;
  background-color: #f1f1f1;
  position: fixed;
  height: 100%;
  overflow: auto;
}

li a {
  display: block;
  color: #000;
  padding: 8px 16px;
  text-decoration: none;
}

li a.active {
  background-color: #04AA6D;
  color: white;
}
```

CSS Horizontal Navigation Bar

```
<!DOCTYPE html>
<html>
<head>
<style>
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
  background-color: #333;
}

li {
  float: left;
}

li a {
  display: block;
  color: white;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
}

li a:hover:not(.active) {
  background-color: #111;
}

.active {
  background-color: #04AA6D;
}
```

```
li a:hover:not(.active) {
  background-color: #555;
  color: white;
}
</style>
</head>
<body>

<ul>
  <li><a class="active" href="#home">Home</a></li>
  <li><a href="#news">News</a></li>
  <li><a href="#contact">Contact</a></li>
  <li><a href="#about">About</a></li>
</ul>

<div style="margin-left:25%;padding:1px 16px;height:1000px;">
  <h2>Fixed Full-height Side Nav</h2>
  <h3>Try to scroll this area, and see how the sidenav sticks to the page</h3>
  <p>Notice that this div element has a left margin of 25%. This is because the side navigation is set to 25% width. If you remove the margin, the sidenav will overlay/sit on top of this div.</p>
  <p>Also notice that we have set overflow:auto to sidenav. This will add a scrollbar when the sidenav is too long (for example if it has over 50 links inside of it).</p>
  <p>Some text..</p>
  <p>Some text..</p>
  <p>Some text..</p>
  <p>Some text..</p>
```

```
</style>
</head>
<body>

<ul>
  <li><a href="#home">Home</a></li>
  <li><a href="#news">News</a></li>
  <li><a href="#contact">Contact</a></li>
  <li style="float:right"><a class="active" href="#about">About</a></li>
</ul>

</body>
</html>
```

```
  <p>Some text..</p>
  <p>Some text..</p>
  <p>Some text..</p>
</div>

</body>
</html>
```

CSS Dropdowns

```
<!DOCTYPE html>
<html>
<head>
<style>
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
  background-color: #333;
}

li {
  float: left;
}

li a, .dropbtn {
  display: inline-block;
  color: white;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
}
```

```
li a:hover, .dropdown:hover .dropbtn {
  background-color: red;
}

li.dropdown {
  display: inline-block;
}

.dropdown-content {
  display: none;
  position: absolute;
  background-color: #f9f9f9;
  min-width: 160px;
  box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
  z-index: 1;
}

.dropdown-content a {
  color: black;
  padding: 12px 16px;
  text-decoration: none;
  display: block;
  text-align: left;
}

.dropdown-content a:hover {background-color: #f1f1f1;}

.dropdown:hover .dropdown-content {
  display: block;
}
</style>
```

```
</head>
<body>

<ul>
  <li><a href="#home">Home</a></li>
  <li><a href="#news">News</a></li>
  <li class="dropdown">
    <a href="javascript:void(0)" class="dropbtn">Dropdown</a>
    <div class="dropdown-content">
      <a href="#">Link 1</a>
      <a href="#">Link 2</a>
      <a href="#">Link 3</a>
    </div>
  </li>
</ul>

<h3>Dropdown Menu inside a Navigation Bar</h3>
<p>Hover over the "Dropdown" link to see the dropdown menu.</p>

</body>
</html>
```

CSS Image Gallery

```
<!DOCTYPE html>
<html>
<head>
<style>
div.gallery {
  margin: 5px;
  border: 1px solid #ccc;
  float: left;
  width: 180px;
}

div.gallery:hover {
  border: 1px solid #777;
}

div.gallery img {
  width: 100%;
  height: auto;
}

div.desc {
  padding: 15px;
  text-align: center;
}
</style>
</head>
<body>

<div class="gallery">
  <a target="_blank" href="img_5terre.jpg">
    <img src="img_5terre.jpg" alt="Cinque Terre" width="600" height="400">
  </a>
  <div class="desc">Add a description of the image here</div>
</div>

<div class="gallery">
  <a target="_blank" href="img_forest.jpg">
    <img src="img_forest.jpg" alt="Forest" width="600" height="400">
  </a>
  <div class="desc">Add a description of the image here</div>
</div>
```

```
<div class="gallery">
  <a target="_blank" href="img_lights.jpg">
    <img src="img_lights.jpg" alt="Northern Lights" width="600" height="400">
  </a>
  <div class="desc">Add a description of the image here</div>
</div>

<div class="gallery">
  <a target="_blank" href="img_mountains.jpg">
    <img src="img_mountains.jpg" alt="Mountains" width="600" height="400">
  </a>
  <div class="desc">Add a description of the image here</div>
</div>

</body>
</html>
```

CSS Forms

Resize the browser window to see the effect. When the screen is less than 600px wide, make the two columns stack on top of each other instead of next to each other.

```
<!DOCTYPE html>
<html>
<head>
<style>
* {
  box-sizing: border-box;
}

input[type=text], select, textarea {
  width: 100%;
  padding: 12px;
  border: 1px solid #ccc;
  border-radius: 4px;
  resize: vertical;
}

label {
  padding: 12px 12px 12px 0;
  display: inline-block;
}

input[type=submit] {
  background-color: #04AA6D;
  color: white;
  padding: 12px 20px;
  border: none;
  border-radius: 4px;
  cursor: pointer;
  float: right;
}

input[type=submit]:hover {
  background-color: #45a049;
}

.container {
  border-radius: 5px;
  background-color: #f2f2f2;
  padding: 20px;
}

.col-25 {
  float: left;
  width: 25%;
  margin-top: 6px;
}

.col-75 {
  float: left;
  width: 75%;
  margin-top: 6px;
}

/* Clear floats after the columns */
.row::after {
  content: "";
  display: table;
  clear: both;
}

/* Responsive layout - when the screen is less than 600px wide, make the two columns
stack on top of each other instead of next to each other */
@media screen and (max-width: 600px) {
  .col-25, .col-75, input[type=submit] {
    width: 100%;
    margin-top: 0;
  }
}
</style>
</head>
<body>

<h2>Responsive Form</h2>
<p>Resize the browser window to see the effect. When the screen is less than 600px
wide, make the two columns stack on top of each other instead of next to each
other.</p>
```

```
<div class="container">
 <form action="/action_page.php">
 <div class="row">
  <div class="col-25">
   <label for="fname">First Name</label>
  </div>
  <div class="col-75">
   <input type="text" id="fname" name="firstname" placeholder="Your name..">
  </div>
 </div>
 <div class="row">
  <div class="col-25">
   <label for="lname">Last Name</label>
  </div>
  <div class="col-75">
   <input type="text" id="lname" name="lastname" placeholder="Your last name..">
  </div>
 </div>
 <div class="row">
  <div class="col-25">
   <label for="country">Country</label>
  </div>
  <div class="col-75">
   <select id="country" name="country">
    <option value="australia">Australia</option>
    <option value="canada">Canada</option>
    <option value="usa">USA</option>
   </select>
  </div>
 </div>
 <div class="row">
  <div class="col-25">
```

```
      <label for="subject">Subject</label>
    </div>
    <div class="col-75">
      <textarea id="subject" name="subject" placeholder="Write something.." style="height:200px"></textarea>
    </div>
  </div>
  <br>
  <div class="row">
    <input type="submit" value="Submit">
  </div>
  </form>
</div>


</body>
</html>
```

Website Layout

A website is often divided into headers, menus, content and a footer:

Header

A header is usually located at the top of the website (or right below a top navigation menu). It often contains a logo or the website name:

Navigation Bar

A navigation bar contains a list of links to help visitors navigating through your website:

Content

The layout in this section, often depends on the target users. The most common layout is one (or combining them) of the following:

- 1-column (often used for mobile browsers)
- 2-column (often used for tablets and laptops)
- 3-column layout (only used for desktops)

Unequal Columns

The main content is the biggest and the most important part of your site.

It is common with **unequal** column widths, so that most of the space is reserved for the main content. The side content (if any) is often used as an alternative navigation or to specify information relevant to the main content. Change the widths as you like, only remember that it should add up to 100% in total:

Footer

The footer is placed at the bottom of your page. It often contains information like copyright and contact info:

Responsive Website Layout

By using some of the CSS code above, we have created a responsive website layout, which varies between two columns and full-width columns depending on screen width:

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>CSS Website Layout</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
body {
  margin: 0;
}

/* Style the header */
.header {
  background-color: #f1f1f1;
  padding: 20px;
  text-align: center;
}
</style>
</head>
<body>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>CSS Website Layout</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
* {
  box-sizing: border-box;
}

body {
  margin: 0;
}

/* Style the header */
.header {
  background-color: #f1f1f1;
  padding: 20px;
  text-align: center;
```

```
<div class="header">
  <h1>Header</h1>
</div>

</body>
</html>
```
	```
	}

	/* Style the top navigation bar */
	.topnav {
	  overflow: hidden;
	  background-color: #333;
	}

	/* Style the topnav links */
	.topnav a {
	  float: left;
	  display: block;
	  color: #f2f2f2;
	  text-align: center;
	  padding: 14px 16px;
	  text-decoration: none;
	}

	/* Change color on hover */
	.topnav a:hover {
	  background-color: #ddd;
	  color: black;
	}
	</style>
	</head>
	<body>

	<div class="header">
	  <h1>Header</h1>
	</div>

	<div class="topnav">
	```

	<a href="#">Link</a>
	<a href="#">Link</a>
	<a href="#">Link</a>
	</div>
	</body>
	</html>
<!DOCTYPE html>	<!DOCTYPE html>
<html lang="en">	<html lang="en">
<head>	<head>
<title>CSS Website Layout</title>	<title>CSS Website Layout</title>
<meta charset="utf-8">	<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">	<meta name="viewport" content="width=device-width, initial-scale=1">
<style>	<style>
* {	* {
box-sizing: border-box;	box-sizing: border-box;
}	}
body {	body {
margin: 0;	margin: 0;
}	}
/* Style the header */	/* Style the header */
.header {	.header {
background-color: #f1f1f1;	background-color: #f1f1f1;
padding: 20px;	padding: 20px;
text-align: center;	text-align: center;
}	}
/* Style the top navigation bar */	/* Style the top navigation bar */

.topnav {	.topnav {
overflow: hidden;	overflow: hidden;
background-color: #333;	background-color: #333;
}	}
/* Style the topnav links */	/* Style the topnav links */
.topnav a {	.topnav a {
float: left;	float: left;
display: block;	display: block;
color: #f2f2f2;	color: #f2f2f2;
text-align: center;	text-align: center;
padding: 14px 16px;	padding: 14px 16px;
text-decoration: none;	text-decoration: none;
}	}
/* Change color on hover */	/* Change color on hover */
.topnav a:hover {	.topnav a:hover {
background-color: #ddd;	background-color: #ddd;
color: black;	color: black;
}	}
/* Create three equal columns that floats next to each other */	/* Create three unequal columns that floats next to each other */
.column {	.column {
float: left;	float: left;
width: 33.33%;	padding: 10px;
padding: 15px;	}
}	
	/* Left and right column */
/* Clear floats after the columns */	.column.side {
.row::after {	width: 25%;
content: "";	}

```
  display: table;
  clear: both;
}

/* Responsive layout - makes the three columns stack on top of each other instead of next to each other */
@media screen and (max-width:600px) {
  .column {
    width: 100%;
  }
}
</style>
</head>
<body>

<div class="header">
  <h1>Header</h1>
  <p>Resize the browser window to see the responsive effect.</p>
</div>

<div class="topnav">
  <a href="#">Link</a>
  <a href="#">Link</a>
  <a href="#">Link</a>
</div>

<div class="row">
  <div class="column">
    <h2>Column</h2>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas sit
```

```

/* Middle column */
.column.middle {
  width: 50%;
}

/* Clear floats after the columns */
.row::after {
  content: "";
  display: table;
  clear: both;
}

/* Responsive layout - makes the three columns stack on top of each other instead of next to each other */
@media screen and (max-width: 600px) {
  .column.side, .column.middle {
    width: 100%;
  }
}
</style>
</head>
<body>

<div class="header">
  <h1>Header</h1>
  <p>Resize the browser window to see the responsive effect.</p>
</div>

<div class="topnav">
  <a href="#">Link</a>
```

```
amet pretium urna. Vivamus venenatis velit nec neque ultricies, eget elementum magna tristique. Quisque vehicula, risus eget aliquam placerat, purus leo tincidunt eros, eget luctus quam orci in velit. Praesent scelerisque tortor sed accumsan convallis.</p>
  </div>


  <div class="column">
    <h2>Column</h2>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas sit amet pretium urna. Vivamus venenatis velit nec neque ultricies, eget elementum magna tristique. Quisque vehicula, risus eget aliquam placerat, purus leo tincidunt eros, eget luctus quam orci in velit. Praesent scelerisque tortor sed accumsan convallis.</p>
  </div>


  <div class="column">
    <h2>Column</h2>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas sit amet pretium urna. Vivamus venenatis velit nec neque ultricies, eget elementum magna tristique. Quisque vehicula, risus eget aliquam placerat, purus leo tincidunt eros, eget luctus quam orci in velit. Praesent scelerisque tortor sed accumsan convallis.</p>
  </div>
</div>


</body>
```

```
  <a href="#">Link</a>
  <a href="#">Link</a>
</div>


<div class="row">
  <div class="column side">
    <h2>Side</h2>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit..</p>
  </div>


  <div class="column middle">
    <h2>Main Content</h2>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas sit amet pretium urna. Vivamus venenatis velit nec neque ultricies, eget elementum magna tristique. Quisque vehicula, risus eget aliquam placerat, purus leo tincidunt eros, eget luctus quam orci in velit. Praesent scelerisque tortor sed accumsan convallis.</p>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas sit amet pretium urna. Vivamus venenatis velit nec neque ultricies, eget elementum magna tristique. Quisque vehicula, risus eget aliquam placerat, purus leo tincidunt eros, eget luctus quam orci in velit. Praesent scelerisque tortor sed accumsan convallis.</p>
  </div>


  <div class="column side">
    <h2>Side</h2>
```

```
</html>
```

```
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit..</p>
  </div>
</div>

</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>CSS Website Layout</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
* {
  box-sizing: border-box;
}

body {
  margin: 0;
}

/* Style the header */
.header {
  background-color: #f1f1f1;
  padding: 20px;
  text-align: center;
}

/* Style the top navigation bar */
.topnav {
```

```
<!DOCTYPE html>
<html>
<head>
<style>
* {
  box-sizing: border-box;
}

body {
  font-family: Arial;
  padding: 10px;
  background: #f1f1f1;
}

/* Header/Blog Title */
.header {
  padding: 30px;
  text-align: center;
  background: white;
}

.header h1 {
  font-size: 50px;
}
```

```
  overflow: hidden;
  background-color: #333;
}

/* Style the topnav links */
.topnav a {
  float: left;
  display: block;
  color: #f2f2f2;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
}

/* Change color on hover */
.topnav a:hover {
  background-color: #ddd;
  color: black;
}

/* Create three unequal columns that
floats next to each other */
.column {
  float: left;
  padding: 10px;
}

/* Left and right column */
.column.side {
  width: 25%;
}
```

```
/* Style the top navigation bar */
.topnav {
  overflow: hidden;
  background-color: #333;
}

/* Style the topnav links */
.topnav a {
  float: left;
  display: block;
  color: #f2f2f2;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
}

/* Change color on hover */
.topnav a:hover {
  background-color: #ddd;
  color: black;
}

/* Create two unequal columns that floats
next to each other */
/* Left column */
.leftcolumn {
  float: left;
  width: 75%;
}

/* Right column */
.rightcolumn {
```

```
/* Middle column */
.column.middle {
  width: 50%;
}

/* Clear floats after the columns */
.row::after {
  content: "";
  display: table;
  clear: both;
}

/* Responsive layout - makes the three columns stack on top of each other instead of next to each other */
@media screen and (max-width: 600px) {
  .column.side, .column.middle {
    width: 100%;
  }
}

/* Style the footer */
.footer {
  background-color: #f1f1f1;
  padding: 10px;
  text-align: center;
}
</style>
</head>
<body>

<div class="header">
```

```
  float: left;
  width: 25%;
  background-color: #f1f1f1;
  padding-left: 20px;
}

/* Fake image */
.fakeimg {
  background-color: #aaa;
  width: 100%;
  padding: 20px;
}

/* Add a card effect for articles */
.card {
  background-color: white;
  padding: 20px;
  margin-top: 20px;
}

/* Clear floats after the columns */
.row::after {
  content: "";
  display: table;
  clear: both;
}

/* Footer */
.footer {
  padding: 20px;
  text-align: center;
  background: #ddd;
  <h1>Header</h1>
  <p>Resize the browser window to see the responsive effect.</p>
</div>

<div class="topnav">
  <a href="#">Link</a>
  <a href="#">Link</a>
  <a href="#">Link</a>
</div>

<div class="row">
  <div class="column side">
    <h2>Side</h2>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit..</p>
  </div>

  <div class="column middle">
    <h2>Main Content</h2>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas sit amet pretium urna. Vivamus venenatis velit nec neque ultricies, eget elementum magna tristique. Quisque vehicula, risus eget aliquam placerat, purus leo tincidunt eros, eget luctus quam orci in velit. Praesent scelerisque tortor sed accumsan convallis.</p>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas sit amet pretium urna. Vivamus venenatis velit nec neque ultricies, eget elementum magna tristique. Quisque vehicula, risus eget aliquam placerat, purus leo tincidunt eros, eget luctus quam orci in velit.
```

```
  margin-top: 20px;
}

/* Responsive layout - when the screen is less than 800px wide, make the two columns stack on top of each other instead of next to each other */
@media screen and (max-width: 800px) {
  .leftcolumn, .rightcolumn {
    width: 100%;
    padding: 0;
  }
}

/* Responsive layout - when the screen is less than 400px wide, make the navigation links stack on top of each other instead of next to each other */
@media screen and (max-width: 400px) {
  .topnav a {
    float: none;
    width: 100%;
  }
}
</style>
</head>
<body>

<div class="header">
  <h1>My Website</h1>
  <p>Resize the browser window to see the effect.</p>
</div>
```

```
Praesent scelerisque tortor sed accumsan convallis.</p>
  </div>

  <div class="column side">
    <h2>Side</h2>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit..</p>
  </div>
</div>

<div class="footer">
  <p>Footer</p>
</div>

</body>
</html>
```

```
<div class="topnav">
  <a href="#">Link</a>
  <a href="#">Link</a>
  <a href="#">Link</a>
  <a href="#" style="float:right">Link</a>
</div>

<div class="row">
  <div class="leftcolumn">
    <div class="card">
      <h2>TITLE HEADING</h2>
      <h5>Title description, Dec 7, 2017</h5>
      <div class="fakeimg" style="height:200px;">Image</div>
      <p>Some text..</p>
      <p>Sunt in culpa qui officia deserunt mollit anim id est laborum consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco.</p>
    </div>
    <div class="card">
      <h2>TITLE HEADING</h2>
      <h5>Title description, Sep 2, 2017</h5>
      <div class="fakeimg" style="height:200px;">Image</div>
      <p>Some text..</p>
      <p>Sunt in culpa qui officia deserunt mollit anim id est laborum consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco.</p>
    </div>
  </div>
  <div class="rightcolumn">
    <div class="card">
      <h2>About Me</h2>
      <div class="fakeimg"
style="height:100px;">Image</div>
      <p>Some text about me in culpa qui
officia deserunt mollit anim..</p>
    </div>
    <div class="card">
      <h3>Popular Post</h3>
      <div
class="fakeimg"><p>Image</p></div>
      <div
class="fakeimg"><p>Image</p></div>
      <div
class="fakeimg"><p>Image</p></div>
    </div>
    <div class="card">
      <h3>Follow Me</h3>
      <p>Some text..</p>
    </div>
  </div>
</div>

<div class="footer">
  <h2>Footer</h2>
</div>

</body>
</html>
```

CSS Gradients

CSS defines three types of gradients:

- Linear Gradients (goes down/up/left/right/diagonally)
- Radial Gradients (defined by their center)
- Conic Gradients (rotated around a center point)

CSS Linear Gradients

To create a linear gradient you must define at least two color stops. Color stops are the colors you want to render smooth transitions among. You can also set a starting point and a direction (or an angle) along with the gradient effect.

Syntax

background-image: linear-gradient(*direction*, *color-stop1*, *color-stop2, ...*);

CSS Radial Gradients

A radial gradient is defined by its center.

To create a radial gradient you must also define at least two color stops.

Syntax

background-image: radial-gradient(*shape size* at *position, start-color, ..., last-color*);

CSS Conic Gradients

A conic gradient is a gradient with color transitions rotated around a center point.

To create a conic gradient you must define at least two colors.

Syntax

background-image: conic-gradient([from *angle*] [at *position*,] *color* [*degree*], *color* [*degree*], *...*);

Section 4

JAVASCRIPT LANGUAGE WITH EXAMPLES

Lectured By: Sardar Azeem

What is JavaScript?

JavaScript (JS) is a programming language that lets developers create interactive web pages. JavaScript is a lightweight, object-oriented programming language that is used by several websites for scripting webpages.

JavaScript is an interpreted language that executes code line by line, providing more flexibility. It is a single-threaded language that executes one task at a time. JavaScript builds upon other programming languages, updating them based on user interactions and other input.

Why learn JavaScript?

- Easy to Learn: JavaScript is beginner-friendly and easier to learn than many other programming languages.
- Versatility: JavaScript can be used to develop websites, and you can use it on both the server side and client side using Angular expressjs and Node.js.
- Client Side: JavaScript is the main language for client-side logic and is supported by almost all browsers. It also offers several frameworks and libraries such as ReactJS, AngularJS, and js.
- Server Side: It is widely used for building server-side applications because it offers runtime environments like Node.js and frameworks like Express.js.
- Build Games: You can use JavaScript to create games, which can be 2D or 3D.

JavaScript Applications

As mentioned before, JavaScript is one of the most widely used programming languages (Front-end as well as Back-end). It has its presence in almost every area of software development. I'm going to list few of them here:

- Client side validation – This is really important to verify any user input before submitting it to the server and JavaScript plays an important role in validating those inputs at front-end itself.
- Manipulating HTML Pages – JavaScript helps in manipulating HTML page on the fly. This helps in adding and deleting any HTML tag very easily using JavaScript and modify your HTML to change its look and feel based on different devices and requirements.
- User Notifications – You can use JavaScript to raise dynamic pop-ups on the webpages to give different types of notifications to your website visitors.
- Back-end Data Loading – JavaScript provides Ajax library which helps in loading back-end data while you are doing some other processing. This really gives an amazing experience to your website visitors.

- Presentations – JavaScript also provides the facility of creating presentations which gives website look and feel. JavaScript provides RevealJS and BespokeJS libraries to build a web-based slide presentation.
- Server Applications – Node JS is built on Chrome's JavaScript runtime for building fast and scalable network applications. This is an event based library which helps in developing very sophisticated server applications including Web Servers.
- Machine learning – Developers can use the ML5.js library to complete the task related to machine learning.
- Game Developments – JavaScript contains multiple libraries and NPM packages to design graphics for the game. We can use HTML, CSS, and JavaScript with libraries to develop the games.
- Mobile applications – We can use frameworks like React Native to build feature-rich mobile applications.
- Internet of Things (IoT) – JavaScript is used to add functionality to devices like smartwatches, earbuds, etc.
- Data visualization – JavaScript contains the libraries like D3.js to visualize the data efficiently. The D3.js is also used to prepare high-quality charts to visualize the data.
- Cloud computing – We can use JavaScript in serverless computing platforms like Cloudflare and AWS lambda to write and deploy functions on the cloud.

Frameworks and Libraries of JavaScript

ReactJS

It is a popular JavaScript library that Meta and the community developed. The original author of ReactJS was Jordan Walke. It is a free and open-source front-end JavaScript library, which means anybody can utilize this framework without paying a single penny.

JQuery

JQuery is one of the oldest frameworks of JavaScript. It was originally authorized by John Resig. It is a free and open-source library that is used to create simple web applications. It helps simplify HTML DOM elements, Ajax, CSS animations, and event handling.

Vue.js

Vue.js is an open-source framework that is utilized to make user interfaces and single-page applications. It was originally authorized by Evan You and was initially released in February 2014.

AngularJS

AngularJS is a robust, free, and open-source JavaScript-based framework that Google developed. AngularJS is used to simplify development and testing applications, and it is utilized as the front-end of the MEAN stack.

Express.js

Express.js is a fast and open-source backend web application framework that TJ Holowaychuk, StrongLoop, and others developed. The main use of Express.js is to create complex web applications and APIs. It has become the prime choice when using the Mean stack.

Ember.js

Ember.js is an open-source framework that was initially released on December 8, 2011. It is used to create scalable SPAs. It is used on various websites such as Nordstrom, HashiCrop, Apple Music, Live Nation, Twitch, Intercom, Ghost, Square etc. With the use of this framework, developers can create desktop and mobile applications.

JavaScript Jobs

In the world, there are 15+ Million active JavaScript developers. Still, there is a shortage of skilled JavaScript developers. So, it could be a great chance for you to start your career as a JavaScript developer.

Here are the most popular companies offering the role of JavaScript developer. You can land as an employee with a high package after pursuing your career as a JavaScript developer.

- Amazon
- Google
- Microsoft
- Apple
- Adobe
- Facebook
- PayPal
- Many more...

Careers Opportunities in JavaScript

There are several career paths that you can choose after learning JavaScript. Here, we have listed some of them.

- Front-end developer
- Back-end developer
- Full-stack developer

- Web developer
- Game developer
- Mobile App developer
- DevOps Engineer
- Many other roles

JavaScript Syntax

JavaScript syntax comprises a set of rules that define how to construct a JavaScript code. JavaScript can be implemented using JavaScript statements that are placed within the <script>... </script> HTML tags in a web page.

You can place the <script> tags, containing your JavaScript, anywhere within your web page, but it is normally recommended that you should keep it within the <head> tags.

The <script> tag alerts the browser program to start interpreting all the text between these tags as a script. A simple syntax of your JavaScript will appear as follows.

```
<script ...>
JavaScript code
</script>
```

The script tag takes two important attributes

- Language —This attribute specifies what scripting language you are using. Typically, its value will be javascript. Although recent versions of HTML (and XHTML, its successor) have phased out the use of this attribute.
- Type —This attribute is what is now recommended to indicate the scripting language in use and its value should be set to "text/javascript". JavaScript has become default lannguage in HTML5, and modern browsers, so now adding *type* is not required.

So your JavaScript segment will look like —

```
<script language = "javascript" type = "text/javascript"> JavaScript code </script>
```

JavaScript Where To

JavaScript in <head>

In this example, a JavaScript function is placed in the <head> section of an HTML page.

The function is invoked (called) when a button is clicked:

Example

```html
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>
</head>
<body>
<h2>Demo JavaScript in Head</h2>
<p id="demo">A Paragraph</p>
<button type="button" onclick="myFunction()">Try it</button>
</body>
</html>
```

JavaScript in <body>

In this example, a JavaScript function is placed in the <body> section of an HTML page.

The function is invoked (called) when a button is clicked:

Example

```html
<!DOCTYPE html>
<html>
<body>

<h2>Demo JavaScript in Body</h2>

<p id="demo">A Paragraph</p>

<button type="button" onclick="myFunction()">Try it</button>

<script>
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>

</body>
</html>
```

External JavaScript

Scripts can also be placed in external files:

External file: myScript.js

```
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
```

External scripts are practical when the same code is used in many different web pages.

JavaScript files have the file extension .js.

To use an external script, put the name of the script file in the src (source) attribute of a <script> tag:

```
 <!DOCTYPE html>
<html>
<body>
<h2>Demo External JavaScript</h2>
<p id="demo">A Paragraph.</p>
<button type="button" onclick="myFunction()">Try it</button>
<p>This example links to "myScript.js".</p>
<p>(myFunction is stored in "myScript.js")</p>
<script src="myScript.js"></script>
</body>
</html>
```

JavaScript Output Statements

JavaScript can "display" data in different ways:

- Writing into an HTML element, using innerHTML or innerText.
- Writing into the HTML output using document.write().
- Writing into an alert box, using window.alert().
- Writing into the browser console, using console.log().

Using innerHTML

To access an HTML element, you can use the document.getElementById(id) method.

Use the id attribute to identify the HTML element.

Then use the innerHTML property to change the HTML content of the HTML element:

Example

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
<p>My First Paragraph</p>
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML = "<h2>Hello World</h2>";
</script>
</body>
</html>
```

Using document.write()

For testing purposes, it is convenient to use document.write():

Example

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
<p>My first paragraph.</p>
<script>
document.write(5 + 6);
</script>
</body>
</html>
```

Using window.alert()

You can use an alert box to display data:

Example

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
<p>My first paragraph.</p>
<script>
window.alert(5 + 6);
</script>
</body></html>
```

Using console.log()

For debugging purposes, you can call the console.log() method in the browser to display data.

Example

```
<!DOCTYPE html>
<html>
<body>
<script>
console.log(5 + 6);
</script>
</body>
</html>
```

JavaScript Operators

In JavaScript, an operator is a special symbol or keyword that operates on one or more operands to produce a result.

Types of JavaScript Operators

There are various operators that JavaScript supports. Such as:

1. Arithmetic Operators
2. Assignment Operators
3. Comparison Operators
4. Logical Operators
5. Bitwise Operators
6. Ternary Operators
7. Delete Operators
8. String Operators
9. Typeof Operators
10. Instanceof Operators
11. Chaining Operators
12. Comma Operators

1. Arithmetic Operators

In JavaScript, arithmetic operators are used to perform mathematical calculations like addition, subtraction, multiplication, etc.

Operator	Description	Example
+ (Addition)	It adds two operands.	10 + 2 // 12
- (Subtraction)	Subtracts the second number from the first.	10 - 2 // 8
* (Multiplication)	It multiplies both operands.	5 * 2 // 10
/ (Division)	It divides two operands.	10 / 2 // 5
% (Modulus)	It gives the remainder of an integer division.	5 % 2 // 1
++ (Increment)	Increases an integer value by one.	3++ or ++3 //4
-- (Decrement)	Decreases an integer value by one.	3-- or --3 //2

Example

```
const add = 3 + 3; //Addition
const sub = 4 - 1; //Subtraction
const a = 3 * 8; // Multiplication
const b = 4/2; //Division
console.log(add, sub, a, b);
```

2. Assignment Operators

Assignment operators in JavaScript are used to assign values to the variables. Assignment operators can perform operations like addition or multiplication before assigning the value.

Operator	Description	Example
=	It assigns the value of the right operand to the left operand.	a = b+ c
+= (Add and)	It adds the right operand to the left operand and assigns the result to the left operand.	a += b // a = a + b
-= (Subtract and)	This operator subtracts the right operand from	a -= b // a = a - b

	the left operand and assigns the result to the left operand.	
*= (Multiply and)	This operator multiplies the right operand by the left operand and assigns the result to the left operand.	a *= b // a = a * b
/= (Divide and)	This operator divides the left operand by the right operand and assigns the result to the left operand.	a /= b // a = a / b
% = (Modulus and)	This operator takes modulus using two operands and assigns the result to the left operand.	a %= b // a= a% b

Example

```
let a = 20;
a += 2;
a *= 3;
console.log(a);
```

3. Comparison Operators

Comparison Operators are used to compare two values and return a Boolean (true or false). Comparison operators are useful for making decisions in conditional Statements.

Operator	Description	Example
== (Equal to)	This operator checks if the value of two operands is equal or not.	2 == 2 gives us true.
!= (Not equal to)	This operator is used to check the inequality of two operands.	2 != 3 gives us true.
> (Greater than)	It is used to check if the value of the left side is greater than the right side.	3 > 4 gives us false.

< (Less than)	It is used to check if the value of the right side is greater than the left side.	4 < 5 gives us true.
>= (Greater than or equal to)	This operator checks if the value of the left operand is greater than or equal to the value of the right operand.	3 >= 3 gives us true.
<= (Less than or equal to)	This operator checks if the value of the right operand is greater than or equal to the value of the left operand.	3 <= 2 gives us false.
=== (Strictly equal to)	This operator checks whether the value and data type of the variable are equal or not.	2 === "2" gives us false.
!== (Strictly not equal to)	This operator is used to compare the inequality of two operands and types.	2 === "2" gives us true.

Example

```
let a = 3;
let b = 5;
let result = a < b;
console.log(result);
```

4. Logical Operators

In JavaScript, logical operators are used to perform the logical operations that determine the equality or difference between the values.

Operator	Description	Example
&& (Logical AND)	This operator checks if both the operands are non-zero; then the condition becomes true.	(expression1 && expression2)
|| (Logical OR)	This operator checks if any of the operands are non-zero; then the condition becomes true.	(expression1 || expression2)

! (Logical NOT)	This operator reverses the logical state of its operand. If a condition is true, then the Logical NOT operator will make it false.	!expression

Example

```
const p = true, q = false;
console.log(p && q);
console.log(p || q);
```

5. Bitwise Operators

In JavaScript, bitwise operators are used to perform operations on binary representative of numbers. In other words, the bitwise operator performs the operations by converting the integers into binary form.

- & performs a bitwise AND.
- | performs a bitwise OR.
- ^ performs a bitwise XOR.
- ~ performs a bitwise NOT.

Operator	Description	Example
& (Bitwise AND)	The & operator is used to perform a Boolean AND operation on each bit of its integer argument.	5 & 3 //1
| (Bitwise OR)	It compares the corresponding bits of two operands. If either bit is 1, the result bit will be 1; otherwise, it's 0.	5 | 3 // 7
^ (Bitwise XOR)	It returns 1 if the bits are different and 0 if they are the same.	5 ^ 3 //6
~ (Bitwise NOT)	It inverts all the bits of its operand. It changes each 0 to 1 and each 1 to 0.	~5 // -6
<< (Left shift)	It shifts the bits of a number to the left by a	5 << 1 // 10

	specified number of positions.	
>> (right shift)	It moves the bits of a number to the right by a specified number of positions.	-10 >> 1 //-5
>>> (zero-fill right shift)	It shifts the bits of a number to the right by a specified number of positions and fills the vacated bits on the left and zeros.	-10 >>> 1 //2147483643

Example

```
console.log(3 & 1);
```

6. Ternary Operator

In JavaScript, the ternary operator makes use of three operands. It is also known as a conditional operator.

Example

```
const age = 10;
const status= age >= 18? "Adult": "Minor";
console.log(status);
```

7. Delete Operator

In JavaScript, the delete operator is used to delete an object's property. The delete operator deletes the value as well as the property. Once deleted, the property cannot be used unless it has been added back.

The delete operator can be used only on properties of objects and not on variables or functions.

Example

```
const student = {
   rollNum: 27,
   Name: "James Bond",
   Grade: "A",
   Age:19
};
```

```
delete student.Age;
```

```
console.log(student);
```

8. String Operator

In JavaScript, string operators take two strings and combine them into one single string. String operators are binary operators and you need to use the + symbol between the two string operands to concatenate them into a single string.

Example

```
str1 = "SardarAzeem";
str2 = "Tech";
result = str1 + str2;
console.log(result);
```

9. Typeof Operator

In JavaScript, the typeof operator is an operator that is used for type checking and returns the data type of the operand passed to it. The operand can be any variable, function, or object whose type you want to find out using the typeof operator.

Example

```
console.log(typeof "SardarAzeemech");
console.log(typeof 50);
console.log(typeof false);
console.log(typeof {});
console.log(typeof undefined);
```

10. Instanceof Operator

Instanceof operator checks if the given object is an instance of the specified object. If it is then we get true, else we get false.

Example

```
const n = [1, 2, 23, 4];
console.log(n instanceof Array);
console.log(n instanceof Object);
console.log(n instanceof Number);
console.log(n instanceof String);
```

What Are JavaScript Comments?

The JavaScript comments are a meaningful way to deliver the message. It is used to add information about the code, warnings or suggestions so that the end user can easily interpret the code.

Lines of text that are ignored by the JavaScript engine during code execution are known as comments. For developers who read or update the code afterwards, they act as notes or clarifications.

Effective use of comments is helpful:

- Make the code easier to read.
- Describe complex logic.
- Parts of the code can be temporarily disabled for debugging.
- Provide future developers with helpful documentation.

Types of JavaScript Comments

There are two types of comments in JavaScript.

1. Single-line Comment
2. Multi-line Comment

JavaScript Single line Comment

It is represented by double forward slashes (//). It can be used before and after the statement.

Comments of just one line are wanted for small observations or explanations. Two forward slashes (//) are used at the start. JavaScript ignores anything that comes after // on the same line.

Syntax

The single-line comment is displayed in the syntax.

```
// This is a single-line comment
```

Example 1

```
//A single line comment
sole.log("SardarAzeem");
```

Example 2

```
let a = 10;// declare a, give it the value of 10
let b = a + 21; // declare b, give it the value of a + 21
console.log(b);// printing b
```

JavaScript Multi-line Comment

A forward slash with an asterisk represents it, then an asterisk with a forward slash.

You can write longer descriptions with multi-line comments. They end with */ and start with /*. All of it is considered a comment.

Syntax

/ your code here */*

Example

```
/* It is a multi-line comment.
It will not be displayed */
console.log("Welcome to SardarAzeem");
```

JavaScript Variables

In JavaScript, a variable is a named storage location that holds a value, which can be any data type, such as numbers, strings, objects, etc. It can be declared using keywords like var, let, or const.

- Variables as Placeholders for unknown values: They can be used in places where the values they represent are unknown when the code is written.
- Variables as Code Clarifies: They can make the purpose of your code cleaner.

Rules of Naming Variables in JavaScript

- Name must start with a letter (a to z or A to Z), underscore(_), or dollar($) sign.
- After the first letter, we can use digits (0 to 9), for example, value1.
- JavaScript variables are case-sensitive; for example, x and X are different variables.

Declaring Variables in JavaScript

You can use the "var" or "let" keywords to declare text as a variable. There are three methods to declare a variable in JavaScript:

Using var keyword

In JavaScript, variables were traditionally declared using the var keyword. It is not used in contemporary programming due to certain problems with scope and complicated operations.

```
var a = "SardarAzeem";
console.log(a);
```

Using let keyword

The variables with block scope are declared using the let keyword. It is only available within the block in which they are specified in the JavaScript function.

```
let a = 20;
console.log(a);
```

Using const keyword

Constant variables are used to avoid modification, and they cannot change value after they are declared variables. It is a represented variable using the const keyword in JavaScript.

```
const p = "SardarAzeem";
console.log(p);
```

JavaScript Variable Data Types

Different types of data can be stored in JavaScript variables. These are the primary types:

- String: Strings represent text values.
- Number: The number represents numerical values.
- Boolean: True or false is represented by boolean.
- Undefined: A declared variable without a value given to it.
- Null:It indicates a purposefully empty value.
- Object: Key-value pairs are represented by objects.
- Array: A collection of values is represented by an array.

Types of Variables

The types of variables are used according to operation and values. If you have to use a single value multiple times, then use global; otherwise, use the local variable. There are following two types of variables in JavaScript:

- Local variable
- Global variable

JavaScript Local Variable

JavaScript local variables are declared inside the curly braces {} or a function. In JavaScript, variables are accessible within the function or block only where they are declared.

Local variables with the same name can be used in different functions or blocks. Local variables are deleted after the function is completed.

Example 1

```
myfunction();


function myfunction(){
   //Local variable
   let word = "SardarAzeem";
   console.log(word);
}


console.log(word);
```

Example 2

```
mydemo1();
mydemo2();
let word;
function mydemo1(){
   //Local variable
   let word = "SardarAzeem";
   console.log(word);
}
function mydemo2(){
   //Local variable
   let word = "SardarAzeem";
   console.log(word);
}
console.log(word);
```

JavaScript Global Variable

In JavaScript, Global variables are those variables that are declared outside of any function and can be accessible from anywhere within the script, including these functions.

Example:

```
var Grade = "B";
// Declaring global variable outside the function
myFunction();
// Global variable accessed from
// Within a function
function myFunction() {
    console.log("global value of Grade is: ", Grade);
}
// Changing value of global
// Variable from outside of function
{
    Grade = "A";
    console.log("local value of Grade is: ", Grade);
}
```

The Best Ways to Use JavaScript Variables

- Use let and const: Avoid using var to avoid scope-related problems by using let and const instead.
- Use meaningful variable names: Select names that are descriptive and help the reader understand the code.
- Use the camelCase naming convention: JavaScript variables should be named using the camelCase format (myVariableName as an example).
- Use const: when the variable should not change or unwanted modifications should be avoided.
- Declare variables at the start of a block: the variable declaration is used to improve readability and prevent problems with hoisting.
- Clear the global variables: use local variables whenever feasible to avoid unforeseen changes.

Data Types in JavaScript

There are two types of variables in JavaScript-data type and user-defined. So, in sum, there exist five data types in JavaScript which are described in the following:

Number

The number is one of the primitive data types in JavaScript. Of special interest is that JavaScript uses a single data type for numbers. It does not distinguish between float, decimal, or double types like most programming languages.

Example

```
let a = 12;
console.log(a)
let b = 10.3;
console.log(b)
let c = Infinity;
console.log(c)
let d = 'something here too' / 2;
console.log(d)
```

String

String is one of the most basic data types in JavaScript. It is, in essence, a sequence of characters or words. Let's look at the following example.

Example

```
let a = "TpointTech";
console.log(a);
let b = 'Single quotes work fine';
console.log(b);
let c = `can embed ${a}`;
console.log(c);
```

Boolean

Boolean is one of the primitive data types in JavaScript. It can take two values: true and false. In JavaScript, Boolean values are used to evaluate conditions. This feature is very helpful in validating different scenarios in the language.

Example

```
let p = true;
console.log(p);
let q = false;
console.log(q);
```

Undefined

Undefined is a basic data type in JavaScript. It is any variable that can be declared but not assigned any value. Thus, such a variable automatically contains the default value of undefined within JavaScript. In short, it identifies a variable that does not have an assigned value.

Example

```
let a;
console.log(a);
```

Null

In JavaScript, null is a primitive data type. There are times when it is imperative to assign the Null value if a particular value needs to be assigned as deliberately empty. For example, during runtime, there may be some condition that calls for user input, and for such a condition, the Null data type can be used.

Example

```
let age = null;
console.log(age)
```

Non-Primitive Data Types in JavaScript

The non-primitive data types available in JavaScript include both Objects and Arrays. In addition, ECMAScript has added another data type known as Symbol.

Object

The Object data type is a core element of the JavaScript programming language. Objects can be created using object literal syntax, which is defined by key-value pairs.

To better understand this concept, one can analyze the following example.

Example

```
let tpointTech = {
   type: "Company",
   location: "Noida"
}
console.log(tpointTech.type)
console.log(tpointTech.location)
```

Arrays

An array is a special object which is defined to hold a sequential list of values, and it can store values of different data types.

Example

```
let a = [1, 2, 3, 4, 5];
console.log(a);
let b = [1, "two", { name: "Object" }, [3, 4, 5]];
console.log(b);
```

Function

A function in JavaScript is a reusable piece of code that is defined specifically to execute a particular task when it is called.

Example

```
// Defining a function to greet a user
function greet(name) { return "Hello, " + name + "!"; }
// Calling the function
console.log(greet("Alice"));
```

Date Object

JavaScript's Date object is designed for date and time management, and it supports the creation, manipulation, and formatting of date values.

Example

```
// Creating a new Date object for the current date and time
let currentDate = new Date();
// Displaying the current date and time
console.log(currentDate);
```

Regular Expression

A Regular Expression (RegExp) in JavaScript is an object which is used to define patterns to search for text within strings.

Example

```
// Creating a regular expression to match the word "hello"
let pattern = /hello/;
// Testing the pattern against a string
let result = pattern.test("Hello, TpointTech");
console.log(result);
```

JavaScript Control Structures

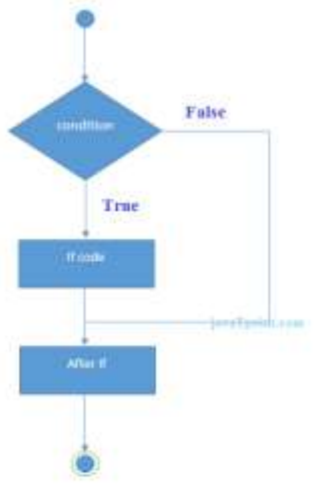
The JavaScript if-else statement is used *to execute the code whether condition is true or false*. There are three forms of if statement in JavaScript.

1. If Statement
2. If else statement
3. if else if statement

JavaScript If statement

It evaluates the content only if expression is true. The signature of JavaScript if statement is given below.

```
if(expression){
//content to be evaluated
}
```



Flowchart of JavaScript If statement

```
<script>
var a=20;
if(a>10){
document.write("value of a is greater than 10");
}
</script>
```

JavaScript If...else Statement

It evaluates the content whether condition is true of false. The syntax of JavaScript if-else statement is given below.

```
if(expression){
//content to be evaluated if condition is true
}
else{
//content to be evaluated if condition is false
}
```

Example of if-else statement in JavaScript to find out the even or odd number.

```
<script>
var a=20;
if(a%2==0){
document.write("a is even number");
}
else{
document.write("a is odd number");
}
</script>
```

JavaScript If...else if statement

It evaluates the content only if expression is true from several expressions. The signature of JavaScript if else if statement is given below.

```
if(expression1){
//content to be evaluated if expression1 is true
}
else if(expression2){
//content to be evaluated if expression2 is true
}
else if(expression3){
//content to be evaluated if expression3 is true
}
```

```
else{
//content to be evaluated if no expression is true
}
```

Example

```
<script>
var a=20;
if(a==10){
document.write("a is equal to 10");
}
else if(a==15){
document.write("a is equal to 15");
}
else if(a==20){
document.write("a is equal to 20");
}
else{
document.write("a is not equal to 10, 15 or 20");
}
</script>
```

JavaScript Loops (For, While, Do-While, For…of, For…in)

In JavaScript, a loop is a programming tool that is used to repeat a set of instructions. Loops are used to reduce repetitive tasks by repeatedly executing a block of code as long as a specified condition is true.

Loops in JavaScript make the code more concise and efficient. The loops are used to iterate the piece of code using for, while, do-while, or for-in loops.

Types of Loops in JavaScript

There are several types of loops present in JavaScript. Such as:

1. JavaScript for Loop
2. JavaScript while Loop
3. JavaScript do-while Loop
4. JavaScript for…of Loop
5. JavaScript for…in Loop

JavaScript for Loop

JavaScript for loop is a control flow statement that allows code to be executed repeatedly based on a condition. It contains initialization, condition, and increment/decrement in one line.

Syntax

```
for(initialization; condition; increment/decrement){
//code
}
```

Example

```
for(let i= 1; i<=5; i++){
   console.log("TpointTech");
}
```

JavaScript while Loop

In JavaScript, the while loop creates a loop that is executed as long as a specific condition is true. JavaScript while loop will continue to run, the condition is evaluated as false.

In the while loop, we specify the condition before the loop, and usually, some variable is incremented or changed in the while loop body to determine when the loop should stop.

Syntax

```
while(condition){
   //code block to be executed
}
```

Example

```
let a = 0;
while(a <= 3){
   console.log("Welcome to TpoinTech");
   a++;
}
```

JavaScript do-while Loop

Do while loop in JavaScript is a statement used to create a loop that executes a block of code once, then checks if a condition is true, and then repeats the loop as long as the condition remains true.

In JavaScript, do-while loops are used when the loop body needs to be executed at least once. The loops end when the condition is false.

Syntax

```
do{
//code to execute
}while(condition)
```

Example

```
let p = 1;
let q = 1;
do{
   pp = p + q;
   console.log(p);
   q++;
}while(q<5)
```

JavaScript for...of loop

In JavaScript, the for...of loop iterates over an object's values rather than their keys. With the help of this you can directly access the items as opposed to index-reference. Some of the iterable objects are as follows:

- An array of elements.
- A string of characters.
- A map of key/value pairs.

Syntax

The syntax of for...of loop is as follows:

```
for(variable of iterable){
   //code to execute
}
```

Example

```
const items = ['BMW', 'Ferrari', 'Mustang'];
for(const item of items){
   console.log(item);
}
```

JavaScript for…in Loop

The for…in loop in JavaScript is used to iterate over the properties of an object. It only iterates over keys of an object that have their enumerable property set to "true."

Syntax

```
for(key in object){
//code to execute
}
```

Example

```
const items = {Phone: 2, Laptop: 1, TV: 1};
for(const Appliance in items){
   console.log(Appliance);
}
```

How to Choose the Right Loop?

In JavaScript, loops are handy if you want to run the same code repeatedly, each time with different values. You can choose the right loop based on:

- When the number of iterations is known, then use the for loop.
- When the condition depends on dynamic factors, use the while loop.
- When you ensure the block executes at least once, use the do-while loop.
- When you want to iterate over object properties, use the for…in loop.
- When you want to iterate through an iterable object, use the for…of loop.

JavaScript Functions

In JavaScript, a function is a reusable chunk of code created to carry out a certain operation. After processing some optional input, it produces an optional output. Large programs can be divided into smaller, more manageable components with the use of functions.

A function is defined using the function keyword, followed by the function name, parentheses for parameters, and curly braces, which contain the code to be executed.

Rules for naming functions:

It must be case-sensitive.

It must start with an alphabetical character (A-Z) or an underscore symbol.

It cannot contain spaces.

It cannot be used as reserve words.

How to declare a Function?

To declare a function, we have to use the reserved keyword "function", followed by its name and a set of arguments.

Syntax

```
function functionName([arg1, arg2, ...argN]){
 //code to be executed
}
```

In the above syntax, a function is a reserved keyword and "functionName" is a name given to the function. JavaScript Functions can have 0 or more arguments.

Example

```
function greet(name) {
    console.log("Hello, " + name + "!");
 }
greet("Alice"); // Output: Hello, Alice!
```

Parameters: variable that is defined in the function's parentheses, used to receive input values when the function is called.

Arguments: The actual values passed to the function when it is called.

Return values: Functions can optionally return a value using the return keyword, which can be used to store the result of the function's execution.

Function Expressions

In JavaScript, the function can also be defined with the use of the expression. A function expression can be stored in a variable:

```
const x = function (a, b) {return a * b};
```

Once, a function expression has been stored in a variable, the variable can be used as a function:

Example

```
const x = function (a, b) {return a * b};
let z = x(4, 8);
console.log(z);
```

JavaScript Function Methods

Method	Description
apply()	It is used to call a function contains this value and a single array of arguments.
bind()	It is used to create a new function.
call()	It is used to call a function contains this value and an argument list.
toString()	It returns the result in the form of a string.

Types of JavaScript Functions

There are several types of JavaScript functions, such as:

Arrow functions

In JavaScript, arrow functions are a simple syntax for writing functions, which was introduced in ES6, and they do not bind their own context.

Syntax:

```
const givenfunctionName = (parameters) => expression;
```

Example

```
const p = ["Water", "Air", "Light", "Earth"];
const pp2 = p.map(function (s) {
    return s.length;
});
console.log("Normal way ", p2);
const pp3 = p.map((s) => s.length);
console.log("Using arrow Function ", p3);
```

Callback Functions

In JavaScript, a callback function is passed into another function as a parameter and is executed after the completion of that function.

Example

```
function showData(name, amt) {
alert(' Hello ' + name + '\n Your entered amount is ' + amt);
}
function getData(callback) {
var name = prompt(" Welcome to the TpointTech.com \n What is your name? ");
var amt = prompt(" Enter some amount...");
callback(name, amt);
}

getData(showData);
```

Anonymous Functions

In JavaScript, anonymous functions are functions without a name. These types of functions are often used as arguments to other functions.

Example

```
let x = function () {
   console.log('It is an anonymous function');
};
x();
```

Example

```
// Regular Function.
function hello()
{
   console.log("Regular function");
};
// Regular Function execution.
hello();

// IIFE creation and execution.
(function() { console.log("Immediately Invoked Function Expression"); })();
```

Nested Functions

In JavaScript, functions defined within other functions is known as nested functions. Nested functions have access to the variables of their parent function.

Example

```
function outerFun(a) {
   function innerFun(b) {
      return a + b;
   }
   return innerFun;
}


const addTen = outerFun(11);
console.log(addTen(5));
```

Advantages of JavaScript Functions

There are several advantages of using functions in JavaScript. Such as:

Reusability

Functions allow you to write a block of code once and use it multiple times throughout your applications, which saves time and effort.

Modularity

JavaScript functions help structure your code by grouping related logic into smaller units, which makes it easier to understand, maintain and debug.

Improved Readability

By using descriptive function names and breaking down complex tasks, functions make your code easier to read and understand.

Easier Maintenance

Changes or updates can be made to a function in one place without affecting other parts of the program, which simplifies the maintenance process.

Debugging

When errors occur, they can be localized to specific functions, which makes it easier to identify and fix problems.

JavaScript Objects

In JavaScript, an object is a variable that can hold multiple values. It is a location where a collection of values is stored. Objects are among the most basic data types in JavaScript. Every object contains properties and types, which are a single unit. Objects are not primitive, unlike primitive data types.

For instance, take a football. A football has properties like weight, shape, color, and design. In this case, the football is an object, and its weight and shape are its properties.

Similarly, in JavaScript, objects have properties that define their nature. An object can hold various types, like Strings, Numbers, Booleans, Arrays, Functions, etc.

Syntax of Object in JavaScript

```
// Object literal syntax
const myObject = {
  key1: value1,
  key2: value2,
  // more key-value pairs...
};
```

1. Curly braces are used to demarcate the start and termination of an object literal.
2. Inside these braces, you define key-value pairs separated by commas.
3. Every key is a string (or symbol in ES6+).
4. A colon comes after each key, followed by its value.
5. They may be of any data type like strings, numbers, booleans, arrays, functions, or other objects.

Creating a JavaScript Object

JavaScript provides several means of object construction, and flexibility as well as versatility are both achievable. The different methods of object construction in JavaScript are:

1. Using an Object Literal
2. Employing the JavaScript Keyword new
3. Constructing a constructor object

Using an Object Literal

Object literal is the easiest method of declaring an object in JavaScript. It is the process of declaring key-value pairs in curly brackets {} to declare an object.

Example

```
let person = {
    name: 'Harsh',
    age: 30,
    city: 'Ghaziabad'
};
```

Using new keyword

You can also create instances using constructor functions with the new keyword. Constructor functions enable you to create multiple objects with the same methods and properties.

Example

```
function Person(name, age, city) {
    this.name = name;
    this.age = age;
    this.city = city;
}
// Creating a new instance of Person
let person1 = new Person('Evan', 30, 'Paris');
let person2 = new Person('Anupam', 25, 'Delhi');
```

JavaScript Objects are Mutable

JavaScript objects are mutable, i.e., you can change their properties even after the object has been initialized. This makes you capable of adding or removing properties, and changing existing ones.

Example

```
let person = {
    name: 'Aman',
    age: 33,
    city: 'Delhi'
};
// Modifying property value
person.age = 31;
// Adding a new property
```

```
person.email = 'aman@example.com';
// Removing a property
delete person.city;
```

Characteristics of Objects in JavaScript

1. Objects are slightly tricky as they can contain any collection of reference data types and simple data types.
2. A reference data type is used to label characters or numbers with a reference value, the reference character or number referencing its position.
3. As an object is being passed from one function to another, this pointer points to where the object is.
4. A class holds methods and attributes.

JavaScript Built-In Methods

Here are some commonly used JavaScript methods:

create(): This method is utilized to create JavaScript objects based on a prototype object, as mentioned earlier.

entries(): This method accepts a JavaScript object as an argument and returns an array that contains arrays of key-value pairs. For instance, let's revisit our "student" object.

keys(): The keys() method takes a JavaScript object as input and returns an array of its property names.

values(): Similarly, the values() method takes a JavaScript object as input and returns an array of its values.

is(): This method takes a second object as an argument and checks if both objects are equal, returning a Boolean value. If the objects are equal, it returns "true"; otherwise, it returns "false."

Object.create()

Object.create() method creates a new object and links it to an existing object.

Code

```
const people = {
printIntro: function () {
console.log(`My full name is ${this.Fname}. Am I Alive? ${this.isalive}`);
}
};
const name = Object.create(people);
```

```
name.Fname = "Robert"; // "Fname" is a property set on "name", but not on "people"

name.isalive = true;
name.printIntro();
```

Object.entries()

The Object.entries() returns the object's array consisting of enumerable properties [key, value] pair.

Syntax:

```
Object.entries(obj)
```

Code

```
const student = { 1: 'Steve', 100: 'Sanju', 45: 'Chris' };
console.log(Object.entries(student));
```

Object.keys()

The Object.keys() method gives you back the enumerable properties of an array or an array-like object, even if the keys are in a random order.

Code

```
let employee_detail = {
employee_name: 'Smith',
employee_salary: 144,
employee_age : 23
} ;
console.log(Object.keys(employee_detail).length);
```

Object.values()

The `Object.values()` function gives you an array of values of an object property. It gives you the values of the object and returns them in an array form.

Code

```
// Returning property values of an array
var check = ['x', 'y', 'z'];
console.log(Object.values(check));
```

Object.is()

The JavaScript Object.is() method is a useful function that allows you to see whether two values are exactly equal. It does so without type coercion, i.e., it looks at values as they are. And, it even accounts for some special cases, such as having differentiating between positive and negative zero.

Code

```
console.log(Object.is(5, 5)); // true
console.log(Object.is('Java', 'Java')); // true
console.log(Object.is(true, false)); // false
console.log(Object.is(0, -0)); // false
console.log(Object.is(NaN, NaN)); // true
```

JavaScript Arrays

In JavaScript, an array is a collection of multiple values stored at different memory locations but sharing the same name. You can access the values in an array by using the indexes within square brackets, starting from 0 up to the length of the array minus 1([0]…[n-1]).

Syntax

The syntax of arrays in JavaScript is as follows:

```
const array_name = [item1, item2,.....];
```

Example

```
const age = [20, 22, 21, 10, 12];
console.log(age);
```

How to Create Arrays in JavaScript?

There are 3 ways to create an array in JavaScript. Such as:

- Array Literals
- Using new keyword
- Array Constructor

Array Literal

An array literal is a list of zero or more expressions, each of which represents an array element, enclosed in square brackets([]).

Syntax

The array can be created with an array literal by using the syntax below.

```
var arrayname=[value1, value2.....valuen];
```

Example

```
let country = [ "India", "Australia", "England" ];
for (let i = 0; i<country.length; i++ ){
console.log( country[i]);
}
```

Using a new keyword

With the use of a new keyword in JavaScript, you can easily create an array.

Syntax

```
var array_name = new Array ();
```

Example

```
let num;
let country = new Array ();
country[0] = "Japan";
country[1] = "Spain";
country[2] = "Germany";
for ( num = 0; num<country.length; num++ )
{
console.log( country[num]);
}
```

Array Constructor

You can create an array instance by passing arguments into a constructor.

Example

```
let employee = new Array("Rohit", "Vivek", "Jhon");
for (let i=0;i<employee.length;i++){
console.log(employee[i]);
}
```

Basic Operations on JavaScript Arrays

In JavaScript, arrays offer several operations for managing and manipulating data efficiently.

Accessing Elements of an Array

In JavaScript, when you want to access an element at a specific index in an array, you can use brackets[] with the index number.

Example

```
let cars = ["BMW", "Buggati", "Skyline"];
console.log(cars[1]);
```

Accessing the First Element of an Array

By using index 0 you can access the first element of an array.

Example

```
let cars = ["BMW", "Buggati", "Skyline"];
console.log(cars[0]);
```

Accessing the Last Element of an Array

By using the index length - 1, you can access the last element of an array.

Example

```
let cars = ["BMW", "Buggati", "Skyline"];
let last = cars[cars.length-1];
console.log(last);
```

Modifying the Array Elements

By assigning a new value to a specific index, you can modify elements in an array.

Example

```
let cars = ["BMW", "Buggati", "Skyline"];
cars[1] = "Ferrari";
console.log(cars);
```

Adding Elements to the Array

By using the push() Method, you can add elements to the end of an array.

Example

```
let cars = ["BMW", "Buggati", "Skyline"];
cars.push("GTR");
console.log(cars);
```

Removing Elements from an Array

By using methods like pop(), shift(), or splice(), you can remove elements from an array.

Example

```
let cars = ["BMW", "Buggati", "Skyline"];
cars.pop(); // removes the last element
console.log(cars);
```

JavaScript Array Methods

Methods	Description
concat()	It returns a new array object that contains two or more merged arrays.
copywithin()	It copies the part of the given array with its own elements and returns the modified array.
entries()	It creates an iterator object and a loop that iterates over each key/value pair.
every()	It determines whether all the elements of an array are satisfying the provided function conditions.
flat()	It creates a new array carrying sub-array elements concatenated recursively till the specified depth.
flatMap()	It maps all array elements via mapping function, then flattens the result into a new array.
fill()	It fills elements into an array with static values.
from()	It creates a new array carrying the exact copy of another array element.
filter()	It returns the new array containing the elements that pass the provided function conditions.
find()	It returns the value of the first element in the given array that satisfies the specified condition.

findIndex()	It returns the index value of the first element in the given array that satisfies the specified condition.
forEach()	It invokes the provided function once for each element of an array.
includes()	It checks whether the given array contains the specified element.
indexOf()	It searches the specified element in the given array and returns the index of the first match.
isArray()	It tests if the passed value ia an array.
join()	It joins the elements of an array as a string.
keys()	It creates an iterator object that contains only the keys of the array, then loops through these keys.
lastIndexOf()	It searches the specified element in the given array and returns the index of the last match.
map()	It calls the specified function for every array element and returns the new array
of()	It creates a new array from a variable number of arguments, holding any type of argument.
pop()	It removes and returns the last element of an array.
push()	It adds one or more elements to the end of an array.
reverse()	It reverses the elements of given array.
reduce(function, initial)	It executes a provided function for each value from left to right and reduces the array to a single value.
reduceRight()	It executes a provided function for each value from right to left and reduces the array to a single value.
some()	It determines if any element of the array passes the test of the implemented function.
shift()	It removes and returns the first element of an array.
slice()	It returns a new array containing the copy of the part of the given array.
sort()	It returns the element of the given array in a sorted order.
splice()	It add/remove elements to/from the given array.
toLocaleString()	It returns a string containing all the elements of a specified array.

toString()	It converts the elements of a specified array into string form, without affecting the original array.
unshift()	It adds one or more elements in the beginning of the given array.
values()	It creates a new iterator object carrying values for each index in the array.

JavaScript DOM

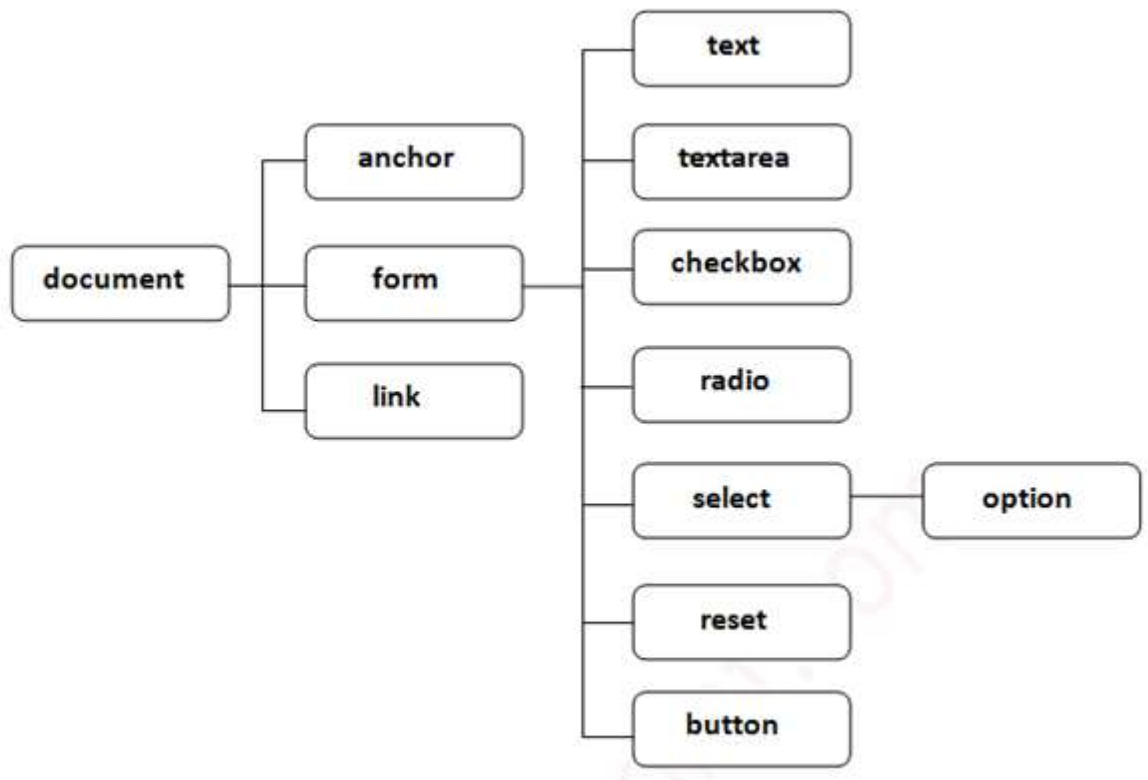
JavaScript Document Object Model, also known as JavaScript DOM, is an interface that is used for web documents, specifically for HTML or XML. By using JavaScript DOM, we can represent the structure of a document as a tree-like structure of nodes, which allows us to access, modify and manipulate the content, structure, as well as style of a web page.

A web page is a document that can be either displayed in the browser window or as the HTML source. In both cases, it is the same document, but the Document Object Model(DOM) representation allows it to be manipulated.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>JavaScript DOM</title>
</head>
<body>
<h1>Hello, World!</h1>
</body>
</html>
```

Properties of Document Object Model



Methods of JavaScript DOM

In JavaScript, with the use of methods, we can access and modify the contents of the document. There are some important methods of the Document Object Model as follows:

Method	Description
document.write("string")	It writes the given string on the document.
document.writeln("string")	It writes the given string on the document with a newline character at the end.
getElementById()	It returns the element having the given id value.
getElementByName()	It returns all the elements having the given name value.
getElementByClassName()	It returns all the elements having the given class name.
getElementByTagName()	It returns all the elements having the given tag name.

querySelector()	It returns the first elements matching a CSS selector.
querySelectorAll()	It returns all elements matching a CSS selector.

JavaScript getElementById()

In JavaScript, the getElementById() is a built-in function that allows you to select an HTML element using its unique ID attribute.

Syntax

The syntax of getElementById is as follows:

```
document.getElementById(elementID);
```

Example

```
<html>
 <head>
  <title>getElementById example</title>
 </head>
 <body>
  <p id="demo">JavaScript DOM</p>
  <button onclick="changeColor('purple');">purple</button>
  <script>
     function changeColor(newColor) {
     const elem = document.getElementById("demo");
     elem.style.color = newColor;
   }
  </script>
 </body>
</html>
```

JavaScript getElementsByName()

In JavaScript, the getElementsByName() method returns a collection of elements with a specified name.

Syntax

The syntax of getElementsByName is as follows:

```
document.getElementsByName(elementName);
```

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>JavaScript DOM</title>
</head>
<body>
  <h2>The getElementByName method</h2>
  Phone:
  <input name="devices" type ="checkbox" value="Phone">
  Laptop:
  <input name="devices" type ="checkbox" value="Laptop">
  Sofa:
  <input name="furnitures" type ="checkbox" value="Sofa">
  <p>Check the checkboxes that have name = "devices"</p>
<script>
  const collection = document.getElementsByName("devices");
  for(let i =0; i <collection.length; i++){
    if(collection[i].type =="checkbox"){
      collection[i].checked = true;
    }
  }
</script>
</body>
</html>
```

JavaScript getElementsByClassName()

In JavaScript, the getElementsByClassName() returns a collection of elements with a specified class.

Syntax

The syntax of getElementsByClassName is as follows:

document.getElementByClassName(classname);

Example

```
<!DOCTYPE html>
<html>
<head>
 <title>getElementsByClassName Example</title>
</head>
<body>
 <p class="highlight">This is paragraph 1.</p>
 <p class="highlight">This is paragraph 2.</p>
 <p>This is a normal paragraph.</p>
 <button onclick="highlightParagraphs()">Highlight Paragraphs</button>
 <script>
  function highlightParagraphs() {
   // Get all elements with the class 'highlight'
   var elements = document.getElementsByClassName("highlight");
   // Loop through the elements and change their background color
   for (var i = 0; i < elements.length; i++) {
    elements[i].style.backgroundColor = "yellow";
   }
  }
 </script>
</body>
</html>
```

JavaScript getElementsByTagName()

In JavaScript, the getElementsByTagName() is a built-in method that returns all the elements having the given tag name.

Syntax

The syntax of getElementsByTagName is as follows:

```
getElementsByTagName(tagName)
```

Example

```
<!DOCTYPE html>
<html>
<head>
<title>My Page</title>
</head>
<body>
<script type="text/javascript">
function countpara(){
var totalpara=document.getElementsByTagName("p");
alert("total p tags are: "+totalpara.length);
}
</script>
<p>This is a paragraph</p>
<p>Here we are going to count a total number of paragraphs by the getElementByTa
gName() method.</p>
<p>Let's see the simple example</p>
<button onclick="countpara()">count paragraph</button>
</body>
</html>
```

JavaScript querySelector()

In JavaScript, the querySelector() method returns the first element that matches a CSS selector.

Syntax

The syntax of querySelector is as follows:

```
querySelector(selectors)
```

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>querySelector Example</title>
</head>
<body>
  <p id="myParagraph">Click the button to change this text.</p>
  <button id="myButton">Click Me</button>
  <script>
    const button = document.querySelector('#myButton');

    button.addEventListener('click', () => {
      const paragraph = document.querySelector('#myParagraph');
      paragraph.textContent = 'The text has been changed!';
    });
  </script>
</body>
</html>
```

JavaScript querySelectorAll()

In JavaScript, querySelectorAll() returns all the child elements that match a CSS selector.

Syntax

The syntax of querySelectorAll() is as follows:

```
element.querySelectorAll(CSS selectors)
```

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>querySelectorAll Example</title>
  <style>
    .highlight {
      font-weight: bold;
      color: blue;
    }
  </style>
</head>
<body>
  <h2 class="highlight">Heading One</h2>
  <h2 class="highlight">Heading Two</h2>
  <h2>Heading Three</h2>
  <script>
   const elements = document.querySelectorAll('.highlight');
   elements.forEach((element) => {
  element.style.color = 'red';
    });
  </script>
</body>
</html>
```

Features of JavaScript DOM

There are some features of JavaScript DOM such as:

Dynamic Manipulation

JavaScript can dynamically add, remove, and modify HTML elements, attributes, and styles, which creates interactive and dynamic web pages.

Event Handling

In JavaScript, the DOM helps to respond to user actions and other events that occur on the page, such as button clicks or form submissions.

Tree structure

JavaScript DOM represents HTML documents as a tree structure where each element is a node, which allows for efficient traversal and manipulation of the document's elements.

DOM Traversal

JavaScript can navigate the DOM tree with the use of various methods to find specific elements and access their properties and methods.

Exception Handling in JavaScript

An exception signifies the presence of an abnormal condition which requires special operable techniques. In programming terms, an exception is the anomalous code that breaks the normal flow of the code. Such exceptions require specialized programming constructs for its execution.

What is Exception Handling?

In programming, exception handling is a process or method used for handling the abnormal statements in the code and executing them. It also enables to handle the flow control of the code/program. For handling the code, various handlers are used that process the exception and execute the code. For example, the Division of a non-zero value with zero will result into infinity always, and it is an exception. Thus, with the help of exception handling, it can be executed and handled.

In exception handling:

A throw statement is used to raise an exception. It means when an abnormal condition occurs, an exception is thrown using throw.

The thrown exception is handled by wrapping the code into the try…catch block. If an error is present, the catch block will execute, else only the try block statements will get executed.

Thus, in a programming language, there can be different types of errors which may disturb the proper execution of the program.

Types of Errors

While coding, there can be three types of errors in the code:

1. Syntax Error: When a user makes a mistake in the pre-defined syntax of a programming language, a syntax error may appear.
2. Runtime Error: When an error occurs during the execution of the program, such an error is known as Runtime error. The codes which create runtime errors are known as Exceptions. Thus, exception handlers are used for handling runtime errors.

3. Logical Error: An error which occurs when there is any logical mistake in the program that may not produce the desired output, and may terminate abnormally. Such an error is known as Logical error.

Error Object

When a runtime error occurs, it creates and throws an Error object. Such an object can be used as a base for the user-defined exceptions too. An error object has two properties:

1. name: This is an object property that sets or returns an error name.
2. message: This property returns an error message in the string form.

Although Error is a generic constructor, there are following standard built-in error types or error constructors beside it:

1. EvalError: It creates an instance for the error that occurred in the eval(), which is a global function used for evaluating the js string code.
2. InternalError: It creates an instance when the js engine throws an internal error.
3. RangeError: It creates an instance for the error that occurs when a numeric variable or parameter is out of its valid range.
4. ReferenceError: It creates an instance for the error that occurs when an invalid reference is de-referenced.
5. SyntaxError: An instance is created for the syntax error that may occur while parsing the eval().
6. TypeError: When a variable is not a valid type, an instance is created for such an error.
7. URIError: An instance is created for the error that occurs when invalid parameters are passed in encodeURI() or decodeURI().

Exception Handling Statements

There are following statements that handle if any exception occurs:

- throw statements
- try…catch statements
- try…catch…finally statements.

JavaScript try…catch

A try…catch is a commonly used statement in various programming languages. Basically, it is used to handle the error-prone part of the code. It initially tests the code for all possible errors it may contain, then it implements actions to tackle those errors (if occur). A good programming approach is to keep the complex code within the try…catch statements.

Let's discuss each block of statement individually:

try{} statement: Here, the code which needs possible error testing is kept within the try block. In case any error occur, it passes to the catch{} block for taking suitable actions and handle the error. Otherwise, it executes the code written within.

catch{} statement: This block handles the error of the code by executing the set of statements written within the block. This block contains either the user-defined exception handler or the built-in handler. This block executes only when any error-prone code needs to be handled in the try block. Otherwise, the catch block is skipped.

Note: catch {} statement executes only after the execution of the try {} statement. Also, one try block can contain one or more catch blocks.

Syntax:

```
try{
expression; } //code to be written.
catch(error){
expression; } // code for handling the error.
```

try...catch example

```
<html>
<head> Exception Handling</br></head>
<body>
<script>
try{
var a= ["34","32","5","31","24","44","67"]; //a is an array
document.write(a);    // displays elements of a
document.write(b); //b is undefined but still trying to fetch its value. Thus catch bloc
k will be invoked
}catch(e){
alert("There is error which shows "+e.message); //Handling error
}
</script>
</body>
</html>
```

Throw Statement

Throw statements are used for throwing user-defined errors. User can define and throw their own custom errors. When throw statement is executed, the statements present after it will not execute. The control will directly pass to the catch block.

Syntax:

```
throw exception;
```

try...catch...throw syntax

```
try{
throw exception; // user can define their own exception
}
catch(error){
expression; }  // code for handling exception.
```

The exception can be a string, number, object, or boolean value.

throw example with try...catch

```
<html>
<head>Exception Handling</head>
<body>
<script>
try {
  throw new Error('This is the throw keyword'); //user-defined throw statement.
}
catch (e) {
 document.write(e.message); // This will generate an error message
}
</script>
</body>
</html>
```

try...catch...finally statements

Finally is an optional block of statements which is executed after the execution of try and catch statements. Finally block does not hold for the exception to be thrown. Any exception is thrown or not, finally block code, if present, will definitely execute. It does not care for the output too.

Syntax:

```
try{
expression;
}
catch(error){
expression;
}
finally{
expression; } //Executable code
```

try...catch...finally example

```
<html>
<head>Exception Handling</head>
<body>
<script>
try{
var a=2;
if(a==2)
document.write("ok");
}
catch(Error){
document.write("Error found"+e.message);
}
finally{
document.write("Value of a is 2 ");
}
</script>
</body>  </html>
```

JavaScript Events

The change in the state of an object is known as an Event. In html, there are various events which represents that some activity is performed by the user or by the browser. When javascript code is included in HTML, js react over these events and allow the execution. This process of reacting over the events is called Event Handling. Thus, js handles the HTML events via Event Handlers.

For example, when a user clicks over the browser, add js code, which will execute the task to be performed on the event.

Event Handler Uses:

It can be used directly within HTML elements by adding special attributes to those elements. They can also be used within the <script> tags or in external JavaScript files.

Some of the HTML events and their event handlers are:

Mouse events:

Event Performed	Event Handler	Description
click	onclick	When mouse click on an element
mouseover	onmouseover	When the cursor of the mouse comes over the element
mouseout	onmouseout	When the cursor of the mouse leaves an element
mousedown	onmousedown	When the mouse button is pressed over the element
mouseup	onmouseup	When the mouse button is released over the element
mousemove	onmousemove	When the mouse movement takes place.

Keyboard events:

Event Performed	Event Handler	Description
Keydown & Keyup	onkeydown & onkeyup	When the user press and then release the key

Form events:

Event Performed	Event Handler	Description
focus	onfocus	When the user focuses on an element
submit	onsubmit	When the user submits the form
blur	onblur	When the focus is away from a form element
change	onchange	When the user modifies or changes the value of a form element

Window/Document events

Event Performed	Event Handler	Description
load	onload	When the browser finishes the loading of the page
unload	onunload	When the visitor leaves the current webpage, the browser unloads it
resize	onresize	When the visitor resizes the window of the browser
reset	onreset	When the window size is resized
scroll	onscroll	When the visitor scrolls a scrollable area

Click Event

```
<html>
<head> Javascript Events </head>
<body>
<script language="Javascript" type="text/Javascript">
   <!--
   function clickevent()
   {
      document.write("This is Learn Java Script By Sardar Azeem");
   }
   //-->
</script>
<form>
<input type="button" onclick="clickevent()" value="Who's this?"/>
</form>
</body>
</html>
```

MouseOver Event

```
<html>
<head>
<h1> Javascript Events </h1>
</head>
<body>
<script language="Javascript" type="text/Javascript">
   <!--
   function mouseoverevent()
   {
      alert("This is Learn Java Script By Sardar Azeem");
   }
   //-->
</script>
```

```
<p onmouseover="mouseoverevent()"> Keep cursor over me</p>
</body>
</html>
```

Focus Event

```
<html>
<head> Javascript Events</head>
<body>
<h2> Enter something here</h2>
<input type="text" id="input1" onfocus="focusevent()"/>
<script>
<!--
   function focusevent()
   {
      document.getElementById("input1").style.background=" aqua";
   }
//-->
</script>
</body>
</html>
```

Keydown Event

```
<html>
<head> Javascript Events</head>
<body>
<h2> Enter something here</h2>
<input type="text" id="input1" onkeydown="keydownevent()"/>
<script>
<!--
   function keydownevent()
   {
      document.getElementById("input1");
      alert("Pressed a key");   }  //-->  </script>  </body>  </html>
```

Load event

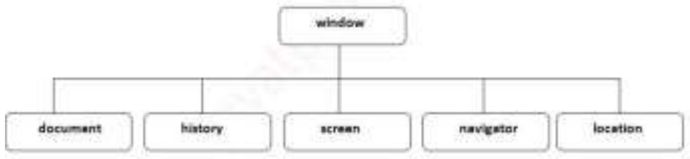
```
<html>
<head>Javascript Events</head>
</br>
<body onload="window.alert('Page successfully loaded');">
<script>
<!--
document.write("The page is loaded successfully");
//-->
</script>
</body>
</html>
```

Browser Object Model

Browser Object Model (BOM)

The Browser Object Model (BOM) is used to interact with the browser.

The default object of browser is window means you can call all the functions of window by specifying window or directly. For example:



window.alert("hello Learn Java Script By Sardar Azeem");

is same as:

alert("hello Learn Java Script By Sardar Azeem");

You can use a lot of properties (other objects) defined underneath the window object like document, history, screen, navigator, location, innerHeight, innerWidth,

Note: The document object represents an html document. It forms DOM (Document Object Model).