

Website Development Essentials

By Sardar Azeem

INTERNET AND WORLD WIDE WEB INTERNET

The Internet is a global network system that has dramatically transformed trade and communication. The network of networks, the Internet connects computer systems across the globe, enabling them (and thus, their users) to interact efficiently. Although its popularity rose gradually, about half of the world's population is now an Internet user, utilizing the Internet in everyday life.

The Internet pervades our daily lives, from reserving plane tickets to purchasing a pair of socks. And without it, the rise of some of the world's largest corporations, such as Facebook, Microsoft, and Amazon, would have been inconceivable.

However, as the saying goes, "*With every blessing comes a curse*". And the Internet is no different. It has simplified our lives, yet whether intentionally or not, we have all become its slaves.

Advantages and Disadvantages of the Internet

Advantages	Disadvantages
Global connections through virtual communications	Loss of personal information
E-commerce	Spread of fake news
Online Education	Internet addiction and time wastage
Abundance of information	Physical and mental health issues

Advantages of Internet

1. Virtual Interview Calls

Consider the following scenario: you're sitting in your home in a rural section of Kerala and you're scheduled for a face-to-face interview in Ontario, Canada. Meanwhile, you contact the support staff to request an alternate form of communication for the interview calls, as you do not have a passport to travel abroad. The support staff sends you an email informing you that they provide the facility of video conferencing. That's all, you're now free to proceed with the interview.

Though Skype, Google Meet, Cisco WebEx, and others deserve recognition, it all began with the first spark for the development of ingenious time-saving tools since the

emergence of internet technology. But that doesn't mean you do not apply for a passport!

2. A Global Connector

The growth of social media platforms like Instagram, Facebook, WhatsApp, Twitter, and other communication forums has paved the way for creating a global social network where anyone may reach out to resolve their concerns and remain connected to family groups and friends. Spending time with people they love is no longer dependent on their physical proximity.

3. Online Services and E-Commerce

With the help of the internet, we can browse the whole inventory of any online store from the comfort of our homes and reap the benefits of online shopping. Additionally, small businesses and individuals use the Internet to expand their reach into the worldwide market. Artists, craftspeople, and others living in remote places can now sell their pieces of art in the global market via the Internet. Not only has the internet proved to be a boon for online shoppers, but you can also access online services. Online booking is available for most essential services, because of which activities like booking a cab, ordering meals, or contacting a mechanic have gotten significantly easier.

4. Online Education

Have you ever considered studying at prestigious universities such as MIT, Harvard, Stanford, Cambridge, IIT, NIT, MIT (Madras Institute of Technology), IISc, or IIM but were unable to pursue the opportunity? Our best buddy "The Internet" has enabled these prestigious universities to provide lecture lessons to other college students via YouTube (e.g. MIT courseware, CS50, etc.), NPTEL.

In today's competitive world, relying solely on traditional education may leave you lacking the critical abilities to compete with your peers. The Internet has enabled us to acquire new skills through online courses from industry professionals, as well as gather a wealth of knowledge about various technologies, current events, sports, art, and culture. The greatest way to illustrate the internet's benefit is to imagine that you're reading about the internet via the internet.

5. A Blessing In Disguise: Internet in COVID-19

COVID-19 shattered the lives and well-being of most people and reinforced geographical barriers. The Internet was a blessing in disguise, enabling individuals seeking help to find the right resources. Additionally, by posting videos on YouTube (on topics ranging from cooking to handicrafts to online business to art creation), Instagram, and LinkedIn, people were able to discover their true potential and use it to make a difference in their communities.

6. Abundant Information

The Internet is full of essential information, be it on financial matters, government legislation and service, market information and economic affairs, technological information, educational and academic issues, and new ideas. It includes a wide range of topics, from scientific publications to topics geared toward children.

Disadvantages of Internet

While we do understand the benefits of having an internet connection, there are certain downsides to using the internet frequently. Let's take a look at some of the negative consequences of excessive internet consumption:

1. Loss of Personal Information

When using social networking websites or online banking services, consumers enter their personal information including email addresses, bank account numbers, credit card details, and phone numbers. This information can be easily accessed by a software expert or a hacker, resulting in privacy exposure or even identity theft. Viruses are designed by hackers to infiltrate computer systems and wreak havoc, further leading to privacy concerns. Online frauds have become increasingly common in recent years, and it's imperative to keep a check on what we share on the world wide web.

2. Internet Addiction

No doubt the internet has made people its slave, reducing offline social abilities. This has led to an adverse impact on people's mental health. Elders and children alike have been impacted. Mobile phone apps like YouTube allow children as young as two to see online videos. They are spending more time in front of a computer screen, which has stifled their cerebral development.

3. Time Wastage

On the Internet, it's easy to lose track of time. Surfing might take a lot longer than you think, and you may not even know it. This also leads to an increase in screen time, which has a negative effect on our memory, especially in children. We aren't using our brains as much as we used to because of the rise of the Internet. When we were younger, we used to remember phone numbers, addresses, and PINs in our heads, as well as perform simple math without the aid of our phones. However, we increasingly rely on the Internet for everything.

4. Obesity and Other Health Issues

Excessive consumption of the internet promotes a sedentary lifestyle, which in turn can lead to weight gain. Being constantly on the Internet, playing games, using

streaming services, or social networking can lead to an unhealthy physique - like poor eyesight, excessive weight gain, etc. In certain cases, the constant use of the internet and/or its misuse also increased mental health issues in teenagers and young adults.

5. Fake News

The spread of fake news is one of the biggest disadvantages of the internet. Since there are limited to no controls in place on a communication forum, website, or social media platform, fake news often goes viral - risking the well-being of entire communities.

Internet Ethics or Cyber Ethics

Internet Ethics or Cyber Ethics can be described as acceptable behavior standards to be followed by digital users while using the internet. They help digital citizens stay safe online by setting up a set of moral principles that govern the usage of Computers and Internet.

Key Areas of Internet Ethics:

1. Privacy:

- Data Protection: Ensuring that personal data is collected, stored, and used responsibly.
- Surveillance: Ethical considerations around government and corporate monitoring of online activities.
- Consent: Obtaining informed consent before collecting or sharing personal information.

2. Security:

- Cybersecurity: Protecting systems, networks, and data from unauthorized access or attacks.
- Ethical Hacking: Engaging in hacking activities with the intent to improve security rather than harm.
- Responsibility: Organizations must take reasonable steps to protect user data.

3. Intellectual Property:

- Copyright: Respecting the rights of creators by not engaging in unauthorized copying or distribution of digital content.
- Piracy: The ethical and legal implications of sharing or downloading copyrighted material without permission.

- Open Source: The ethics of sharing and contributing to software and content openly.

4. Digital Divide:

- Access: Ensuring equitable access to the internet and digital technologies for all.
- Education: Promoting digital literacy to enable everyone to participate fully in the digital world.
- Economic Impact: Considering the ethical implications of the gap between those who have access to technology and those who do not.

5. Freedom of Expression:

- Censorship: Balancing the need to protect society from harmful content with the right to free speech.
- Hate Speech: Addressing the ethical concerns around harmful, offensive, or dangerous speech online.
- Content Moderation: The role of platforms in regulating user-generated content.

6. Responsibility and Accountability:

- Anonymity: The ethical implications of anonymous online behavior.
- Cyberbullying: Addressing the moral and ethical issues of harassment and bullying in online spaces.
- Digital Footprint: Awareness of the long-term impact of online actions on one's reputation and others.

7. Ethical Use of AI and Automation:

- Algorithmic Bias: The fairness and transparency of algorithms used in various online platforms.
- Automation: The ethical use of automated systems, including bots, in online environments.
- AI Ethics: Ensuring that artificial intelligence is used in ways that are fair, just, and do not harm society.

Importance of Internet Ethics:

- Trust: Fostering trust in digital interactions is crucial for the growth and sustainability of online communities.

- **Security:** Ethical behavior helps in maintaining the security and integrity of digital systems.
- **Social Good:** Ethical use of the internet promotes positive societal outcomes and prevents harm.
- **Legal Compliance:** Adhering to ethical principles often aligns with legal requirements, reducing the risk of legal repercussions.

World Wide Web (www)

The World Wide Web (WWW) is a vast information space where documents and other web resources are identified by Uniform Resource Locators (URLs) and can be accessed over the Internet. It is one of the most significant innovations of the 20th century, transforming how people access and share information globally.

Key Concepts and Components of the World Wide Web:

1. Web Pages:

- **HTML:** The World Wide Web is built primarily using Hypertext Markup Language (HTML), which structures content on web pages.
- **CSS:** Cascading Style Sheets (CSS) are used to style and design web pages, defining their appearance.
- **JavaScript:** A scripting language that enables interactive features on web pages, such as forms, animations, and dynamic content.

2. URLs (Uniform Resource Locators):

- **Addressing:** URLs are the addresses used to locate web resources on the internet. They typically consist of a protocol (e.g., HTTP or HTTPS), a domain name (e.g., www.example.com), and a path to a specific resource or page.
- **Domains:** The domain name is a human-readable address that corresponds to a numeric IP address used by computers to communicate.

3. Web Browsers:

- **Function:** Web browsers are software applications used to access and display web pages. Examples include Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge.
- **Rendering:** Browsers interpret HTML, CSS, and JavaScript to render web pages visually.

4. Hyperlinks:

- Navigation: Hyperlinks, or links, are a core feature of the World Wide Web. They allow users to navigate between web pages, websites, and different sections within a document.
- Interconnectedness: Hyperlinks create a web-like structure of interconnected documents, allowing for seamless exploration of related content.

5. HTTP/HTTPS:

- Protocols: Hypertext Transfer Protocol (HTTP) and its secure version, HTTPS, are the protocols used for transmitting data between a web server and a browser. HTTPS encrypts data to enhance security.
- Requests/Responses: The communication between browsers and servers involves sending requests (e.g., for a web page) and receiving responses (e.g., the HTML of that page).

6. Web Servers:

- Hosting: Web servers are computers that store web pages and other resources and serve them to users upon request.
- Software: Common web server software includes Apache, Nginx, and Microsoft IIS.

7. Search Engines:

- Indexing: Search engines like Google, Bing, and Yahoo index web pages and provide a means for users to search for information on the web.
- Algorithms: These engines use complex algorithms to rank pages based on relevance, quality, and other factors.

8. Web Standards:

- W3C: The World Wide Web Consortium (W3C) is the main international standards organization for the web. It develops protocols and guidelines to ensure the long-term growth and compatibility of the web.
- Accessibility: Standards also include guidelines to make the web accessible to people with disabilities.

9. Evolution of the Web:

- Web 1.0: The early web, characterized by static pages and limited interactivity.

- **Web 2.0:** Marked by increased user interaction, social media, and dynamic content.
- **Web 3.0:** The emerging concept of a more intelligent, decentralized web, often associated with the use of AI and blockchain technologies.

10. Impact of the World Wide Web:

- **Global Communication:** The WWW has revolutionized communication, enabling instant sharing of information across the globe.
- **E-Commerce:** It has transformed business, giving rise to online shopping, digital marketing, and the global economy.
- **Education:** The web has democratized access to information, making education and knowledge more accessible than ever.
- **Social Interaction:** Social media and other online platforms have changed how people connect, share, and build communities.

Services of Internet

The internet offers a wide range of services that have become integral to everyday life, transforming how people communicate, work, learn, and entertain themselves. Here's an overview of the most significant internet services:

1. Email (Electronic Mail):

- **Communication:** Email allows users to send and receive messages, documents, and other files instantly across the globe.
- **Services:** Popular email providers include Gmail, Outlook, and Yahoo Mail.

2. World Wide Web (WWW):

- **Information Access:** The web provides access to vast amounts of information through websites, blogs, and online articles.
- **E-commerce:** Online shopping platforms like Amazon and eBay enable users to purchase goods and services over the internet.
- **Education:** E-learning platforms and online courses, such as Coursera and Khan Academy, offer educational resources.

3. Search Engines:

- **Information Retrieval:** Search engines like Google, Bing, and DuckDuckGo allow users to search for information, websites, and content on the web.

- SEO (Search Engine Optimization): Techniques used by website owners to increase visibility on search engines.

4. Social Networking:

- Social Media: Platforms like Facebook, Twitter, Instagram, and LinkedIn enable users to connect, share, and interact with friends, family, and professional networks.
- Content Sharing: Users can share photos, videos, and updates, engage in discussions, and follow public figures or interests.

5. Instant Messaging (IM):

- Real-Time Communication: Services like WhatsApp, Telegram, and Messenger allow users to send text messages, voice notes, and images instantly.
- Group Chats: Enables users to create groups for family, friends, or colleagues to communicate collectively.

6. Voice over Internet Protocol (VoIP):

- Internet Calling: VoIP services like Skype, Zoom, and Google Meet allow users to make voice and video calls over the internet.
- Video Conferencing: Used for virtual meetings, webinars, and online collaboration in both personal and professional contexts.

7. Streaming Services:

- Video Streaming: Platforms like YouTube, Netflix, and Hulu offer on-demand video content, including movies, TV shows, and live broadcasts.
- Music Streaming: Services such as Spotify, Apple Music, and Pandora provide access to millions of songs, podcasts, and radio stations.
- Live Streaming: Twitch and YouTube Live allow users to broadcast live video to audiences around the world.

8. Cloud Services:

- Data Storage: Cloud storage services like Google Drive, Dropbox, and OneDrive offer users the ability to store and access files online from anywhere.
- Software as a Service (SaaS): Online applications like Google Workspace, Microsoft 365, and Salesforce are provided via the cloud, eliminating the need for local installation.

9. E-commerce and Online Banking:

- **Online Shopping:** Websites like Amazon, eBay, and Alibaba enable users to purchase goods and services over the internet.
- **Banking and Payments:** Online banking services allow users to manage accounts, transfer money, and pay bills. Payment gateways like PayPal and Venmo facilitate secure online transactions.

10. Forums and Online Communities:

- **Discussion Boards:** Forums like Reddit, Quora, and Stack Overflow allow users to engage in discussions, ask questions, and share knowledge on various topics.
- **Special Interest Groups:** Online communities where users with similar interests can connect, share resources, and collaborate.

11. Online Gaming:

- **Multiplayer Games:** Platforms like Steam, PlayStation Network, and Xbox Live enable users to play video games with others around the world.
- **Esports:** Competitive gaming events are streamed live, allowing users to watch professional gamers compete in real-time.

12. Content Creation and Sharing:

- **Blogs and Vlogs:** Services like WordPress, Medium, and Blogger allow users to create and share written content, while YouTube and TikTok are popular for video content.
- **Podcasting:** Platforms like Anchor and SoundCloud enable users to create, publish, and distribute audio content.

13. File Sharing:

- **Peer-to-Peer (P2P):** Services like BitTorrent allow users to share files directly with others over the internet.
- **File Transfer Services:** Tools like WeTransfer and Google Drive are used for sending large files between users.

14. Online Education:

- **E-Learning Platforms:** Websites like Coursera, Udemy, and Khan Academy provide online courses and educational resources.
- **Virtual Classrooms:** Tools like Google Classroom, Blackboard, and Moodle facilitate online learning and collaboration between students and teachers.

15. Remote Work and Collaboration:

- **Project Management:** Tools like Trello, Asana, and Monday.com help teams manage projects and tasks online.
- **Collaboration Software:** Platforms like Slack and Microsoft Teams facilitate communication and collaboration among remote teams.

16. Virtual and Augmented Reality (VR/AR):

- **Virtual Worlds:** VR platforms like Oculus and AltspaceVR provide immersive experiences for gaming, socializing, and training.
- **AR Apps:** Augmented reality services, like those found in mobile apps, enhance real-world experiences with digital overlays.

17. Internet of Things (IoT):

- **Smart Devices:** IoT refers to the network of connected devices that communicate and share data, such as smart thermostats, security cameras, and wearable technology.
- **Home Automation:** Services that allow users to control home devices remotely via the internet, like smart lighting and appliances.

18. Online Dating:

- **Matchmaking Services:** Websites and apps like Tinder, Bumble, and OkCupid connect people for romantic relationships, friendships, or social networking.

19. News and Information Services:

- **Online News Portals:** Websites like BBC, CNN, and The New York Times provide up-to-date news, articles, and analysis.
- **RSS Feeds:** Allows users to subscribe to and receive updates from their favorite websites and blogs.

20. Virtual Private Networks (VPNs):

- **Security and Privacy:** VPNs like NordVPN and ExpressVPN provide secure and private access to the internet by encrypting data and masking users' IP addresses.

Web Browser

Web browsers are software applications that allow users to access, retrieve, and view content on the World Wide Web. They are the primary tools for navigating the internet, enabling users to interact with websites, multimedia content, and web applications.

Here's an overview of web browsers, including their key features, examples, and how they work:

Key Functions of Web Browsers:

1. Rendering Web Pages:

- **HTML/CSS Parsing:** Web browsers interpret HTML and CSS to display web pages. They render text, images, videos, and interactive elements according to the structure and styling defined by the web page's code.
- **JavaScript Execution:** Browsers execute JavaScript to provide dynamic content and interactivity, such as animations, form validations, and real-time updates.

2. Navigating the Web:

- **URL Bar:** The address bar in a browser allows users to enter a Uniform Resource Locator (URL) to visit specific websites.
- **Hyperlink Navigation:** Users can click on hyperlinks to move from one web page to another, navigating through the interconnected content of the web.

3. Security and Privacy:

- **HTTPS:** Browsers support secure connections using HTTPS, which encrypts data between the browser and the web server to protect against eavesdropping and tampering.
- **Incognito/Private Mode:** Most browsers offer a private browsing mode that doesn't save browsing history, cookies, or form data.
- **Pop-up Blockers and Ad Blockers:** Browsers often include or support extensions that block unwanted pop-ups and advertisements.

4. Extensions and Plugins:

- **Customization:** Users can enhance the functionality of their browser through extensions and plugins, such as ad blockers, password managers, and productivity tools.
- **Add-ons:** Many browsers have dedicated marketplaces for users to find and install these extensions, like the Chrome Web Store or Firefox Add-ons.

5. Tabs and Window Management:

- **Tabbed Browsing:** Modern browsers allow users to open multiple web pages in separate tabs within a single window, making it easier to multitask.
- **Window Management:** Users can open multiple browser windows, each with its own set of tabs.

6. **Bookmarking and History:**

- **Bookmarks:** Browsers let users save their favorite websites for easy access later.
- **Browsing History:** Browsers keep a history of visited websites, which users can review and manage.

7. **Search Integration:**

- **Default Search Engine:** Most browsers have a default search engine integrated into the address bar, allowing users to search the web directly without navigating to a search engine's homepage.
- **Custom Search:** Users can often change the default search engine to their preferred provider.

Popular Web Browsers:

1. **Google Chrome:**

- **Overview:** Chrome is a popular web browser developed by Google. Known for its speed, simplicity, and extensive library of extensions, it is widely used across all major operating systems.
- **Key Features:** Syncing across devices, extensive extension support, strong integration with Google services, and regular updates for enhanced security.

2. **Mozilla Firefox:**

- **Overview:** Firefox is an open-source web browser developed by Mozilla. It emphasizes privacy, customization, and performance.
- **Key Features:** Strong privacy features, customizable interface, wide range of extensions, and support for open web standards.

3. **Apple Safari:**

- **Overview:** Safari is the default web browser for Apple devices, including macOS and iOS. It is optimized for performance and energy efficiency on Apple hardware.

- Key Features: Fast performance on Apple devices, strong privacy protections, integration with Apple's ecosystem (e.g., iCloud), and energy-efficient design.

4. Microsoft Edge:

- Overview: Edge is Microsoft's web browser, originally introduced with Windows 10. The latest version, based on Chromium, offers improved compatibility and performance.
- Key Features: Integration with Windows, support for Chrome extensions, built-in tools for productivity (e.g., collections and vertical tabs), and enhanced security features.

5. Opera:

- Overview: Opera is a web browser known for its innovative features, such as a built-in VPN, ad blocker, and a sidebar for easy access to messaging apps.
- Key Features: Free built-in VPN, integrated ad blocker, customizable interface, and support for Chrome extensions.

6. Brave:

- Overview: Brave is a privacy-focused web browser that blocks ads and trackers by default. It also offers a unique reward system for users who opt into viewing privacy-respecting ads.
- Key Features: Strong privacy protections, built-in ad and tracker blocking, cryptocurrency wallet integration, and faster page loading times.

7. Vivaldi:

- Overview: Vivaldi is a highly customizable web browser developed by former Opera developers. It allows users to tailor nearly every aspect of the browser's interface and functionality.
- Key Features: Extensive customization options, built-in tools like note-taking and screen capture, and support for Chrome extensions.

How Web Browsers Work:

1. DNS Resolution:

- When a user enters a URL, the browser communicates with a Domain Name System (DNS) server to resolve the domain name into an IP address. This IP address is used to locate the web server hosting the requested website.

2. HTTP/HTTPS Request:

- The browser sends an HTTP or HTTPS request to the web server, asking for the specific resource (e.g., a web page). The request includes information such as the browser type, cookies, and other relevant data.

3. Response and Rendering:

- The web server responds with the requested resource, usually in the form of an HTML document. The browser then parses the HTML, along with any CSS and JavaScript files, to render the web page visually on the screen.

4. Interactivity:

- The browser continues to interact with the server as needed, fetching additional resources, handling user inputs, and updating the page dynamically based on JavaScript execution.

Web Server

A web server is a software system that delivers web content and services to users over the internet or an intranet. It hosts websites, processes client requests, and serves the requested web pages to users' browsers. Web servers play a crucial role in the functioning of the World Wide Web by ensuring that websites are accessible to users around the globe.

Key Functions of a Web Server:

1. Hosting Websites:

- **File Storage:** Web servers store website files, including HTML documents, images, CSS stylesheets, JavaScript files, and other resources necessary for a website to function.
- **Database Interaction:** Many web servers interact with databases to retrieve, store, and manage dynamic content, such as user profiles, product listings, and blog posts.

2. Processing Client Requests:

- **HTTP/HTTPS Requests:** Web servers receive requests from clients (typically web browsers) via HTTP or HTTPS protocols. These requests can be for web pages, images, or other resources.
- **Request Handling:** The server processes the request, often by retrieving the requested file or generating content dynamically, and then sends the appropriate response back to the client.

3. Serving Content:

- **Static Content:** This includes fixed files like HTML, CSS, images, and videos that are delivered directly to the user as they are stored on the server.
- **Dynamic Content:** Generated on-the-fly based on user interaction or other factors. For example, a web server might run server-side scripts (like PHP, Python, or Node.js) to generate custom content for each user request.

4. Security:

- **SSL/TLS Encryption:** Web servers often use SSL/TLS certificates to secure data transmission via HTTPS, protecting sensitive information such as passwords and credit card details.
- **Access Control:** Servers can restrict access to certain resources, requiring authentication (e.g., usernames and passwords) before allowing access.

5. Load Balancing:

- **Traffic Management:** Web servers often work with load balancers to distribute incoming traffic across multiple servers, ensuring no single server is overwhelmed and improving performance and reliability.

6. Logging and Monitoring:

- **Access Logs:** Web servers maintain logs of all requests, including the requested resource, the client's IP address, the time of the request, and the response status. These logs are essential for troubleshooting, security auditing, and analytics.
- **Monitoring:** Continuous monitoring tools can track server performance, uptime, and other critical metrics to ensure the server is running smoothly.

Popular Web Server Software:

1. Apache HTTP Server:

- **Overview:** Apache is one of the most widely used web server software systems. It's open-source, highly configurable, and supports a wide range of operating systems, including Linux and Windows.
- **Key Features:** Modular architecture, extensive documentation, SSL/TLS support, and strong community support.

2. Nginx:

- Overview: Nginx (pronounced "engine-x") is a high-performance web server known for its speed, efficiency, and ability to handle high levels of concurrent connections. It's also commonly used as a reverse proxy server.
- Key Features: Load balancing, reverse proxy capabilities, low resource consumption, and support for static and dynamic content.

3. Microsoft Internet Information Services (IIS):

- Overview: IIS is a web server developed by Microsoft, designed to run on Windows Server operating systems. It's deeply integrated with the Windows ecosystem and supports a range of Microsoft technologies.
- Key Features: Strong integration with ASP.NET, comprehensive GUI for management, and extensive support for Windows-based applications.

4. LiteSpeed:

- Overview: LiteSpeed is a commercial web server known for its high performance, security features, and compatibility with Apache configurations. It is often used as a drop-in replacement for Apache.
- Key Features: Built-in DDoS protection, advanced caching mechanisms, and easy integration with control panels like cPanel.

5. Node.js:

- Overview: Node.js is not a traditional web server but a JavaScript runtime that allows developers to create server-side applications using JavaScript. It's often used to build custom web servers.
- Key Features: Non-blocking I/O model, real-time web application support, and a large ecosystem of packages via npm (Node Package Manager).

6. Tomcat:

- Overview: Apache Tomcat is an open-source Java Servlet container that functions as a web server for Java-based applications. It's widely used for hosting Java web applications.
- Key Features: Support for Java Servlet, JSP, and WebSocket technologies, integration with other Apache software, and scalability for enterprise applications.

How a Web Server Works:

1. Client Request:

- A user enters a URL in their web browser or clicks on a hyperlink. This action sends a request to the web server associated with that URL.

2. DNS Resolution:

- The browser sends the domain name to a DNS server, which resolves the domain name into an IP address. This IP address is used to locate the web server hosting the requested website.

3. Connection Establishment:

- The browser establishes a connection with the web server using TCP/IP, typically on port 80 (for HTTP) or port 443 (for HTTPS).

4. Request Handling:

- The web server receives the request, processes it (which may involve retrieving a static file or executing server-side code), and prepares a response.

5. Response:

- The web server sends the requested content back to the client's browser, usually in the form of an HTML document, along with associated resources like images, CSS, and JavaScript files.

6. Rendering:

- The browser renders the received content, displaying the web page to the user. The user can then interact with the page, possibly generating more requests.

Role of Web Servers in Modern Web Applications:

- **Scalability:** Web servers are designed to handle thousands or even millions of requests per second. Modern architectures often employ multiple web servers in a load-balanced environment to distribute traffic and ensure scalability.
- **Microservices:** In cloud-based and microservices architectures, web servers often host microservices that communicate via APIs (Application Programming Interfaces) to provide complex functionality.
- **Content Delivery Networks (CDNs):** Web servers work in conjunction with CDNs to cache and deliver content from geographically distributed servers, reducing latency and improving load times for users worldwide.

- **Serverless Computing:** Some modern applications use serverless architectures, where code is executed in response to events, and the underlying server infrastructure is abstracted away. However, web servers still play a role in handling HTTP requests and routing them to the appropriate functions.

Web Directories

Web directories are organized collections of websites and online resources, categorized and listed based on topics, subjects, or themes. Unlike search engines, which index the web automatically using algorithms, web directories are typically curated by human editors who evaluate and categorize websites manually. This human curation ensures that the listed sites are relevant, trustworthy, and of high quality.

Key Features of Web Directories:

1. Categorization:

- **Hierarchical Structure:** Web directories are organized into categories and subcategories, much like a library's classification system. For example, a directory might have a top-level category like "Health," with subcategories such as "Nutrition," "Diseases," and "Fitness."
- **Topic-Specific Listings:** Websites are listed under specific categories that best describe their content, making it easier for users to find relevant resources.

2. Human Curation:

- **Quality Control:** Human editors review submissions to ensure that only reputable and relevant websites are included. This reduces the likelihood of spam, low-quality content, or irrelevant sites.
- **Content Descriptions:** Each listing in a web directory typically includes a brief description of the website, helping users understand its purpose and content before visiting.

3. Search Functionality:

- **Search within Directory:** While primarily organized by categories, many web directories also offer a search feature that allows users to find listings by entering keywords or phrases.
- **Filter Options:** Users can often filter search results by category, language, region, or other criteria to narrow down their options.

4. Submission Process:

- Site Submission: Website owners can submit their sites to be listed in a web directory. This submission often requires providing details such as the website's title, URL, description, and category.
- Approval and Listing: After submission, an editor reviews the site for relevance, quality, and adherence to the directory's guidelines before it is approved and listed.

5. Regional or Niche Directories:

- Local Directories: Some web directories focus on specific geographic regions, listing websites relevant to a particular country, state, or city.
- Niche Directories: These directories specialize in a specific industry or topic, such as business, education, technology, or health.

Examples of Web Directories:

1. DMOZ (Open Directory Project):

- Overview: DMOZ was one of the largest and most comprehensive web directories, created and maintained by a community of volunteer editors. It categorized websites across a wide range of topics and regions.
- Status: DMOZ was discontinued in 2017, but its legacy continues through clones and archives.

2. Yahoo! Directory:

- Overview: Yahoo! Directory was one of the earliest and most popular web directories, organizing websites into categories for easy navigation. It played a significant role in the early days of the web before being discontinued in 2014.
- Legacy: Yahoo! Directory was instrumental in shaping how people navigated the web in the late 1990s and early 2000s.

3. Best of the Web (BOTW):

- Overview: BOTW is a long-standing web directory that categorizes websites across various industries and topics. It offers both free and paid listings, with a focus on quality and relevance.
- Key Features: High editorial standards, a wide range of categories, and a reputation for listing quality websites.

4. Business.com:

- Overview: Business.com started as a web directory focused on business-related websites and resources. It serves as a B2B directory, connecting businesses with vendors, products, and services.
- Key Features: Focus on business categories, useful for entrepreneurs, small businesses, and corporate users.

5. Jasmine Directory:

- Overview: Jasmine Directory is a hand-picked web directory that emphasizes quality over quantity. It covers a variety of topics and industries, with detailed descriptions for each listing.
- Key Features: Strict editorial review, focus on trustworthy sites, and detailed categories.

How Web Directories Differ from Search Engines:

1. Curation vs. Automation:

- Web Directories: Curated by humans, with editors evaluating each site before listing it. This ensures high-quality and relevant results but can limit the number of listings.
- Search Engines: Use automated bots (crawlers) to index millions of web pages, providing a vast array of results. However, the quality and relevance of these results depend on algorithms, which can sometimes include irrelevant or low-quality sites.

2. Purpose and Use Cases:

- Web Directories: Best for finding vetted, high-quality websites within specific categories. Useful for niche research, business listings, and discovering reputable resources.
- Search Engines: Ideal for broad searches and finding information on almost any topic. Useful for quick searches, finding recent content, and discovering new websites.

3. Search Scope:

- Web Directories: Typically cover a smaller, more curated subset of the web, focusing on specific categories or regions.
- Search Engines: Index a vast portion of the internet, providing results from a much broader range of websites.

Advantages and Disadvantages of Web Directories:

Advantages:

- **Quality Control:** Listings are reviewed by human editors, reducing the chances of encountering spam, malware, or low-quality content.
- **Organized Browsing:** The hierarchical structure allows users to easily navigate and explore related websites within a category.
- **Niche Focus:** Specialized directories can be invaluable for finding resources in specific fields or industries.

Disadvantages:

- **Limited Listings:** Due to the manual submission and review process, web directories may not have as many listings as search engines.
- **Outdated Information:** Some web directories may not be updated frequently, leading to outdated or inactive links.
- **Declining Popularity:** With the rise of advanced search engines, the use of web directories has declined, making them less relevant in some contexts.

The Role of Web Directories Today:

While the popularity of web directories has waned with the advent of sophisticated search engines like Google, they still serve important roles, especially in niche industries and for specialized research. Some modern directories have evolved to focus on quality over quantity, offering curated lists of reputable sites in specific areas of interest.

Websites

are collections of interlinked web pages that are hosted on a web server and accessible through the internet via a web browser. They serve as the digital presence for individuals, businesses, organizations, and governments, providing information, services, products, and entertainment to users worldwide.

Key Components of a Website:

1. **Web Pages:**
 - **HTML Documents:** Each web page is typically an HTML document that can include text, images, videos, and other multimedia elements. HTML (HyperText Markup Language) is the standard language used to create web pages.
 - **CSS (Cascading Style Sheets):** CSS is used to style the appearance of web pages, controlling layout, colors, fonts, and other visual aspects.

- JavaScript: JavaScript is a scripting language used to add interactivity and dynamic content to web pages, such as animations, form validations, and real-time data updates.

2. Domain Name:

- URL (Uniform Resource Locator): The address of a website on the internet, such as "www.example.com." A domain name is a human-readable version of an IP address, which identifies the location of the website's server.
- Top-Level Domain (TLD): The suffix at the end of a domain name, such as ".com," ".org," ".net," ".edu," and country-specific TLDs like ".uk" or ".jp."

3. Web Hosting:

- Web Server: A server that stores website files and delivers them to users when requested. The server processes requests and sends the appropriate web pages to the user's browser.
- Hosting Provider: A company that provides the technology and services needed to host a website, including server space, bandwidth, security, and maintenance.

4. Content Management System (CMS):

- Overview: A CMS is a software platform that allows users to create, manage, and modify website content without needing extensive technical knowledge. Popular CMSs include WordPress, Joomla, and Drupal.
- Functionality: CMSs typically offer templates, plugins, and a user-friendly interface for content creation and management.

5. Multimedia Elements:

- Images and Graphics: Visual elements that enhance the user experience, convey information, or add aesthetic value to the website.
- Videos and Animations: Multimedia content that can be embedded in web pages to provide tutorials, demonstrations, entertainment, or marketing content.
- Audio: Sound files, such as podcasts, music, or voiceovers, that can be played directly from a web page.

6. Navigation:

- Menus: A list of links, usually displayed at the top or side of a web page, that allows users to navigate different sections of the website.

- **Internal Links:** Hyperlinks within a website that connect different pages, helping users move through the site seamlessly.
- **Breadcrumbs:** A navigation aid that shows users their current location within the website's hierarchy and provides links to previous sections.

7. Interactive Features:

- **Forms:** Web forms allow users to submit information, such as contact details, feedback, or orders, directly through the website.
- **User Accounts:** Some websites offer user registration, enabling personalized experiences, such as saved preferences, order history, and member-only content.
- **Search Bar:** A feature that allows users to search for specific content within the website.

Types of Websites:

1. Personal Websites:

- **Overview:** Created by individuals to share personal information, blogs, portfolios, hobbies, or resumes. They often serve as a digital presence for the owner.
- **Examples:** Blogs, personal portfolios, hobby sites, and family photo galleries.

2. Business Websites:

- **Overview:** These websites represent businesses, providing information about products, services, contact details, and e-commerce functionality.
- **Examples:** Company websites, e-commerce sites, and online stores.

3. Educational Websites:

- **Overview:** Websites dedicated to providing educational content, resources, and tools for students, teachers, and lifelong learners.
- **Examples:** Online courses, school websites, educational blogs, and e-learning platforms.

4. Government Websites:

- **Overview:** Websites created by local, state, or federal government agencies to provide information, services, and resources to citizens.

- Examples: Government portals, public service websites, and online tax filing services.

5. Nonprofit and Advocacy Websites:

- Overview: Websites that represent nonprofit organizations, charities, or advocacy groups. They often include information about the organization's mission, donation options, and ways to get involved.
- Examples: Charity websites, environmental advocacy sites, and social justice organizations.

6. Entertainment Websites:

- Overview: Websites focused on providing entertainment content, such as movies, music, games, and news.
- Examples: Streaming platforms, gaming sites, celebrity news, and online magazines.

7. News and Media Websites:

- Overview: Websites that deliver news articles, videos, and other media content to the public. They often include real-time updates, opinion pieces, and multimedia content.
- Examples: Online newspapers, news portals, and TV networks' websites.

8. Social Media Websites:

- Overview: Platforms that allow users to create profiles, share content, and interact with others through posts, comments, and messages.
- Examples: Facebook, Twitter, Instagram, LinkedIn.

9. Forums and Community Websites:

- Overview: Websites that provide a platform for users to discuss topics, share advice, and build communities around shared interests.
- Examples: Reddit, Quora, online discussion boards, and niche community sites.

10. Portfolio Websites:

- Overview: Websites created by professionals such as artists, designers, writers, and photographers to showcase their work and attract clients or employers.

- Examples: Online portfolios, photography galleries, and creative showcases.

How Websites Work:

1. Client-Server Interaction:

- User Request: A user enters a website's URL in their browser, sending a request to the web server hosting the website.
- Server Response: The web server processes the request and sends the requested web page (HTML document) back to the user's browser.
- Rendering: The browser interprets the HTML, CSS, and JavaScript files to display the web page on the user's screen.

2. Backend and Frontend:

- Frontend: The part of the website that users interact with directly, including the layout, design, and content. It involves technologies like HTML, CSS, and JavaScript.
- Backend: The server-side part of the website that handles data processing, database management, and server logic. It involves programming languages like PHP, Python, Ruby, or Java, and databases like MySQL or MongoDB.

3. Responsive Design:

- Mobile Optimization: Websites are often designed to be responsive, meaning they automatically adjust their layout and content to fit different screen sizes, such as smartphones, tablets, and desktops.
- Cross-Browser Compatibility: Ensuring that a website functions correctly across different web browsers (e.g., Chrome, Firefox, Safari) is crucial for accessibility.

4. Content Delivery Networks (CDNs):

- Global Distribution: CDNs are networks of servers distributed across multiple locations worldwide. They cache and deliver website content from the server closest to the user, reducing load times and improving performance.

The Role of Websites in the Digital World:

- Information Dissemination: Websites are a primary source of information for billions of users, providing access to knowledge, news, and resources across various fields.

- **Commerce and Services:** E-commerce websites enable businesses to sell products and services online, while service-oriented sites offer everything from banking to healthcare.
- **Communication and Interaction:** Social media and community websites allow people to connect, share ideas, and engage in discussions on a global scale.
- **Entertainment and Culture:** Websites offer a vast array of entertainment options, including streaming services, online games, and cultural content, accessible to users anytime, anywhere.
- **Education and Learning:** Educational websites and platforms provide courses, tutorials, and resources for learners of all ages, making education more accessible.

Static and dynamic websites

Static and dynamic websites differ primarily in how their content is delivered and how they function. Here's a detailed comparison:

Static Websites:

Definition:

- A static website consists of fixed content. Each page is a separate HTML file, and the content displayed to the user does not change unless manually updated by the website owner or developer.

Key Characteristics:

1. **Fixed Content:**
 - The content on a static website remains the same for every user. It is pre-written and stored on the server, and each time a user accesses the website, the same content is delivered.
2. **Simple Development:**
 - Static websites are generally simpler and faster to develop because they consist of straightforward HTML, CSS, and sometimes JavaScript. There's no need for server-side scripting or database interaction.
3. **Low Server Load:**
 - Since static websites don't require server-side processing or database queries, they put less load on the server. This often results in faster loading times.
4. **Security:**

- Static websites are generally more secure than dynamic websites because they do not involve server-side processing or databases, reducing the risk of hacking and exploits.

5. Scalability:

- They are easily scalable because serving static content to a large number of users is straightforward and can be efficiently managed using Content Delivery Networks (CDNs).

6. Updates:

- Content updates must be done manually by editing the HTML files. This can be time-consuming, especially for larger websites.

Examples:

- Personal blogs (without frequent updates)
- Small business websites
- Portfolios
- Brochure sites

Advantages:

- **Speed:** Static websites load quickly due to their simplicity and lack of complex processing.
- **Cost-Effective:** Easier and cheaper to host since they require minimal server resources.
- **Security:** Fewer points of vulnerability due to the absence of server-side logic.

Disadvantages:

- **Limited Functionality:** Cannot handle complex, interactive features like user logins, comments, or real-time updates.
- **Maintenance:** Updating content can be cumbersome, especially for non-technical users or large sites.

Dynamic Websites:

Definition:

- A dynamic website generates content on the fly based on user interactions or other factors. Content can change dynamically, and pages are often generated by server-side scripts and connected to a database.

Key Characteristics:

1. Dynamic Content:

- Content on dynamic websites can change based on user input, preferences, or other criteria. For example, a user might see different content based on their location, login status, or previous interactions.

2. Server-Side Processing:

- Dynamic websites rely on server-side technologies like PHP, ASP.NET, Ruby on Rails, Python (Django, Flask), or Node.js to generate web pages. They often interact with databases to fetch and display content.

3. Interactivity:

- These websites can offer interactive features like user accounts, comments, forms, and real-time data updates. For example, an e-commerce site displays different products and prices based on user choices and inventory levels.

4. Content Management:

- Dynamic websites often use Content Management Systems (CMS) like WordPress, Joomla, or Drupal, allowing non-technical users to update content easily through a user-friendly interface.

5. Personalization:

- Content can be personalized for each user, such as displaying a personalized dashboard, recommendations, or tailored information.

6. Database Connectivity:

- Dynamic websites usually connect to a database to store and retrieve data. This allows for the management of large volumes of content, user data, and transactions.

Examples:

- Social media platforms
- E-commerce sites
- News websites
- Online forums
- Streaming services

Advantages:

- **Functionality:** Can handle complex, interactive features that static websites cannot, such as user authentication, personalized content, and data-driven applications.
- **Ease of Updates:** Content can be updated dynamically through a CMS without the need for direct coding.
- **Scalability:** Suitable for large websites with a lot of content that needs to be managed dynamically.

Disadvantages:

- **Complexity:** More complex to develop and maintain due to the need for server-side scripting, database management, and security measures.
- **Cost:** Typically more expensive to host and manage, as they require more server resources and technical expertise.
- **Security:** Greater risk of security vulnerabilities, such as SQL injection or cross-site scripting (XSS), due to the complexity of server-side processing.

Summary of Differences:

Aspect	Static Website	Dynamic Website
Content	Fixed, does not change	Dynamic, changes based on user interaction
Development	Simple, uses HTML, CSS	Complex, uses server-side scripting
Interactivity	Limited	High, supports interactive features
Server Load	Low, no server-side processing	Higher, requires server-side processing
Security	Generally more secure	Greater risk of vulnerabilities
Update Process	Manual, requires HTML edits	Easy, often through CMS
Use Cases	Simple sites like portfolios, blogs	Complex sites like social networks, e-commerce
Cost	Lower hosting and development costs	Higher due to resource and maintenance needs
Speed	Generally faster to load	Can be slower, depending on server load

Search Engine

A search engine is a software system designed to search for information on the internet. It retrieves and organizes relevant data based on the user's query, presenting it in a ranked list of results. Search engines are one of the most essential tools for navigating the vast amount of information available online.

Key Components of a Search Engine:

1. Web Crawlers (Spiders or Bots):

- **Function:** These are automated programs that systematically browse the web to index pages. Crawlers visit websites, follow links, and gather information from web pages, storing it in the search engine's database.
- **Crawling Process:** The crawler begins with a list of known URLs and fetches the content of these pages. It then follows hyperlinks within these pages to discover new URLs, continuing this process to build a comprehensive index of the web.

2. Indexing:

- **Function:** After crawling, the collected data is organized and stored in an index. The index is a vast database of all the web pages and their content, making it easier and faster for the search engine to retrieve relevant information when a user submits a query.
- **Content Organization:** The index includes details like keywords, meta tags, page titles, and the structure of the content. This organization helps the search engine understand what each page is about.

3. Search Algorithm:

- **Function:** The search algorithm is the heart of the search engine, determining which pages are most relevant to a user's query. It analyzes the indexed data and ranks pages based on various factors, such as keyword relevance, page authority, user engagement, and more.
- **Factors Considered:** Common factors include keyword usage, backlinks (other sites linking to the page), page load speed, mobile-friendliness, and user behavior signals (like click-through rates).

4. Query Processor:

- **Function:** When a user enters a query, the search engine's query processor interprets it and matches it against the indexed content. It

considers factors like spelling corrections, synonyms, and search intent to provide the most accurate results.

- Natural Language Processing (NLP): Advanced search engines use NLP to understand the context and meaning behind a query, improving the accuracy of search results, especially for more complex or conversational queries.

5. Ranking System:

- Function: The ranking system orders the search results based on their relevance and quality. The most relevant and authoritative pages appear higher in the search results, while less relevant pages are ranked lower.
- PageRank: Originally developed by Google, PageRank is one of the many factors used in ranking algorithms, which evaluates the importance of a page based on the number and quality of backlinks it has.

6. Search Results Page (SERP):

- Function: The Search Engine Results Page (SERP) is what users see after submitting a query. It typically includes a list of web pages, along with additional features like ads, images, videos, local results, and rich snippets.
- Features: SERPs often include organic results, paid ads (sponsored links), knowledge panels, featured snippets, "People also ask" sections, and more.

Types of Search Engines:

1. General Search Engines:

- Examples: Google, Bing, Yahoo.
- Function: These search engines index a broad range of websites and provide general-purpose search services. They are designed to answer a wide variety of queries across different topics.

2. Vertical Search Engines:

- Examples: Google Scholar (academic papers), Yelp (local businesses), Zillow (real estate).
- Function: Vertical search engines specialize in a specific niche or industry, focusing on particular types of content like academic papers, job listings, local businesses, or real estate.

3. Metasearch Engines:

- Examples: DuckDuckGo, Dogpile.
- Function: Metasearch engines aggregate results from multiple search engines. They don't have their own index; instead, they pull data from other search engines and present a combined list of results.

4. Private Search Engines:

- Examples: DuckDuckGo, StartPage.
- Function: These search engines emphasize user privacy, avoiding the tracking and profiling practices common in many general search engines. They often do not store user data or use personalized search results.

How Search Engines Work:

1. Crawling:

- The search engine sends out crawlers to explore the web, following links and gathering data from web pages. This process is continuous, with the crawler constantly updating the index with new and updated content.

2. Indexing:

- The collected data is processed and stored in an index, a vast database of all the content the crawler has found. The index is organized so that the search engine can quickly retrieve relevant information when needed.

3. Processing the Query:

- When a user enters a search query, the search engine's query processor interprets it and matches it against the indexed content. The processor considers the intent behind the query, synonyms, and other linguistic factors.

4. Ranking the Results:

- The search engine's algorithm evaluates the indexed pages to determine which ones are most relevant to the user's query. It ranks these pages based on factors like keyword relevance, authority, user experience, and more.

5. Displaying the Results:

- The search engine presents the ranked list of results on the SERP. The results may include a mix of organic listings, ads, images, videos, news, and other specialized content.

Importance of Search Engines:

1. Information Retrieval:

- Search engines are vital tools for finding information quickly and efficiently. They help users locate relevant content across the web, whether for research, shopping, news, or entertainment.

2. Business Visibility:

- For businesses, appearing in search engine results is crucial for visibility and attracting customers. Search engine optimization (SEO) strategies are often employed to improve a website's ranking on SERPs.

3. Content Discovery:

- Search engines help users discover new content, websites, and resources. This is particularly important for content creators, businesses, and marketers looking to reach a broader audience.

4. User Experience:

- The effectiveness of a search engine greatly impacts user experience. A good search engine delivers relevant, accurate, and timely results, enhancing user satisfaction.

Challenges and Considerations:

1. Search Engine Optimization (SEO):

- Websites often employ SEO strategies to improve their ranking in search results. This includes optimizing content, using keywords effectively, building quality backlinks, and ensuring a good user experience.

2. Search Engine Marketing (SEM):

- SEM involves paid advertising on search engines. Businesses pay to have their ads appear at the top of SERPs, often labeled as "sponsored" or "ad."

3. Privacy Concerns:

- Many search engines track user behavior to deliver personalized results and ads. This has raised privacy concerns, leading to the popularity of private search engines that prioritize user anonymity.

4. Algorithm Bias:

- Search engine algorithms can sometimes reflect biases, whether through the ranking of content or the filtering of information. This can impact what

information users are exposed to, potentially reinforcing certain perspectives.

Popular Search Engines:

1. Google:
 - The dominant search engine globally, known for its powerful algorithms, vast index, and user-friendly features like Google Images, Maps, and News.
2. Bing:
 - Microsoft's search engine, known for integrating with other Microsoft services and offering rewards for users who search through its platform.
3. Yahoo:
 - Once a leading search engine, Yahoo now relies on partnerships with other search engines like Bing for its search results.
4. DuckDuckGo:
 - A privacy-focused search engine that does not track users or personalize search results, gaining popularity for its commitment to user privacy.
5. Baidu:
 - The leading search engine in China, offering services similar to Google but tailored to the Chinese language and market.
6. Yandex:
 - A popular search engine in Russia, known for its comprehensive local services and search capabilities.

Web Page Program Development

Web page program development refers to the process of creating, designing, and coding web pages that can be accessed via the internet. This involves a combination of front-end and back-end development, utilizing various programming languages, tools, and frameworks.

Stages of Web Page Program Development:

1. Planning and Requirements Gathering:
 - **Goal Setting:** Define the purpose of the website or web page (e.g., informational, e-commerce, portfolio).

- Audience Analysis: Understand the target audience and their needs.
- Feature Specification: Identify the features and functionalities the web page needs (e.g., forms, user login, interactive elements).
- Content Planning: Plan the content that will be included, such as text, images, videos, and other multimedia.

2. Design:

- Wireframing: Create basic sketches or wireframes that outline the layout and structure of the web page.
- Prototyping: Develop a more detailed prototype that includes design elements like color schemes, typography, and user interface (UI) components.
- Responsive Design: Ensure the design adapts to different screen sizes and devices (mobile, tablet, desktop).
- User Experience (UX): Focus on creating a user-friendly experience, with intuitive navigation and clear calls to action.

3. Front-End Development:

- HTML (HyperText Markup Language):
 - Structure: HTML is the backbone of a web page, defining the structure and content. It uses tags to denote different elements (e.g., headings, paragraphs, links, images).
 - Semantic Markup: Use HTML5 for semantic tags like `<header>`, `<footer>`, `<article>`, and `<section>` to improve accessibility and SEO.
- CSS (Cascading Style Sheets):
 - Styling: CSS is used to style the HTML content, including layout, colors, fonts, and other visual elements.
 - Responsive Design: Use media queries to create responsive designs that adapt to different screen sizes.
 - CSS Frameworks: Utilize frameworks like Bootstrap, Tailwind CSS, or Foundation to speed up development with pre-designed components.
- JavaScript:

- **Interactivity:** JavaScript adds interactivity to web pages, enabling dynamic content, animations, and user interactions (e.g., form validation, modal windows, dropdowns).
- **DOM Manipulation:** JavaScript can manipulate the Document Object Model (DOM) to update content dynamically without reloading the page.
- **JavaScript Frameworks/Libraries:** Use libraries like jQuery or frameworks like React, Angular, or Vue.js to simplify and enhance front-end development.
- **Version Control:**
 - **Git:** Implement version control using Git to track changes in the codebase, collaborate with other developers, and manage code history.
 - **Repositories:** Use platforms like GitHub, GitLab, or Bitbucket to host and manage the codebase.

4. Back-End Development (for Dynamic Web Pages):

- **Server-Side Scripting:**
 - **Languages:** Use server-side languages like PHP, Python (Django, Flask), Ruby (Ruby on Rails), JavaScript (Node.js), or Java (Spring) to handle server-side logic.
 - **Dynamic Content:** Generate dynamic content based on user input, database queries, or other criteria.
- **Database Integration:**
 - **Databases:** Connect the web page to a database (e.g., MySQL, PostgreSQL, MongoDB) to store and retrieve data like user information, product details, or blog posts.
 - **CRUD Operations:** Implement Create, Read, Update, and Delete (CRUD) operations to manage data within the database.
- **APIs (Application Programming Interfaces):**
 - **RESTful APIs:** Develop or consume RESTful APIs to allow communication between the front-end and back-end or to integrate third-party services.

- Authentication: Implement authentication methods like OAuth, JWT (JSON Web Tokens), or session-based authentication for secure user access.
- Security:
 - Data Protection: Ensure data security with practices like encryption (SSL/TLS), secure authentication, and input validation to prevent attacks like SQL injection and cross-site scripting (XSS).
 - User Authentication: Implement secure login and session management to protect user accounts.

5. Testing:

- Unit Testing: Write and run tests for individual components or functions to ensure they work as expected.
- Integration Testing: Test how different components of the web page interact with each other.
- Cross-Browser Testing: Ensure the web page functions correctly across different web browsers (e.g., Chrome, Firefox, Safari, Edge).
- Performance Testing: Analyze and optimize the web page's load time and responsiveness.
- User Testing: Gather feedback from users to identify usability issues and make necessary improvements.

6. Deployment:

- Hosting: Choose a web hosting service to make the web page accessible online. Options include shared hosting, VPS (Virtual Private Server), cloud hosting, or dedicated servers.
- Domain Name: Register a domain name (e.g., www.example.com) and configure DNS settings to point to the hosting server.
- Deployment Tools: Use tools like FTP (File Transfer Protocol), SSH, or continuous integration/continuous deployment (CI/CD) pipelines to deploy the web page to the server.
- SSL Certificate: Implement an SSL certificate to enable HTTPS, ensuring secure communication between the server and users' browsers.

7. Maintenance and Updates:

- **Monitoring:** Continuously monitor the web page for performance, security, and uptime using tools like Google Analytics, New Relic, or other monitoring services.
- **Bug Fixes:** Regularly update the web page to fix bugs, patch security vulnerabilities, and improve performance.
- **Content Updates:** Keep the content fresh and relevant by regularly updating text, images, and other media.
- **Scaling:** As the web page grows, consider scaling the infrastructure (e.g., upgrading hosting plans, optimizing databases) to handle increased traffic.

Tools and Technologies Commonly Used:

1. Text Editors/IDEs:

- **VS Code, Sublime Text, Atom, IntelliJ IDEA:** Popular text editors and integrated development environments (IDEs) for coding.

2. Frameworks and Libraries:

- **Front-End:** React.js, Angular, Vue.js, Bootstrap, Tailwind CSS.
- **Back-End:** Express.js (Node.js), Django (Python), Ruby on Rails, Laravel (PHP).

3. Version Control Systems:

- **Git, GitHub, GitLab, Bitbucket:** For managing code versions and collaboration.

4. Design Tools:

- **Figma, Adobe XD, Sketch:** For creating wireframes, prototypes, and UI designs.

5. Testing Tools:

- **Jest, Mocha, Selenium, Cypress:** For automated testing and ensuring code quality.

6. Deployment Platforms:

- **Netlify, Vercel, Heroku, AWS, DigitalOcean:** For deploying and hosting web pages.

Roles in Web site development team

A website development team typically consists of various roles, each specializing in different aspects of the development process. The specific roles can vary depending on the size and scope of the project, but here are the common roles you might find in a web development team:

1. Project Manager (PM)

- Responsibilities:
 - Oversee the project from inception to completion.
 - Manage timelines, budgets, and resources.
 - Coordinate between different team members and stakeholders.
 - Ensure that the project meets its goals and deadlines.
- Skills:
 - Strong organizational and leadership skills.
 - Excellent communication and problem-solving abilities.
 - Familiarity with project management tools (e.g., Jira, Trello).

2. Business Analyst (BA)

- Responsibilities:
 - Gather and analyze business requirements from stakeholders.
 - Translate business needs into technical specifications.
 - Create detailed documentation and user stories.
 - Ensure that the final product meets business objectives.
- Skills:
 - Analytical and problem-solving skills.
 - Experience in creating requirements documents and user stories.
 - Understanding of both business processes and technology.

3. UI/UX Designer

- Responsibilities:
 - Design the user interface (UI) and user experience (UX) of the website.

- Create wireframes, prototypes, and mockups.
- Conduct user research and usability testing.
- Ensure the website is visually appealing and user-friendly.
- Skills:
 - Proficiency in design tools (e.g., Figma, Adobe XD, Sketch).
 - Understanding of user behavior and design principles.
 - Experience in creating responsive and accessible designs.

4. Front-End Developer

- Responsibilities:
 - Implement the visual and interactive aspects of the website using HTML, CSS, and JavaScript.
 - Ensure the website is responsive and works across different devices and browsers.
 - Collaborate with designers to translate designs into functional web pages.
 - Optimize website performance and load times.
- Skills:
 - Proficiency in HTML, CSS, JavaScript, and front-end frameworks (e.g., React, Vue.js).
 - Knowledge of responsive design techniques and cross-browser compatibility.
 - Familiarity with version control systems (e.g., Git).

5. Back-End Developer

- Responsibilities:
 - Develop and maintain the server-side logic, databases, and APIs.
 - Handle server configuration, data storage, and application security.
 - Integrate the back-end with the front-end to ensure data flow and functionality.
 - Optimize server performance and scalability.

- Skills:
 - Proficiency in server-side languages (e.g., PHP, Python, Ruby, Node.js).
 - Experience with databases (e.g., MySQL, PostgreSQL, MongoDB).
 - Knowledge of server management and API development.

6. Full-Stack Developer

- Responsibilities:
 - Work on both the front-end and back-end of the website.
 - Bridge the gap between design and functionality by handling both client-side and server-side development.
 - Ensure smooth integration between front-end and back-end components.
- Skills:
 - Proficiency in both front-end and back-end technologies.
 - Experience with full-stack frameworks and tools.
 - Strong problem-solving and communication skills.

7. Quality Assurance (QA) Tester

- Responsibilities:
 - Test the website for bugs, usability issues, and compliance with requirements.
 - Create and execute test cases, report defects, and work with developers to resolve issues.
 - Ensure the website meets quality standards before deployment.
- Skills:
 - Experience with manual and automated testing tools.
 - Attention to detail and strong analytical skills.
 - Familiarity with testing frameworks and bug-tracking tools.

8. Content Strategist/Writer

- Responsibilities:
 - Develop and manage the content strategy for the website.

- Write, edit, and proofread website content, including text, images, and multimedia.
- Ensure content is aligned with the website's goals and user needs.
- Collaborate with designers and developers to integrate content effectively.
- Skills:
 - Strong writing and editing skills.
 - Experience with content management systems (CMS).
 - Understanding of SEO and content optimization.

9. SEO Specialist

- Responsibilities:
 - Optimize the website for search engines to improve visibility and ranking.
 - Conduct keyword research, on-page optimization, and backlink strategies.
 - Monitor and analyze website performance using SEO tools.
 - Stay updated with the latest SEO trends and algorithms.
- Skills:
 - Knowledge of SEO best practices and tools (e.g., Google Analytics, SEMrush).
 - Understanding of search engine algorithms and ranking factors.
 - Experience with keyword research and link-building strategies.

10. DevOps Engineer

- Responsibilities:
 - Manage deployment processes and server infrastructure.
 - Automate and streamline development workflows and deployment pipelines.
 - Ensure the website's reliability, scalability, and performance.
 - Monitor and maintain the website's uptime and security.
- Skills:

- Proficiency in deployment tools and automation (e.g., Docker, Jenkins).
- Experience with cloud services (e.g., AWS, Azure, Google Cloud).
- Knowledge of server management and infrastructure as code (IaC).

11. Web Administrator

- Responsibilities:
 - Maintain and manage the website's hosting environment and server configurations.
 - Handle backups, updates, and security patches.
 - Troubleshoot and resolve server and hosting issues.
- Skills:
 - Experience with web hosting platforms and server administration.
 - Knowledge of web server technologies (e.g., Apache, Nginx).
 - Familiarity with security practices and backup solutions.

12. Graphic Designer

- Responsibilities:
 - Create visual assets for the website, such as logos, icons, and graphics.
 - Work with UI/UX designers to ensure visual consistency and branding.
 - Design promotional materials and other visual elements as needed.
- Skills:
 - Proficiency in graphic design software (e.g., Adobe Photoshop, Illustrator).
 - Understanding of visual design principles and branding.
 - Ability to create high-quality, web-ready graphics.

Web Development Scope

Web development scope refers to the range of tasks, features, and functionalities that are defined for a web development project. It outlines what the project will include and what it will not, setting clear boundaries and expectations for stakeholders. Defining the scope is crucial for successful project management, ensuring that resources are used effectively and that the final product meets the intended goals.

Components of Web Development Scope

1. Project Objectives:

- Definition: Clearly define the goals and purpose of the web development project.
- Examples: Create an e-commerce platform, develop a corporate website, build a portfolio site, or establish an online community.

2. Functional Requirements:

- Definition: Outline the specific functionalities and features the website will include.
- Examples: User authentication, shopping cart, search functionality, content management system (CMS), interactive forms, and user profiles.

3. Non-Functional Requirements:

- Definition: Specify the quality attributes and constraints that the project must meet.
- Examples: Performance (e.g., load times), security (e.g., data protection), scalability (e.g., handling increased traffic), and accessibility (e.g., compliance with WCAG).

4. Technical Requirements:

- Definition: Identify the technical specifications and tools that will be used in the project.
- Examples: Technology stack (e.g., HTML, CSS, JavaScript, frameworks), hosting environment (e.g., cloud, shared hosting), and database systems (e.g., MySQL, MongoDB).

5. Design and User Experience (UX):

- Definition: Define the design and UX elements, including the look and feel of the website.
- Examples: Wireframes, prototypes, visual design guidelines, responsive design considerations, and user interface (UI) components.

6. Content Requirements:

- Definition: Specify the content that will be included on the website and how it will be managed.

- Examples: Text, images, videos, blog posts, product descriptions, and content management procedures.

7. Integration Requirements:

- Definition: Detail any external systems or services that the website will need to integrate with.
- Examples: Payment gateways, social media platforms, third-party APIs, and CRM systems.

8. Testing and Quality Assurance (QA):

- Definition: Outline the testing procedures and criteria for ensuring the website meets the defined requirements.
- Examples: Functional testing, usability testing, performance testing, and security testing.

9. Deployment and Maintenance:

- Definition: Define the procedures for deploying the website and maintaining it post-launch.
- Examples: Deployment strategy, hosting setup, update procedures, bug fixes, and ongoing support.

10. Timeline and Milestones:

- Definition: Set the project timeline and key milestones to track progress.
- Examples: Project kick-off, design completion, development phases, testing, and launch date.

11. Budget and Resources:

- Definition: Estimate the budget and resources required for the project.
- Examples: Development costs, design costs, software licenses, and team members' time.

12. Stakeholder Involvement:

- Definition: Identify who will be involved in the project and their roles.
- Examples: Project manager, developers, designers, content creators, and clients.

Scope Management

1. Scope Definition:

- Activities: Document and define the project scope based on initial requirements and objectives.
 - Documents: Scope statement, requirements specification, and project charter.
2. Scope Verification:
- Activities: Regularly review and verify that the project is aligning with the defined scope.
 - Techniques: Stakeholder meetings, status reports, and scope review sessions.
3. Scope Control:
- Activities: Manage changes to the project scope and handle scope creep (uncontrolled changes).
 - Techniques: Change control process, impact analysis, and approval workflows.
4. Scope Documentation:
- Activities: Maintain and update scope-related documents throughout the project lifecycle.
 - Documents: Change logs, updated scope statements, and revised project plans.

Importance of Defining Scope

1. Clarity and Focus:
 - Ensures that all team members and stakeholders have a clear understanding of what the project will deliver.
2. Resource Management:
 - Helps allocate resources effectively and avoid overuse or misallocation.
3. Budget Control:
 - Provides a basis for budgeting and helps prevent cost overruns by defining the project boundaries.
4. Timeline Management:
 - Assists in creating a realistic timeline and scheduling tasks to meet deadlines.

5. Risk Management:

- Identifies potential risks related to scope changes and helps mitigate them.

6. Quality Assurance:

- Ensures that the final product meets the defined requirements and quality standards.

Scripting languages JavaScript and PHP

JavaScript and PHP are both popular scripting languages used in web development, but they serve different purposes and operate in different environments. Here's a comparative overview of both languages:

JavaScript

Overview:

- **Type:** Client-side scripting language (can also be used server-side with Node.js).
- **Usage:** Primarily used for adding interactivity and dynamic content to websites. JavaScript runs in the user's browser, allowing for real-time updates and interaction without requiring a page reload.

Key Features:

- **Interactivity:** Enables dynamic updates to web pages (e.g., form validation, animations, interactive maps).
- **Event Handling:** Responds to user actions such as clicks, hover, and keyboard inputs.
- **DOM Manipulation:** Allows modification of HTML and CSS elements on the fly through the Document Object Model (DOM).
- **Asynchronous Programming:** Supports asynchronous operations with callbacks, promises, and async/await, making it ideal for handling operations like API calls and timers.
- **Frameworks/Libraries:** Popular frameworks and libraries include React, Angular, Vue.js, and jQuery.

Syntax and Examples:

- **Variables:**

javascript

Copy code

```
let name = 'John';
```

```
const age = 30;
```

- Function:

javascript

Copy code

```
function greet(name) {  
  return `Hello, ${name}!`;  
}
```

- DOM Manipulation:

javascript

Copy code

```
document.getElementById('myElement').innerText = 'Updated Text';
```

Strengths:

- **Performance:** Executes directly in the browser, providing immediate feedback and interactivity.
- **Ecosystem:** Rich ecosystem with numerous libraries and frameworks for various functionalities.
- **Versatility:** Can be used on both the client and server sides (Node.js).

Weaknesses:

- **Browser Compatibility:** Differences in how browsers interpret JavaScript can lead to inconsistencies.
- **Security:** JavaScript code is visible to users, which can potentially expose vulnerabilities.

PHP

Overview:

- **Type:** Server-side scripting language.

- Usage: Primarily used for server-side development, generating dynamic content on the server and delivering it to the client. PHP code is executed on the server before the web page is sent to the browser.

Key Features:

- Server-Side Processing: Handles tasks like form submissions, database interactions, and user authentication.
- Integration with Databases: Often used in conjunction with databases like MySQL or PostgreSQL to manage and retrieve data.
- Session Management: Supports session handling for maintaining user state across multiple pages.
- File Handling: Provides functionality for file uploads, downloads, and manipulation on the server.

Syntax and Examples:

- Variables:

php

Copy code

```
$name = 'John';
```

```
$age = 30;
```

- Function:

php

Copy code

```
function greet($name) {
```

```
    return "Hello, $name!";
```

```
}
```

- Database Connection:

php

Copy code

```
$conn = new mysqli('localhost', 'username', 'password', 'database');
```

```
if ($conn->connect_error) {
```

```
die("Connection failed: " . $conn->connect_error);
}
```

Strengths:

- **Server-Side Capabilities:** Efficient at handling server-side operations and integrating with databases.
- **Ease of Use:** Generally considered easy to learn and use for building dynamic websites.
- **Wide Hosting Support:** Supported by most web hosting providers and compatible with various database systems.

Weaknesses:

- **Performance:** Can be slower compared to some modern server-side languages due to its synchronous nature.
- **Security:** Security practices must be carefully followed to prevent vulnerabilities such as SQL injection and cross-site scripting (XSS).

Comparison:

Feature	JavaScript	PHP
Execution	Client-side (browser) and server-side (Node.js)	Server-side
Primary Use	Interactivity and dynamic content	Server-side processing and data management
Syntax	Similar to other C-style languages	Unique syntax with a focus on server-side scripting
Performance	Fast client-side operations	Depends on server and execution environment
Ecosystem	Extensive libraries and frameworks (React, Angular, etc.)	Mature frameworks (Laravel, Symfony)
Security	Can be vulnerable if not properly handled	Requires attention to avoid server-side vulnerabilities

Summary

- JavaScript is essential for creating interactive and dynamic user experiences on the client side, with a growing presence on the server side thanks to Node.js.

- PHP is widely used for server-side scripting, handling backend operations, database interactions, and generating dynamic web content.

Web hosting

Web hosting refers to the service that allows individuals and organizations to publish a website or web application on the internet. Web hosting provides the infrastructure and resources needed to store and serve website files, such as HTML, CSS, JavaScript, images, and databases.

Key Concepts in Web Hosting

1. Types of Web Hosting:

- **Shared Hosting:** Multiple websites share the same server resources (CPU, RAM, storage). It is cost-effective but can have limitations in performance and customization.
- **Virtual Private Server (VPS) Hosting:** A physical server is divided into multiple virtual servers. Each VPS has its own dedicated resources and more control compared to shared hosting.
- **Dedicated Hosting:** The entire server is dedicated to a single website or client. It offers maximum control, performance, and security but is more expensive.
- **Cloud Hosting:** Uses a network of interconnected virtual servers. It offers scalability, reliability, and flexibility, as resources can be scaled up or down based on demand.
- **Managed Hosting:** The hosting provider manages the server and its infrastructure, including updates, security, and backups. This can be applied to shared, VPS, or dedicated hosting.
- **Reseller Hosting:** Allows individuals or businesses to resell hosting services to others. Resellers manage their own customer accounts but use the resources provided by the primary hosting provider.
- **WordPress Hosting:** Optimized specifically for WordPress sites, including features like automatic updates, backups, and WordPress-specific support.

2. Key Components of Web Hosting:

- **Domain Name:** The address of the website (e.g., www.example.com). Domain names need to be registered and linked to the hosting service.

- **Server:** The physical or virtual machine where website files and data are stored. Servers can be managed by the hosting provider or by the client (in the case of dedicated or VPS hosting).
- **Storage:** Space on the server where website files, databases, and other content are stored.
- **Bandwidth:** The amount of data transferred between the server and users. Hosting plans usually come with a certain amount of bandwidth, and exceeding this can result in additional charges or throttling.
- **Email Accounts:** Many hosting plans include email services, allowing users to create custom email addresses associated with their domain (e.g., info@example.com).
- **Control Panel:** A web-based interface for managing hosting features, such as file uploads, email accounts, databases, and domain settings. Common control panels include cPanel, Plesk, and custom dashboards provided by the hosting provider.

3. Key Features to Consider:

- **Uptime Guarantee:** The percentage of time the hosting service is operational. A high uptime guarantee (e.g., 99.9%) is crucial for ensuring your website is available to users.
- **Security:** Features like SSL certificates, firewalls, malware scanning, and backups to protect your website from threats and ensure data integrity.
- **Performance:** Factors like server speed, load times, and resource allocation impact how quickly and efficiently your website loads for users.
- **Customer Support:** Availability of support channels (e.g., phone, chat, email) and the quality of assistance provided by the hosting provider.
- **Scalability:** The ability to upgrade resources (e.g., storage, bandwidth) as your website grows or experiences increased traffic.
- **Backup and Recovery:** Regular backups and recovery options to protect against data loss and ensure quick restoration in case of issues.

4. Popular Web Hosting Providers:

- **Bluehost:** Known for WordPress hosting and reliable customer support.
- **HostGator:** Offers a variety of hosting options, including shared, VPS, and dedicated hosting.

- SiteGround: Known for its performance and customer support, with strong WordPress hosting options.
- Amazon Web Services (AWS): Provides cloud hosting with extensive scalability and flexibility.
- Google Cloud Platform (GCP): Offers cloud hosting with robust infrastructure and global reach.
- DigitalOcean: Provides cloud hosting with a focus on simplicity and developer-friendly features.

Cookie

Cookies are small pieces of data that are stored on a user's device by a web browser while they are browsing a website. They serve various purposes, such as maintaining user sessions, tracking user behavior, and personalizing web content.

Types of Cookies

1. Session Cookies:

- Definition: Temporary cookies that are deleted when the browser is closed.
- Purpose: Used to store information during a browsing session, such as keeping a user logged in or remembering items in a shopping cart.

2. Persistent Cookies:

- Definition: Cookies that remain on the user's device for a specified period, even after the browser is closed.
- Purpose: Used to remember login information, user preferences, or tracking information across multiple sessions.

3. First-Party Cookies:

- Definition: Cookies set by the website that the user is currently visiting.
- Purpose: Used to store information related to the website, such as user preferences and session data.

4. Third-Party Cookies:

- Definition: Cookies set by domains other than the one the user is visiting, typically by third-party advertisers or analytics providers.
- Purpose: Used for tracking user behavior across different websites, serving targeted ads, and collecting analytics data.

5. Secure Cookies:

- Definition: Cookies that are only transmitted over secure (HTTPS) connections.
- Purpose: Used to enhance security by ensuring that cookie data is encrypted during transmission.

6. HttpOnly Cookies:

- Definition: Cookies that are not accessible via JavaScript and can only be used by the server.
- Purpose: Used to help prevent cross-site scripting (XSS) attacks by reducing the risk of cookie theft through client-side scripts.

7. SameSite Cookies:

- Definition: Cookies that have a SameSite attribute which controls how cookies are sent with cross-site requests.
- Purpose: Provides additional security by restricting how cookies are sent in cross-site requests. The SameSite attribute can be set to Strict, Lax, or None.

Key Functions of Cookies

1. Session Management:

- Purpose: Keeps users logged in and manages their session state. For example, cookies are used to store session identifiers for authenticated users.

2. Personalization:

- Purpose: Remembers user preferences and settings to provide a personalized experience. For example, storing language preferences or theme choices.

3. Tracking and Analytics:

- Purpose: Tracks user behavior and collects data for analytics. For example, cookies can track page visits, time spent on a site, and user interactions.

4. Advertising:

- Purpose: Delivers targeted ads based on user behavior and interests. For example, cookies can track users' browsing history to display relevant advertisements.

Managing Cookies

1. Creating Cookies:

- Example (JavaScript):

javascript

Copy code

```
document.cookie = "username=JohnDoe; expires=Fri, 31 Dec 2024 23:59:59 GMT; path="/;
```

2. Reading Cookies:

- Example (JavaScript):

javascript

Copy code

```
const cookies = document.cookie.split('; ');  
const cookie = cookies.find(row => row.startsWith('username='));  
const username = cookie ? cookie.split('=')[1] : null;
```

3. Deleting Cookies:

- Example (JavaScript):

javascript

Copy code

```
document.cookie = "username=; expires=Thu, 01 Jan 1970 00:00:00 GMT; path="/;
```

4. Browser Settings:

- Users can manage cookies through their browser settings, including deleting cookies, blocking cookies from specific sites, or setting preferences for cookie handling.

Privacy and Security

1. Privacy Concerns:

- Cookies can be used to track user behavior, which raises privacy concerns. Users can manage cookie settings and preferences to control how cookies are used.

2. Regulations:

- Various regulations, such as the General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA), require websites to obtain user consent before storing cookies and to provide transparency about cookie usage.

3. Security Measures:

- **Secure Attribute:** Ensures cookies are only sent over secure connections.
- **HttpOnly Attribute:** Helps protect cookies from being accessed by JavaScript.
- **SameSite Attribute:** Helps mitigate cross-site request forgery (CSRF) attacks.

HTML

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: Hypertext 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.

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 in 1991.

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 Code Editors and Execution of HTML Program

An HTML file is a text file, so to create an HTML file we can use any text editors as code editor.

- 1) 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.
- 2) 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)**.

- 3) After learning the basics, you can easily use other professional text editors which are,
 - a. Notepad++
 - b. Sublime Text
 - c. Vim
 - d. VS-Code

HTML Program Execution can be carried out by using any web browser program.

Google Chrome		Safari Web Browser		Maxthon	
Mozilla Firefox		Internet Explorer		Slim Browser	
Opera Web Browser		Slim jet Browser		Netscape Browser	

HTML Program Structure

Most Important Tags Used in Every Html Program Are Listed Below.

<!DOCTYPE>

It defines the document type or it instruct the browser about the version of HTML.

<html > And </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> And </head> (Optional TAG)

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> And </title> (Optional TAG)

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.

<meta> tag(Optional TAG)

HTML <meta> tag is used to represent the metadata about the HTML document. It specifies page description, keywords, copyright, language, author of the documents, etc.

```
<!DOCTYPE html>
<html>
<head>
  <title>Page Title</title>
</head>
<body>
  <h2>Heading Content</h2>
  <p>Paragraph Content</p>
</body>
</html>
```

The metadata does not display on the webpage, but it is used by search engines, browsers and other web services which scan the site or webpage to know about the webpage.

With the help of meta tag, you can experiment and preview that how your webpage will render on the browser.

The <meta> tag is placed within the <head> tag, and it can be used more than one times in a document.

Syntax and Explanation of Meta TAG

1. <meta charset="utf-8">

It defines the character encoding. The value of charset is "utf-8" which means it will support to display any language.

2. <meta name="keywords" content="HTML, CSS, JavaScript, By Sir Azeem">

It specifies the list of keyword which is used by search engines.

3. <meta name="description" content="Free Online tutorials">

It defines the website description which is useful to provide relevant search performed by search engines.

4. <meta name="author" content="Sardar Azeem">

It specifies the author of the page. It is useful to extract author information by Content management system automatically.

5. <meta name="refresh" content="50">

It specifies to provide instruction to the browser to automatically refresh the content after every 50sec (or any given time).

<body> and </body>

Body tag contains everything you want to display on the Web Page. In other words the whole code of the webpage is written between <body> and </body>. So, we can say that body is an essential tag of every HTML program.

How To Type, Save and Execute

1. Type Same Code using notepad.
2. click file then save
3. type filename as (first.html)
4. save on desktop
5. double click to execute

Example 1

```
<!DOCTYPE html>
```

```
<html>
```

```

<head>
  <title>Learn HTML From Sardar Azeem</title>
</head>
<body>
  <h1>HTML Program Structure</h1>
  <p>Learn HTML With Examples and 100% Practical</p>
  <p style="color: red">Welcome to Website Development Course</p>
</body> </html>

```

Example 2

HTML 4 Example Program	HTML 5 Example Program
<pre> <html> <!-- Defines languages of content: English --> <head> <!-- Information about website and creator --> <meta charset="UTF-8" /> <meta http-equiv="X-UA-Compatible" content="IE=edge" /> <!-- Defines the compatibility of version with browser --> <meta name="viewport" content="width=device-width, initial-scale=1.0" /> <!-- for make website responsive --> <meta name="author" content="SardarAzeem" /> <meta name="LinkedIn profile" content="WWW.linkedin.com/SardarAzeem" /> <!-- To give information about author or owner --> <meta name="description " content="This is An Educational institute PICT" /> <!-- to explain about website in few words --> <title>Learn From The Experts</title> <!-- Name of website or content to display --> </head> <body> <!-- Main content of website --> <h1>Hello World!!!</h1> <p>Welcome to Website Development </p> </body> </html> </pre>	<pre> <!DOCTYPE html> <!-- Defines types of documents : Html 5.0 --> <html lang="en"> <!-- Defines languages of content: English --> <head> <!-- Information about website and creator --> <meta charset="UTF-8" /> <meta http-equiv="X-UA-Compatible" content="IE=edge" /> <!-- Defines the compatibility of version with browser --> <meta name="viewport" content="width=device-width, initial-scale=1.0" /> <!-- for make website responsive --> <meta name="author" content="SardarAzeem" /> <meta name="LinkedIn profile" content="WWW.linkedin.com/SardarAzeem" /> <!-- To give information about author or owner --> <meta name="description " content="This is An Educational institute PICT" /> <!-- to explain about website in few words --> <title>Learn from The Experts</title> <!-- Name of website or content to display --> </head> <body> <!-- Main content of website --> <h1>Hello World!!!</h1> <p>Welcome to Website Development </p> </body> </html> </pre>

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

1. opening tag
2. content
3. closing tag.

Example Syntax		
	Azeem	
Tag Opening	Content of Tag	Tag Closing

But some HTML tags are unclosed tags. (e.g.
, <hr>)

All HTML tags must be enclosed within < > these brackets.

Every tag in HTML performs different tasks.

Syntax

<tag> content </tag>

Unclosed HTML Tags

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

 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>, , , <abbr>, <acronym>, <address>, <bdo>, <blockquote>, <cite>, <q>, <code>, <ins>, , <dfn>, <kbd>, <pre>, <samp>, <var> and

HTML Link Tags

<a> and <base>

HTML Image and Object Tags

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

HTML List Tags

, , , <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 no script

HTML Attribute

HTML attributes are special words which provide additional information about the elements or attributes are the modifier of the HTML element.

1. Each element or tag can have attributes, which defines the behavior of that element.
2. Attributes should always be applied with start tag.
3. The Attribute should always be applied with its name and value pair.
4. The Attributes name and values are case sensitive
5. You can add multiple attributes in one HTML element, but need to give space between two attributes.

Syntax

<element attribute name="value">content</element>

e.g. <body bgcolor="cyan"></body>

Attribute list

Attribute Name	Elements	Description
accept	<form>, <input>	List of types the server accepts, typically a file type.
accept-charset	<form>	List of supported charsets.
accesskey	Global attribute	Keyboard shortcut to activate or add focus to the element.
action	<form>	The URI of a program that processes the information submitted via the form.
align Deprec ated	<caption>, <col>, <colgroup>, <hr>, <iframe>, , <table>, <tbody>, <td>, <tfoot>, <th>, <thead>, <tr>	Specifies the horizontal alignment of the element.
allow	<iframe>	Specifies a feature-policy for the iframe.

alt	<area>, , <input>	Alternative text in case an image can't be displayed.
as	<link>	Specifies the type of content being loaded by the link.
async	<script>	Executes the script asynchronously.
autocapitalize	Global attribute	Sets whether input is automatically capitalized when entered by user
autocomplete	<form>, <input>, <select>, <textarea>	Indicates whether controls in this form can by default have their values automatically completed by the browser.
autoplay	<audio>, <video>	The audio or video should play as soon as possible.
background	<body>, <table>, <td>, <th>	Specifies the URL of an image file.
bgcolor	<body>, <col>, <colgroup>, <marquee>, <table>, <tbody>, <tfoot>, <td>, <th>, <tr>	Background color of the element.
border	, <object>, <table>	The border width.
capture	<input>	From the Media Capture specification, specifies a new file can be captured.
charset	<meta>	Declares the character encoding of the page or script.
checked	<input>	Indicates whether the element should be checked on page load.
cite	<blockquote>, , <ins>, <q>	Contains a URI which points to the source of the quote or change.
class	Global attribute	Often used with CSS to style elements with common properties.
color	, <hr>	This attribute sets the text color using either a named color or a color specified in the hexadecimal #RRGGBB format.
cols	<textarea>	Defines the number of columns in a textarea.
colspan	<td>, <th>	The colspan attribute defines the number of columns a cell should span.
content	<meta>	A value associated with http-equiv or name depending on the context.
contenteditable	Global attribute	Indicates whether the element's content is editable.
controls	<audio>, <video>	Indicates whether the browser should show playback controls to the user.
coords	<area>	A set of values specifying the coordinates of the hot-spot region.
crossorigin	<audio>, , <link>, <script>, <video>	How the element handles cross-origin requests
csp Experimental	<iframe>	Specifies the Content Security Policy that an embedded document must agree to enforce upon itself.
data	<object>	Specifies the URL of the resource.
data-*	Global attribute	Lets you attach custom attributes to an HTML element.
datetime	, <ins>, <time>	Indicates the date and time associated with the element.

decoding		Indicates the preferred method to decode the image.
default	<track>	Indicates that the track should be enabled unless the user's preferences indicate something different.
defer	<script>	Indicates that the script should be executed after the page has been parsed.
dir	Global attribute	Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
dirname	<input>, <textarea>	
disabled	<button>, <fieldset>, <input>, <optgroup>, <option>, <select>, <textarea>	Indicates whether the user can interact with the element.
download	<a>, <area>	Indicates that the hyperlink is to be used for downloading a resource.
draggable	Global attribute	Defines whether the element can be dragged.
enctype	<form>	Defines the content type of the form data when the method is POST.
enterkeyhint Experimental	<textarea>, contenteditable	The enterkeyhint specifies what action label (or icon) to present for the enter key on virtual keyboards. The attribute can be used with form controls (such as the value of textarea elements), or in elements in an editing host (e.g., using contenteditable attribute).
for	<label>, <output>	Describes elements which belongs to this one.
form	<button>, <fieldset>, <input>, <label>, <meter>, <object>, <output>, <progress>, <select>, <textarea>	Indicates the form that is the owner of the element.
formaction	<input>, <button>	Indicates the action of the element, overriding the action defined in the <form>.
formenctype	<button>, <input>	If the button/input is a submit button (e.g. type="submit"), this attribute sets the encoding type to use during form submission. If this attribute is specified, it overrides the enctype attribute of the button's form owner.
formmethod	<button>, <input>	If the button/input is a submit button (e.g. type="submit"), this attribute sets the submission method to use during form submission (GET, POST, etc.). If this attribute is specified, it overrides the method attribute of the button's form owner.
formnovalidate	<button>, <input>	If the button/input is a submit button (e.g. type="submit"), this boolean attribute specifies that the form is not to be validated when it is submitted. If this attribute is specified, it overrides the novalidate attribute of the button's form owner.
formtarget	<button>, <input>	If the button/input is a submit button (e.g. type="submit"), this attribute

		specifies the browsing context (for example, tab, window, or inline frame) in which to display the response that is received after submitting the form. If this attribute is specified, it overrides the target attribute of the button's form owner.
headers	<td>, <th>	IDs of the <th> elements which applies to this element.
height	<canvas>, <embed>, <iframe>, , <input>, <object>, <video>	Specifies the height of elements listed here. For all other elements, use the CSS height property. Note: In some instances, such as <div>, this is a legacy attribute, in which case the CSS height property should be used instead.
hidden	Global attribute	Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
high	<meter>	Indicates the lower bound of the upper range.
href	<a>, <area>, <base>, <link>	The URL of a linked resource.
hreflang	<a>, <link>	Specifies the language of the linked resource.
http-equiv	<meta>	Defines a pragma directive.
id	Global attribute	Often used with CSS to style a specific element. The value of this attribute must be unique.
integrity	<link>, <script>	Specifies a Subresource Integrity value that allows browsers to verify what they fetch.
intrinsicsize Deprecated		This attribute tells the browser to ignore the actual intrinsic size of the image and pretend it's the size specified in the attribute.
inputmode	<textarea>, contenteditable	Provides a hint as to the type of data that might be entered by the user while editing the element or its contents. The attribute can be used with form controls (such as the value of textarea elements), or in elements in an editing host (e.g., using contenteditable attribute).
ismap		Indicates that the image is part of a server-side image map.
itemprop	Global attribute	
kind	<track>	Specifies the kind of text track.
label	<optgroup>, <option>, <track>	Specifies a user-readable title of the element.
lang	Global attribute	Defines the language used in the element.
language Deprecated	<script>	Defines the script language used in the element.
loading Experimental	, <iframe>	Indicates if the element should be loaded lazily (loading="lazy") or loaded immediately (loading="eager").

list	<input>	Identifies a list of pre-defined options to suggest to the user.
loop	<audio>, <marquee>, <video>	Indicates whether the media should start playing from the start when it's finished.
low	<meter>	Indicates the upper bound of the lower range.
manifest Deprecated	<html>	Specifies the URL of the document's cache manifest. Note: This attribute is obsolete, use <link rel="manifest"> instead.
max	<input>, <meter>, <progress>	Indicates the maximum value allowed.
maxlength	<input>, <textarea>	Defines the maximum number of characters allowed in the element.
minlength	<input>, <textarea>	Defines the minimum number of characters allowed in the element.
media	<a>, <area>, <link>, <source>, <style>	Specifies a hint of the media for which the linked resource was designed.
method	<form>	Defines which HTTP method to use when submitting the form. Can be GET (default) or POST.
min	<input>, <meter>	Indicates the minimum value allowed.
multiple	<input>, <select>	Indicates whether multiple values can be entered in an input of the type email or file.
muted	<audio>, <video>	Indicates whether the audio will be initially silenced on page load.
name	<button>, <form>, <fieldset>, <iframe>, <input>, <object>, <output>, <select>, <textarea>, <map>, <meta>, <param>	Name of the element. For example used by the server to identify the fields in form submits.
novalidate	<form>	This attribute indicates that the form shouldn't be validated when submitted.
open	<details>, <dialog>	Indicates whether the contents are currently visible (in the case of a <details> element) or whether the dialog is active and can be interacted with (in the case of a <dialog> element).
optimum	<meter>	Indicates the optimal numeric value.
pattern	<input>	Defines a regular expression which the element's value will be validated against.
ping	<a>, <area>	The ping attribute specifies a space-separated list of URLs to be notified if a user follows the hyperlink.
placeholder	<input>, <textarea>	Provides a hint to the user of what can be entered in the field.
playsinline	<video>	A Boolean attribute indicating that the video is to be played "inline"; that is, within the element's playback area. Note that the absence of this attribute does not imply that the video will always be played in fullscreen.
poster	<video>	A URL indicating a poster frame to show until the user plays or seeks.
preload	<audio>, <video>	Indicates whether the whole resource, parts of it or nothing should be preloaded.

readonly	<input>, <textarea>	Indicates whether the element can be edited.
referrerpolicy	<a>, <area>, <iframe>, , <link>, <script>	Specifies which referrer is sent when fetching the resource.
rel	<a>, <area>, <link>	Specifies the relationship of the target object to the link object.
required	<input>, <select>, <textarea>	Indicates whether this element is required to fill out or not.
reversed		Indicates whether the list should be displayed in a descending order instead of an ascending order.
role	Global attribute	Defines an explicit role for an element for use by assistive technologies.
rows	<textarea>	Defines the number of rows in a text area.
rowspan	<td>, <th>	Defines the number of rows a table cell should span over.
sandbox	<iframe>	Stops a document loaded in an iframe from using certain features (such as submitting forms or opening new windows).
scope	<th>	Defines the cells that the header test (defined in the th element) relates to.
scoped Non-standard Deprecated	<style>	
selected	<option>	Defines a value which will be selected on page load.
shape	<a>, <area>	
size	<input>, <select>	Defines the width of the element (in pixels). If the element's type attribute is text or password then it's the number of characters.
sizes	<link>, , <source>	
slot	Global attribute	Assigns a slot in a shadow DOM shadow tree to an element.
span	<col>, <colgroup>	
spellcheck	Global attribute	Indicates whether spell checking is allowed for the element.
src	<audio>, <embed>, <iframe>, , <input>, <script>, <source>, <track>, <video>	The URL of the embeddable content.
srcdoc	<iframe>	
srclang	<track>	
srcset	, <source>	One or more responsive image candidates.
start		Defines the first number if other than 1.
step	<input>	
style	Global attribute	Defines CSS styles which will override styles previously set.

summary Deprecated	<table>	
tabindex	Global attribute	Overrides the browser's default tab order and follows the one specified instead.
target	<a>, <area>, <base>, <form>	Specifies where to open the linked document (in the case of an <a> element) or where to display the response received (in the case of a <form> element)
title	Global attribute	Text to be displayed in a tooltip when hovering over the element.
translate	Global attribute	Specify whether an element's attribute values and the values of its Text node children are to be translated when the page is localized, or whether to leave them unchanged.
type	<button>, <input>, <embed>, <object>, , <script>, <source>, <style>, <menu>, <link>	Defines the type of the element.
usemap	, <input>, <object>	
value	<button>, <data>, <input>, , <meter>, <option>, <progress>, <param>	Defines a default value which will be displayed in the element on page load.
width	<canvas>, <embed>, <iframe>, , <input>, <object>, <video>	For the elements listed here, this establishes the element's width. Note: For all other instances, such as <div>, this is a legacy attribute, in which case the CSS width property should be used instead.
wrap	<textarea>	Indicates whether the text should be wrapped.

Attributes Of Body TAG in HTML

HTML body tag Contains three attributes

bgcolor

Used to assign the background color to HTML webpage.

Syntax

Bgcolor="color name" or "HEX code"

text

Used to assign the text color to HTML webpage.

Syntax

text="color name" or "HEX code"

background

Used to set a picture or a wallpaper to the background color to HTML webpage.

Syntax

Background="URL"(universal resource locator)

Example 1(color with name)	Example 2(color with HEX code)
<pre><!DOCTYPE html> <html> <head><title>pict.com</title></head> <body bgcolor="cyan" text="green"> <h1>Pict Computer College Abbottabad</h1> <p>Welcome to website development Course</p> </body> </html></pre>	<pre><!DOCTYPE html> <html> <head><title>pict.com</title></head> <body bgcolor="#B1FC2D " text="#BD12B8 "> <h1>Pict Computer College Abbottabad</h1> <p>Welcome to website development Course</p> </body> </html></pre>
Example 3(setting an image background)	
<pre><!DOCTYPE html> <html> <head><title>pict.com</title></head> <body text="#BD12B8 " background="E:\Wallpapers\Sardar (2).png"> <h1>Pict Computer College Abbottabad</h1> <p>Welcome to website development Course</p> </body> </html></pre>	

HTML FONT tag and its attributes

HTML FONT tag is used to assign font color,size and font family to text written on a webpage. FONT tag was an essential tag in HTML 4 but Now It has been excluded in HTML 5.but still can be used.

Syntax

` any text `

Attributes in font tag

Size

Used to assign font size to given text.

Syntax size= 1 to 7

Color

Used to apply any color to given text.

Syntax color="Color name" or "HEX Code"

Face

Used to apply any font family to given text

Syntax face = "Style Name"

Other Relevant TAGs for Font Formatting

Element name/TAG name	Description
 Any Text	This is a physical tag, which is used to bold the text written between it.
 Any Text	This is a logical tag, which tells the browser that the text is important.
<i> Any Text</i>	This is a physical tag which is used to make text italic.
 Any Text	This is a logical tag which is used to display content in italic.
<mark> Any Text</mark>	This tag is used to highlight text.
<u> Any Text</u>	This tag is used to underline text written between it.
<tt> Any Text</tt>	This tag is used to appear a text in teletype. (not supported in HTML5)
<strike> Any Text</strike>	This tag is used to draw a strikethrough on a section of text. (Not supported in HTML5)
^{Any Text}	It displays the content slightly above the normal line.
_{Any Text}	It displays the content slightly below the normal line.
 Any Text	This tag is used to display the deleted content.
<ins> Any Text</ins>	This tag displays the content which is added
<big> Any Text</big>	This tag is used to increase the font size by one conventional unit.
<small>Any Text</small>	This tag is used to decrease the font size by one unit from base font size.
<center>Any TEXT</center>	To Center align TEXT.

Practical Demonstration Of All Of the above By Example

```
<!DOCTYPE html>
<html>
<head><title>pict.com</title></head>
<body>
<font size=7>Pict Computer College Yousaf Jamal Plaza Abbottabad</font><br><hr>
<font size=7 color="blue">Pict Computer College Yousaf Jamal Plaza Abbottabad</font><br><hr>
<font size=7 color=green face="ToySans">Pict Computer College Yousaf Jamal Plaza
Abbottabad</font><br><hr>
<font size=7>
```

```
<b>Pict Abbottabad</b><br><hr>
<strong>Pict Abbottabad</strong><br><hr>
<i>Pict Abbottabad</i><br><hr>
<em>Pict Abbottabad</em><br><hr>
<u>Pict Abbottabad</u><br><hr>
<del>Pict Abbottabad</del><br><hr>
<center>Pict Abbottabad</center><br><hr>
<mark>Pict Abbottabad</mark><br><hr>
<big>Pict Abbottabad</big><br><hr>
<small>Pict Abbottabad</small><br><hr>
H<sub>2</sub>O<br><hr>
10<sup>50</sup><br><hr>
</font><br><hr>
</body>
```

HTML Heading

A HTML heading or HTML h tag can be defined as a title or a subtitle which you want to display on the webpage. When you place the text within the heading tags `<h1>.....</h1>`, it is displayed on the browser in the bold format and size of the text depends on the number of headings.

There are six different HTML headings which are defined with the `<h1>` to `<h6>` tags, from highest level h1 (main heading) to the least level h6 (least important heading).

h1 is the largest heading tag and h6 is the smallest one. So h1 is used for most important heading and h6 is used for least important.

Syntax

```
<h1>Sardar Azeem PICT Abbottabad</h1>
<h2> Sardar Azeem PICT Abbottabad </h2>
<h3> Sardar Azeem PICT Abbottabad </h3>
<h4> Sardar Azeem PICT Abbottabad </h4>
<h5> Sardar Azeem PICT Abbottabad </h5>
<h6> Sardar Azeem PICT Abbottabad </h6>
```

Sub tags/Attributes

1. Style

We can apply style attribute to any heading to apply font size/color/style/alignment

Syntax

<H1 Style=" property1: value1; property2: value2;Property N: value N">

Example 1(Without Attributes)	Example 2(With Attributes)
<pre><!DOCTYPE html> <html> <body> <h1>SardarAzeem Pict Abbottabad KPK Pak</h1> <h2> SardarAzeem Pict Abbottabad KPK Pak </h2> <h3> SardarAzeem Pict Abbottabad KPK Pak </h3> <h4> SardarAzeem Pict Abbottabad KPK Pak </h4> <h5> SardarAzeem Pict Abbottabad KPK Pak </h5> <h6> SardarAzeem Pict Abbottabad KPK Pak </h6> </body> </html></pre>	<pre><!DOCTYPE html> <html> <body> <h1 title="This is heading tag"> SardarAzeem Pict Abbottabad KPK Pak </h1> <h1 style="height: 50px; color: blue"> SardarAzeem Pict Abbottabad KPK Pak </h1> <h1 style="font-family: Verdana; color: blue;font- size:50px"> SardarAzeem Pict Abbottabad KPK Pak </h1> </body> </html></pre>

HTML Paragraph

HTML paragraph or HTML p tag is used to define a paragraph in a webpage. An HTML <p> tag indicates starting of new paragraph. </p> indicates closing of paragraph. HTML paragraph is a block element. (i.e. every paragraph starts from new line)

Sub tags/Attributes

1. Style

We can apply style attribute to any heading to apply font size/color/style/alignment

Syntax

<P Style="property1: value1;property2: value2;Property N: value N">

Example 1(Without Attributes)	Example 2(With Attributes)
<pre><!DOCTYPE html> <html> <body> <p>This is first paragraph.</p> <p>This is second paragraph.</p> <p>This is third paragraph.</p> </body> </html></pre>	<pre><!DOCTYPE html> <html> <head> </head> <body> <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></pre>

HTML pre tag

The HTML <pre> tag is used to specify pre-formatted texts. Texts within <pre>.....</pre> tag is displayed in a fixed-width font. Usually, it is displayed in Courier font. It maintains both space and line break.

Example

```
<!DOCTYPE>
<html>
<body>
<pre>
Sardar Azeem
    PICT Computer College
        Abbottabad KPK Pakistan
            03135879331
</pre>
</body>
</html>
```

HTML Marquee/Anchor/Image Tags

HTML Marquee Tag

The Marquee HTML tag is a non-standard HTML element which is used to scroll a image or text horizontally or vertically.

In simple words, you can say that it scrolls the image or text up, down, left or right automatically.

Marquee tag was first introduced in early versions of Microsoft's Internet Explorer. It is compared with Netscape's blink element.

Marquee tag has the following attributes.

Attribute	Description
behavior	It facilitates user to set the behavior of the marquee to one of the three different types: scroll, slide and alternate.
direction	defines direction for scrolling content. It may be left, right, up and down.
width	defines width of marquee in pixels or %.
height	defines height of marquee in pixels or %.
hspace	defines horizontal space in pixels around the marquee.
vspace	defines vertical space in pixels around the marquee.
scrolldelay	defines scroll delay in seconds.
scrollamount	defines scroll amount in number.
loop	defines loop for marquee content in number.
bgcolor	defines background color. It is now deprecated.

HTML Image TAG

HTML img tag is used to display image on the web page. HTML img tag is an empty tag that contains attributes only, closing tags are not used in HTML image element.

Attributes of HTML img tag

The src and alt are important attributes of HTML img tag. All attributes of HTML image tag are given below.

1) src="URL"

It is a necessary attribute that describes the source or path of the image. It instructs the browser where to look for the image on the server.

The location of image may be on the same directory or another server.

2) alt

The alt attribute defines an alternate text for the image, if it can't be displayed. The value of the alt attribute describe the image in words. The alt attribute is considered good for SEO prospective.

3) width=1 to 1000 px

It is an optional attribute which is used to specify the width to display the image. It is not recommended now. You should apply CSS in place of width attribute.

4) height=1 to 1000px

Used to add height of images. It is not recommended now. You should apply CSS in place of height attribute.

5) Border=1 to 10px

Used to apply border to images

HTML Anchor TAG

The HTML anchor tag defines a hyperlink that links one page to another page. It can create hyperlink to other web page as well as files, location, or any URL. The "href" attribute is the most important attribute of the HTML a tag. and which links to destination page or URL.

href attribute of HTML anchor tag

The href attribute is used to define the address of the file to be linked. In other words, it points out the destination page.

The syntax of HTML anchor tag

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

Example Of Using Marquee/Image/Anchor Tags

```
<html>
<body>
<marquee bgcolor="black" direction="left" behaviour="scroll" delay=300>
<font size=7 color="white" >Pict Computer College Yousaf Jamal Plaza Abbottabad</font>

</marquee><br><hr>

<marquee bgcolor="red" direction="right" behaviour="slide" delay=300>
<font size=7 color="yellow" face="Stencil" >Pict Computer College Yousaf Jamal Plaza
Abbottabad</font>

</marquee>

<marquee bgcolor="yellow" direction="up" behaviour="alternate" delay=300 width=300px height=300px>
<font size=4 color="blue" >What is Lorem Ipsum?
Our mission is to eliminate social abuses, unemployment and poverty through the technical education.Using the
modern and sophisticated sources of IT, the institute is busy in providing all the possible resources of technical
education and developing technical skills in our students</font></marquee><br><hr>
<marquee bgcolor="yellow" direction="down" behaviour="alternate" delay=300 width=300px
height=300px>
<font size=4 color="blue" >What is Lorem Ipsum?
To make sure that the professional computer training must be provided with all available resources of modern
technology, because the city of Abbottabad is known by the people as the CITY OF SCHOOLS & COLLEGES.
The technical education is assured to be provided with 100 per cent of the result orientation.
So as to make the standard of living of each & every citizen with a positive approach
</font> </marquee><br><hr>

</img>
<br><hr>

<marquee>
</img>
</img>
</img>
</img>
</marquee>
<a href="home.html">Home Page</a>
<a href="https://pictacademy.com">Home Page</a>
<a href="https://www.youtube.com/channel/UCZP2P4t10Q3ZhiEihDBcaSg">Home Page</a>
</body></html>
```

HTML List TAGS(OL/UL/MENU/DL/SELECT)

HTML Lists are used to specify lists of information. All lists may contain one or more list elements. There are Following types of HTML lists:

- 1) Ordered List or Numbered List (ol)
- 2) Unordered List or Bulleted List (ul)

- 3) Menu List
- 4) Description List or Definition List (dl)
- 5) Select List

HTML Ordered List | HTML Numbered List

HTML Ordered List or Numbered List displays elements in numbered format. The HTML ol tag is used for ordered list. There can be different types of numbered list:

Type	Description
Type "1"	This is the default type. In this type, the list items are numbered with numbers.
Type "I"	In this type, the list items are numbered with upper case roman numbers.
Type "i"	In this type, the list items are numbered with lower case roman numbers.
Type "A"	In this type, the list items are numbered with upper case letters.
Type "a"	In this type, the list items are numbered with lower case letters.

HTML Unordered List | HTML Bulleted List

HTML Unordered List or Bulleted List displays elements in bulleted format . We can use unordered list where we do not need to display items in any particular order. The HTML ul tag is used for the unordered list. There can be 4 types of bulleted list:

Type	Description
"disc"/"fillround"	This is the default style. In this style, the list items are marked with bullets.
"circle"	In this style, the list items are marked with circles.
"square"	In this style, the list items are marked with squares.
"none"	In this style, the list items are not marked.

HTML <menu> tag

HTML <menu> tag specifies a list or menu of commands that a user can perform or activate. It is used for creating context menu as well as lists menu.

A <menu> element can contain one or more or <menuitem> elements within it.

Syntax

<menu>.....</menu>

HTML Description List | HTML Definition List

HTML Description List or Definition List displays elements in definition form like in dictionary. The <dl>, <dt> and <dd> tags are used to define description list.

The 3 HTML description list tags are given below:

<dl> tag defines the description list.

<dt> tag defines data term.

<dd> tag defines data definition (description).

Select Tag in HTML

In HTML, with the help of a select tag, we can create the dropdown list while creating the form where the user can choose one of the options from the given option. If the developer allows the user, then the user also selects multiple options from the given option. We can implement the select tag only inside the HTML form. We can preserve the space by choosing the dropdown menu while accepting the information from the users.

Syntax

<select>.....</select>

Example 1	Example 2
<pre><!DOCTYPE html> <html> <body> <ol type=1> Saturday Sunday Monday Tuesday Wednesday Thursday Friday
<hr> <ol type=A> Saturday Sunday Monday Tuesday Wednesday Thursday Friday
<hr> <ol type=a> Saturday</pre>	<pre><!DOCTYPE html> <html> <head> <title>T-Shirt Color Selection</title> <style> body { font-family: Arial, sans-serif; background-color: #f2f2f2; } label { font-size: 18px; font-weight: bold; color: #333; } select { font-size: 16px; padding: 8px; margin: 5px 0; border: 1px solid #ccc; border-radius: 5px; width: 200px;</pre>

<pre> Sunday Monday Tuesday Wednesday Thursday Friday
<hr> <ol type=l> Saturday Sunday Monday Tuesday Wednesday Thursday Friday
<hr> <ol type=i> Saturday Sunday Monday Tuesday Wednesday Thursday Friday
<hr> <ul type="disk"> DIT CIT Office Automation Computer Hardware Computer Networking Website Development E-Commerce MGMT
<hr> <ul type="circle"> DIT CIT Office Automation Computer Hardware Computer Networking Website Development E-Commerce MGMT
<hr> <ul type="square"> DIT CIT Office Automation Computer Hardware Computer Networking Website Development E-Commerce MGMT
<hr> <menu </pre>	<pre> background-color: #fff, color: #555; } option { color: #555; } </style> </head> <body> <label for="shirt-colors">What color t- shirt would you like?</label>
 <select name="shirts" id="shirt-colors"> <option value="red">Red t-shirt</option> <option value="blue">Blue t-shirt</option> <option value="orange">Orange t-shirt</option> </select>
<hr> <dl> <dt>HTML</dt> <dd>is a markup language</dd> <dt>Java</dt> <dd>is a programming language and platform</dd> <dt>JavaScript</dt> <dd>is a scripting language</dd> <dt>SQL</dt> <dd>is a query language</dd> </dl> </body> </html> </pre>
---	---

<pre> DIT BTE TTB SDC CIT 6 Months 4 Months 8 Months Office Automation Basic Intermediate Advanced Computer Hardware Computer Networking Website Development E-Commerce MGMT </menu>
<hr> </body> </html> </pre>	
--	--

HTML Table TAG

HTML table tag is used to display data in tabular form (row * column). There can be many columns in a row.

We can create a table to display data in tabular form, using <table> element, with the help of <tr> , <td>, and <th> elements.

In Each table, table row is defined by <tr> tag, table header is defined by <th>, and table data is defined by <td> tags.

HTML tables are used to manage the layout of the page e.g. header section, navigation bar, body content, footer section etc. But it is recommended to use div tag over table to manage the layout of the page.

HTML Table Tags

Tag	Description
<table></table>	It defines a table.
<tr></tr>	It defines a row in a table.
<th></th>	It defines a header cell in a table.
<td></td>	It defines a cell in a table.
<caption></caption>	It defines the table caption.
<colgroup>	It specifies a group of one or more columns in a table for formatting.
<col>	It is used with <colgroup> element to specify column properties for each column.
<tbody>	It is used to group the body content in a table.
<thead>	It is used to group the header content in a table.
<tfooter>	It is used to group the footer content in a table.

Example 1	Example 2
<pre> <!DOCTYPE> <html> <head> <style> table, th, td { border: 1px solid black; border-collapse: collapse; } th, td { padding: 10px; } table#alter tr:nth-child(even) { background-color: #eee; } table#alter tr:nth-child(odd) { background-color: #fff; } table#alter th { color: white; background-color: gray; } </style> </head> <body> <table id="alter"> <tr> <th>First_Name</th> <th>Last_Name</th> <th>Marks</th> </tr> <tr> <td>Imran</td> <td>Noman</td> </pre>	<pre> <!DOCTYPE html> <html lang="en"> <head> <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="3" height=400px style="padding:10px 20px; background-color#3D43E2;"> <h1 style="font-size:24px;">My Slider Here</h1> </td> </tr> <tr> <td colspan="3" height=50px style="padding:10px 20px; background-color:#acb3b9;"> <h1 style="font-size:24px;">Navigation Menu Here</h1> </td> </tr> <tr style="height:700px;"> <td style="width:20%; padding:20px; background-color:#d4d7dc; vertical-align: top;"> <ul style="list-style:none; padding:0px; line-height:24px;"> Home About Contact US </td> <td style="padding:20px; background-color:#f2f2f2; vertical-align:top;"> <h2>Welcome to our site</h2> </pre>

<pre> <td>95</td> </tr> <tr> <td>Zia</td> <td>Zafar</td> <td>80</td> </tr> <tr> <td>Maryam</td> <td>Javed</td> <td>82</td> </tr> <tr> <td>Nazia</td> <td>Shaheen</td> <td>72</td> </tr> </table> </body></html> </pre>	<pre> <p>Here you will learn how to create websites...</p> </td> <td style="width:20%; padding:20px; background- color:#d4d7dc; vertical-align: top;"> Ads Here </td> </tr> <tr> <td colspan="3" height=400px style="padding:10px 20px; background-color#3D43E2;"> <h1 style="font-size:24px;">Footer Section Here</h1> </td> </tr> <tr> <td colspan="2" style="padding:5px; background-color:#acb3b9; text-align:center;"> <p>copyright &copy; sardarazeem</p> </td> </tr> </table> </body> </html> </pre>
--	---

HTML Form

An HTML form is a section of a document which contains controls such as text fields, password fields, checkboxes, radio buttons, submit button, menus etc.

An HTML form facilitates the user to enter data that is to be sent to the server for processing such as name, email address, password, phone number, etc. .

Why use HTML Form

HTML forms are required if you want to collect some data from of the site visitor.

HTML Form Tags

Tag	Description
<form></form>	It defines an HTML form to enter inputs by the used side.
<input>	It defines an input control.
<textarea>	It defines a multi-line input control.
<label>	It defines a label for an input element.
<fieldset>	It groups the related element in a form.
<legend>	It defines a caption for a <fieldset> element.
<select>	It defines a drop-down list.
<optgroup>	It defines a group of related options in a drop-down list.
<option>	It defines an option in a drop-down list.
<button>	It defines a clickable button.

HTML 5 Form Tags

Tag	Description
<datalist>	It specifies a list of pre-defined options for input control.

<keygen>	It defines a key-pair generator field for forms.
<output>	It defines the result of a calculation.

HTML Form Input Types

In HTML <input type=" " > is an important element of HTML form. The "type" attribute of input element can be various types, which defines information field. Such as <input type="text" name="name"> gives a text box.

type=" " "	Description
text	Defines a one-line text input field
password	Defines a one-line password input field
submit	Defines a submit button to submit the form to server
reset	Defines a reset button to reset all values in the form.
radio	Defines a radio button which allows select one option.
checkbox	Defines checkboxes which allow select multiple options form.
button	Defines a simple push button, which can be programmed to perform a task on an event.
file	Defines to select the file from device storage.
image	Defines a graphical submit button.

HTML5 added new types on <input> element. Following is the list of types of elements of HTML5

type=" " "	Description
color	Defines an input field with a specific color.
date	Defines an input field for selection of date.
datetime-local	Defines an input field for entering a date without time zone.
email	Defines an input field for entering an email address.
month	Defines a control with month and year, without time zone.
number	Defines an input field to enter a number.
url	Defines a field for entering URL
week	Defines a field to enter the date with week-year, without time zone.
search	Defines a single line text field for entering a search string.
tel	Defines an input field for entering the telephone number.

HTML <form> element attributes

In HTML there are various attributes available for <form> element which are given below:

HTML action attribute

The action attribute of <form> element defines the process to be performed on form when form is submitted, or it is a URI to process the form information.

The action attribute value defines the web page where information proceed. It can be .php, .jsp, .asp, etc. or any URL where you want to process your form.

Example

```
<!DOCTYPE html>
<html>
  <body>
    <h2>Demo of action attribute of form element</h2>
    <form action="action.html" method="post">
      <label>User Name:</label><br>
      <input type="text" name="name"><br><br>
      <label>User Password</label><br>
      <input type="password" name="pass"><br><br>
      <input type="submit">
    </form>
    <p><b>It will redirect to a new page "action.html" when you click on submit button</b></p>
  </body>
</html>
```

HTML method attribute

The method attribute defines the HTTP method which browser used to submit the form. The possible values of method attribute can be:

post:

We can use the post value of method attribute when we want to process the sensitive data as it does not display the submitted data in URL.

Example:

```
<form action="action.html" method="post">
```

get:

The get value of method attribute is default value while submitting the form. But this is not secure as it displays data in URL after submitting the form.

Example:

```
<form action="action.html" method="get">
```

HTML autocomplete attribute

The HTML autocomplete attribute is a newly added attribute of HTML5 which enables an input field to complete automatically. It can have two values "on" and "off" which enables autocomplete either ON or OFF. The default value of autocomplete attribute is "on".

Example:

```
<form action="action.html" method="get" autocomplete="on">
```

Example:

```
<form action="action.html" method="get" autocomplete="off">
```

HTML enctype attribute

The HTML enctype attribute defines the encoding type of form-content while submitting the form to the server. The possible values of enctype can be:

application/x-www-form-urlencoded: It is default encoding type if the enctype attribute is not included in the form. All characters are encoded before submitting the form.

Example:

```
<form action="action.html" method="post" enctype="application/x-www-form-urlencoded" >
```

multipart/form-data: It does not encode any character. It is used when our form contains file-upload controls.

Example:

```
<form action="action.html" method="post" enctype="multipart/form-data">
```

HTML novalidate attribute HTML5

The novalidate attribute is newly added Boolean attribute of HTML5. If we apply this attribute in form then it does not perform any type of validation and submit the form.

Example:

```
<form action = "action.html" method = "get" novalidate>
```

HTML name attribute

The HTML name attribute defines the name of an input element. The name and value attribute are included in HTTP request when we submit the form.

Example

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
    <h2>Fill the form</h2>
```

```
    <form action = "action.html" method = "get">
```

```
        Enter name:<br><input type="name" name="uname"><br>
```

```
        Enter age:<br><input type="number" name="age"><br>
```

```
        Enter email:<br><input type="email"><br>
```

```
        <input type="submit" value="Submit">
```

```
    </form>
```

<p>Note: If you will not use name attribute in any input field, then that input field will not be submitted, when submit the form.</p>

<p>Click on submit and see the URL where email is not included in HTTP request as we have not used name attribute in the email input field</p>

```
    </body>
```

```
</html>
```

HTML value attribute

The HTML value attribute defines the initial value or default value of an input field.

Example

```
<!DOCTYPE html>
```

```
<html>
```

```
    <body>
```

```
        <h2>Fill the form</h2>
```

```
        <form>
```

```
            <label>Enter your Name</label><br>
```

```
            <input type="text" name="uname" value="Enter Name"><br><br>
```

```
            <label>Enter your Email-address</label><br>
```

```
            <input type="text" name="uname" value="Enter email"><br><br>
```

```
            <label>Enter your password</label><br>
```



```
<input type="password" name="pass" value=""><br><br>
<input type="submit" value="login">
</form>
<p><b>Note: In password input field the value attribute will always be empty</b></p>
</body>
</html>
```

HTML required attribute HTML5

HTML required is a Boolean attribute which specifies that user must fill that field before submitting the form.

Example

```
<!DOCTYPE html>
<html>
<body>
  <h2>Fill the form</h2>
  <form>
    <label>Enter your Email-address</label><br>
    <input type="text" name="uname" required><br><br>
    <label>Enter your password</label><br>
    <input type="password" name="pass"><br><br>
    <input type="submit" value="login">
  </form>
  <p><b> If you will try to submit the form without completing email field then it will give an error pop up.</b></p>
</body>
</html>
```

HTML autofocus attribute HTML5

The autofocus is a Boolean attribute which enables a field automatically focused when a webpage loads.

Example:

```
<form>
  <label>Enter your Email-address</label><br>
  <input type="text" name="uname" autofocus><br><br>
  <label>Enter your password</label><br>
  <input type="password" name="pass"><br><br>
  <input type="submit" value="login">
</form>
```

HTML placeholder attribute HTML5

The placeholder attribute specifies a text within an input field which informs the user about the expected input of that field.

The placeholder attribute can be used with text, password, email, and URL values.

Example

```
<form>
  <label>Enter your name</label><br>
  <input type="text" name="uname" placeholder="Your name"><br><br>
  <label>Enter your Email address</label><br>
  <input type="email" name="email" placeholder="example@gmail.com"><br><br>
  <label>Enter your password</label><br>
  <input type="password" name="pass" placeholder="your password"><br><br>
  <input type="submit" value="login">
</form>
```

HTML disabled attribute

The HTML disabled attribute when applied then it disable that input field. The disabled field does not allow the user to interact with that field.

The disabled input field does not receive click events, and these input value will not be sent to the server when submitting the form.

Example:

```
<input type="text" name="uname" disabled><br><br>
```

HTML size attribute

The size attribute controls the size of the input field in typed characters.

Example:

```
<label>Account holder name</label><br>  
    <input type="text" name="uname" size="40" required><br><br>  
    <label>Account number</label><br>  
    <input type="text" name="an" size="30" required><br><br>  
    <label>CVV</label><br>  
    <input type="text" name="cvv" size="1" required><br><br>
```

HTML form attribute

HTML form attribute allows a user to specify an input field outside the form but remains the part of the parent form.

Example

```
<!DOCTYPE html>  
  
<html>  
  
    <body>  
        <form id="fcontrol">  
            User Name:<br><input type="text" name="uname"><br><br>  
            User password:<br><input type="password" name="pass"><br><br>  
        </form>  
        <p>The email field is outside the form but still it will remain part of the form</p>  
        User email: <br><input type="email" name="email" form="fcontrol" required><br>  
        <input type="submit" form="fcontrol">  
    </body>  
</html>
```

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>
```

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.javatpoint.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.javatpoint.com/" height="50%" width="70%"></iframe>
```

```
</body>
```

```
</html>
```

CSS to set the height and width of the iframe.

```
<!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.javatpoint.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.

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<h2>Remove the Iframe Border</h2>
```

<p>This iframe example doesn't have any border</p>

```
<iframe src="https://www.javatpoint.com/" style="border:none;"></iframe>
```

```
</body>
```

```
</html>
```

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

Example:

```
<!DOCTYPE html>
```

```
<html>
<body>
<h2>Custom Iframe Border</h2>
<iframe src="https://www.javatpoint.com/" style="border:2px solid tomato;"></iframe>
</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.javatpoint.com" target="iframe_a">JavaTpoint.com</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>
```

new.html output code:

```
<!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:

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

Example:

```
<iframe width="550" height="315" src="https://www.youtube.com/embed/JHq3pL4cdy4"
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/O5hShUO
6wxs" frameborder="0" allow="accelerometer; autoplay; encrypted-
media; gyroscope; picture-in-picture" style="padding:20px;">></iframe>
```

HTML Comments

Comments are some text or code written in your code to give an explanation about the code, and not visible to the user. Comments which are used for HTML file are known as HTML comments. Anything written between these tags will be ignored by the browser, so comments will not be visible on the webpage.

- 1) Comments of any code make code easy to understand and increase readability of code.
- 2) Comments are also part of the code, which gives an explanation of the code.

How to add comment In HTML

Syntax

```
<!-- Write commented text here -->  
<!-- <p>There is some text</p>  
      <p>There is second text</p> -->
```

Example:

```
<!DOCTYPE html>  
<html>  
<!-- This is Header section -->  
<head>  
  <!-- Internal CSS -->  
  <style>  
    body{  
      text-align: center;  
      background-color: #f0f8ff;  
      font-size: 30px;  
      color: red;  
    }  
  </style>  
</head>  
  
<!-- This is body section, write code here which you want to display on web-page -->  
<body>  
  <!-- heading tag -->  
  <h2>First WebPage</h2>  
  
  <!-- Paragraph tag -->
```



```
<p>Write your Content here!!!</p>
</body>
</html>
```

Multiline Comment

In HTML code, we can also comments multiple lines at a time. In multiline comment we can use any description about code or multiple line code to debug, etc.

Syntax

```
<!--
```

Your code is commented.

Write description of code.

It will not display at webpage.

```
-->
```

Example:

```
<h2>Cake Gallery</h2>
```

```
<!-- This is image for a yummy cake
```

you can see it on your web-page

of your favorite browser -->

```

```

HTML <frameset> tag (Not supported in HTML5)

HTML <frameset> tag is used to contain the group of frames which can be controlled and styled as a unit. The <frameset> element also specifies the number of rows and columns in the frameset, and how much space they will occupy in a frame.

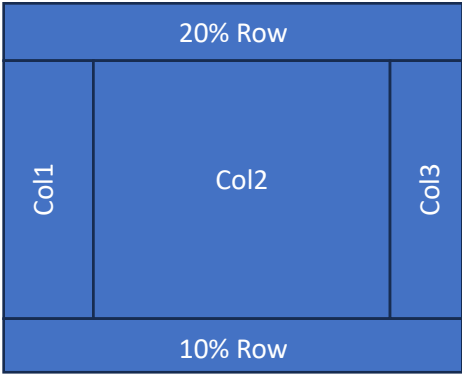
Syntax

```
<frameset cols=" ">.....</frameset>
```

Display	Block
Start tag/End tag	Both Start and End Tag

Usage	Frames
-------	--------

<p>Example 1</p> <pre><html> <Frameset rows="20%,*"> <frame src="URL"> <frame src="URL"> </frameset> </html></pre>	
<p>Example 2</p> <pre><html> <Frameset rows="20%,*,20%"> <frame src="URL"> <frame src="URL"> <frame src="URL"> </frameset> </html></pre>	
<p>Example 3</p> <pre><html> <Frameset cols="20%,*"> <frame src="URL"> <frame src="URL"> </frameset> </html></pre>	
<p>Example 4</p> <pre><html> <Frameset cols="20%,*,20%"> <frame src="URL"> <frame src="URL"> <frame src="URL"> </frameset> </html></pre>	

<p>Example 5</p> <pre> <html> <Frameset rows="20%,*,10%"> <frame src="URL"> <frameset cols="20%,*,20%"> <frame src="URL"> <frame src="URL"> </frameset> <frame src="URL" </frameset> </html> </pre>	
<p>Example 6</p> <pre> <html> <Frameset rows="20%,*,10%"> <frame src="navigator.html"> <frameset cols="20%,*,20%"> <frame src="verticalMenu.html"> <frame src="content.html"> <frame src="ads.html"> </frameset> <frame src="copyright.html"> </frameset> </html> </pre>	<p>Same As Above</p>

HTML Layouts

HTML layouts provide a way to arrange web pages in well-mannered, well-structured, and in responsive form or we can say that HTML layout specifies a way in which the web pages can be arranged. Web-page layout works with arrangement of visual elements of an HTML document.

Web page layout is the most important part to keep in mind while creating a website so that our website can appear professional with the great look. You can also use CSS and JAVASCRIPT based frameworks for creating layouts for responsive and dynamic website designing.

Every website has a specific layout to display content in a specific manner.

Following are different HTML5 elements which are used to define the different parts of a webpage.

- <header>: It is used to define a header for a document or a section.
- <nav>: It is used to define a container for navigation links
- <section>: It is used to define a section in a document

- <article>: It is used to define an independent self-contained article
- <aside>: It is used to define content aside from the content (like a sidebar)
- <footer>: It is used to define a footer for a document or a section
- <details>: It is used to define additional details
- <summary>: It is used to define a heading for the <details> element

HTML <header>

The <header> element is used to create header section of web pages. The header contains the introductory content, heading element, logo or icon for the webpage, and authorship information.

```
<!DOCTYPE html>
<html>
  <head>
    <title>First Webpage</title>
  </head>
  <body>
    <header style="background-color: #303030; height: 80px;width: 100%">
      <h1 style="font-size: 30px; color: white;text-align: center; padding-top: 15px;">Welcome to MyFirstWebpage</h1>
    </header>
  </body>
</html>
```

HTML <nav>

The <nav> elements is a container for the main block of navigation links. It can contain links for the same page or for other pages.

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      li{ display: inline-block;
          padding: 10px}
    </style>
  </head>
```

```

<body>
  <nav style="background-color:#bcdeef;">
    <h1 style="text-align: center;">Navigation Links</h1>
    <ul>
      <li><a href="#">link1</a></li>
      <li><a href="#">link2</a></li>
      <li><a href="#">link3</a></li>
      <li><a href="#">link4</a></li>
    </ul>
  </nav>
</body>
</html>

```

HTML <section>

HTML <section> elements represent a separate section of a web page which contains related element grouped together. It can contain: text, images, tables, videos, etc.

```

<!DOCTYPE html>
<html>
<head>
  <title>Page Section</title>
</head>
<body>
  <section style="background-color:#ff7f50; width: 100%; border: 1px solid black;">
    <h2>Introduction to HTML</h2>
    <p>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.</p>
  </section>
</body>
</html>

```

HTML <article>

The HTML tag is used to contain a self-contained article such as big story, huge article.

```

<!DOCTYPE html>
<html>
<head>
  <title>Article Example</title>
</head>
<body>
  <article style="width: 100%; border:2px solid black; background-color: #fff0f5;">
    <h2>History of Computer</h2>
    <p>Write your content here for the history of computer</p>
  </article>

```

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

HTML <aside>

HTML <aside> define aside content related to primary content. The <aside> content must be related to the primary content. It can function as side bar for the main content of web page.

```
<!DOCTYPE html>  
<html>  
<head>  
  <title>Aside Example</title>  
</head>  
<body>  
  <aside style="background-color:#e6e6fa">  
    <h2>Sidebar information</h2>  
    <p>This conatins information which will represent like a side bar for a  
webpage</p>  
  </aside>  
</body>  
</html>
```

HTML <footer>

HTML <footer> element defines the footer for that document or web page. It mostly contains information about author, copyright, other links, etc.

```
<!DOCTYPE html>  
<html>  
<head>  
  <title>Footer Section</title>  
</head>  
<body>  
  <footer style="background-color:#f0f8ff; width: 100%; text-align: center;">  
    <h3>Footer Example</h3>  
    <p>© Copyright 2018-2020. </p>  
  </footer>  
</body>  
</html>
```

HTML <details>

HTML <details> element is used to add extra details about the web page and use can hide or show the details as per requirement.

```

<!DOCTYPE html>
<html>
<head>
  <title>Deatils element</title>
</head>
<body>
  <details style="background-color: #f5deb3">
    <summary>This is visible section: click to show other details</summary>
    <p>This section only shows if user want to see it. </p>
  </details>
</body>
</html>

```

HTML <summary>

HTML <summary> element is used with the <details> element in a web page. It is used as summary, captions about the content of <details> element.

```

<!DOCTYPE html>
<html>
<head>
  <title>Summary Example</title>
</head>
<body>
  <details>
    <summary>HTML is acronym for?</summary>
    <p style="color: blue; font-size: 20px;">Hypertext Markup Language</p>
  </details>
</body>
</html>

```

Full Webpage Layout

```

<!DOCTYPE html>
<html>

<head>
  <title>Web Page Layout</title>
  <style>
    .head1 {
      font-size: 40px;
      color: #009900;
      font-weight: bold;
    }

    .head2 {
      font-size: 17px;

```

```
margin-left: 10px;
margin-bottom: 15px;
}

body {
margin: 0 auto;
background-position: center;
background-size: contain;
}

.menu {
position: sticky;
top: 0;
background-color: #009900;
padding: 10px 0px 10px 0px;
color: white;
margin: 0 auto;
overflow: hidden;
}

.menu a {
float: left;
color: white;
text-align: center;
padding: 14px 16px;
text-decoration: none;
font-size: 20px;
}

.menu-log {
right: auto;
float: right;
}

footer {
width: 100%;
bottom: 0px;
background-color: #000;
color: #fff;
position: absolute;
padding-top: 20px;
padding-bottom: 50px;
text-align: center;
font-size: 30px;
```



```
        font-weight: bold;
    }

    .body_sec {
        margin-left: 20px;
    }
</style>
</head>

<body>

    <!-- Header Section -->
    <header>
        <div class="head1">
            Sardar Azeem Website Developer
        </div>
        <div class="head2">
            A Website Development Platform For PICT Academy Students
        </div>
    </header>

    <!-- Menu Navigation Bar -->
    <nav class="menu">
        <a href="#home">HOME</a>
        <a href="#news">NEWS</a>
        <a href="#notification">
            NOTIFICATIONS
        </a>
        <div class="menu-log">
            <a href="#login">LOGIN</a>
        </div>
    </nav>

    <!-- Body section -->
    <main class="body_sec">
        <section id="Content">
            <h3>Content section</h3>
        </section>
    </main>

    <!-- Footer Section -->
    <footer>Footer Section</footer>
</body>
```

```
</html>
```

HTML Computer code

When we are programming, sometimes it is mandatory to show the Output result, error message, or coding part to user on a webpage. Hence to solve this issue HTML uses different tags for the user inputs, codes, programs, etc. With the help of these tags, you will be able to write codes to display on your webpage.

Following is a list of some tags which are used in HTML for this task.

- `<code>`
- `<kbd>`
- `<samp>`
- `<var>`
- `<pre>`

HTML `<code>` element

It is used to represent some programming code on your website. The content written between tag will be displayed in default monospace font.

Example:

```
<!DOCTYPE html>
<html>
<body>
<h2>Computer Code</h2>
<p>This is a programming code:</p>
<code>
x = 5;<br>
y = 6;<br>
z = x + y;
</code>
</body>
</html>
```

HTML <kbd> Element

It is used to represent user input, keyboard input, voice command etc. Text written within <kbd>.....</kbd> tags is typically displayed in the browser's default monospace font.

Example:

```
<!DOCTYPE html>
<html>
<body>
<h2>The kbd Element</h2>
<kbd>This is how content written within kbd element looks like.</kbd></p>
</body>
</html>
```

HTML <samp> Element

The HTML <samp> element is used to represent a program's output. Text written within samp element is typically displayed in the browser's default monospace font.

Example:

```
<!DOCTYPE html>
<html>
<body>
<h2>The samp Element</h2>
<samp>This is how the content within samp element looks like. </samp>
</body>
</html>
```

HTML <var> element

The HTML <var> element is used to define a variable. The variable could be a variable in a mathematical expression or a variable in programming context.

Example:

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

```
<body>
<h2>The var Element</h2>
<p>This is a famous formula: <var>E</var> = <var>mc</var><sup>2</sup>.</p>
</body>
</html>
```

HTML <pre> element

The <pre> element defines preformatted text, which displays the content within it in a fixed-width font. It keeps the content into its original format and ignores all formatting.

Example:

```
<!DOCTYPE html>
<html>
<body>
<h3>Example of pre tag</h3>
  <pre>
    This is content written
    within pre tag, and pre tag will ignore all
    spaces, break lines, and will display the content
    as in original format.
  </pre>
</body>
</html>
```

HTML 5 Event Attributes

When a browser reacts on user action, then it is called as an event. For example, when you click on the submit button, then if the browser displays an information box.

In HTML5 there are lots of event attributes available which can be activated using a programming language such as JavaScript.

Following is a table of event attributes, using these attributes you can perform several events.

Windows Event Attributes

Windows events are related for the window object, and it can only be applied with <body> tag.

Attribute	Description
onafterprint	Executed the script after the document is printed.
onbeforeprint	Executed the script before the document is printed.
onbeforeunload	Executed the script before a document being unloaded.
onerror	Executed the script when an error occurs.
onhashchange	Executed the script when the anchor part in URL of the webpage is changed.
onload	Executed the script when the webpage is entirely loaded.
onmessage	Executed the script when a message event occurs.
onoffline	Executed the script when the network connection is disconnected, and browser started working offline.
ononline	Executed the script when the browser started working online
onpagehide	Executed the script when the current webpage is hidden such as if the user has moved away from the current webpage.
onpageshow	Executed the script when the current webpage is focused.
onpopstate	Executed the script when the window's active history is changed.
onresize	Executed the script when the window is resized.
onstorage	Executed the script when web storage is updated.
onunload	Executed the script when the current webpage is unloaded, or window is closed.

Form Event Attributes

Form event occurs when the user performs some action within the form such as submitting the form, selecting input field, etc.

The form events can be used with any element, but these are mainly used with HTML form elements.

Attribute	Description
onblur	Executed the script when form element loses the focus.
onchange	Executed the script when the value of the element is changed.
onfocus	Trigger an event when the element gets focused.
oninput	Executed the script when the user enters input to the element.
oninvalid	Executed the script when the element does not satisfy its predefined constraints.
onreset	Triggers the event when user reset the form element values.
onsearch	Triggers the event when a search field receives some input.

onselect	Triggers the event when the user has selected some text.
onsubmit	Triggers the event when a form is submitted.

Keyboard Event Attributes

Keyboard event occurs when a user interacts with the keyboard. Following is a list of the Keyboard event.

Attribute	Description
onkeydown	Triggers the event when the user presses down a key on the keyboard.
onkeypress	Trigger the event when the user presses the key which displays some character.
onkeyup	Trigger the event when the user releases the currently pressed key.

Mouse Event Attributes

Attribute	Description
onclick	Trigger the event when the mouse clicks on the element.
ondblclick	Trigger the event when mouse double-click occurs on the element.
onmousedown	Trigger the event when the mouse button is pressed on the element.
onmousemove	Trigger the event when the mouse pointer moves over the element.
onmouseout	Trigger the event when the mouse moves outside the element.
onmouseover	Trigger the event when the mouse moves onto the element.
onmouseup	Trigger the event when the mouse button is released.
onmousewheel	Deprecated. Use the onwheel attribute.
onwheel	Trigger the event when the mouse wheel rolls up or down on the element

Clipboard Event Attributes

Attribute	Description
oncopy	Trigger the event when the user copies the content to the system clipboard.
oncut	Trigger the event when the content of an element is cut and copy to the clipboard.
onpaste	Trigger the event when the user pastes some content in an element.

Media Event Attributes

Attribute	Description
onabort	Executed the script when media playback is aborted.
oncanplay	Executed the script when the media file is ready to play.
oncanplaythrough	Executed the script when the media file is ready to play without buffering or stopping.
oncuechange	Executed the script text cue of <track> element is changed.
ondurationchange	Executed the script when the media file duration is changed.
onemptied	Executed the script if media occurs some fatal error, and the file becomes unavailable.
onended	Executed the script when the media file occurs its end point.
onerror	Executed the script when some error occurred while fetching the media data.
onloadeddata	Executed the script when media data is loaded.
onloadedmetadata	Executed the script when metadata of media file is loaded.
onloadstart	Executed the script when loading of media file starts.
onpause	Executed the script when media playback is paused.
onplay	Executed the script when media file ready to play after being paused.
onplaying	Executed the script when media file is started playing.
onprogress	Executed the script when the browser is in the process of getting the media data.
onratechange	Executed the script when playback speed changed.
onseeked	Executed the script when seek operation is ended and seeking attribute is set to false.
onseeking	Executed the script when seek operation is active and seeking attribute is set to true.
onstalled	Executed the script when browser unexpectedly stopped fetching the data media.
onsuspend	Executed the script if fetching of media data is intentionally stopped.
ontimeupdate	Executed the script when playback position is changed, such as if a user fasts forward the track.
onvolumechange	Executed the script when media volume is changed (muted or unmuted).
onwaiting	Executed the script if playback pause to wait for loading more data.

HTML 5 Google Maps

HTML Google Map is used to display maps on your webpage. You can simply add a map on your basic HTML page.

Syntax:

```
<!DOCTYPE html>
<html>
<body>
<h1>First Google Map Example</h1>
<div id="map">My map will go here...</div>
</body>
</html>
```

Set the Map Size

You can set the map size by using the following syntax:

```
<div id="map" style="width:400px;height:400px;background:grey"></div>
```

You can set the map properties by creating a function. Here, the function is myMap(). This example shows the Google map centered in London, England.

We have to use the functionalities of Google Maps API provided by a JavaScript library located at Google. Use the following script to refer to the Google Maps API with a callback to the myMap function.

```
<script src="https://maps.googleapis.com/maps/api/js?callback=myMap"></script>
```

Example:

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Google Map</h1>
<div id="map" style="width:400px;height:400px;background:grey"></div>
<script>
function myMap() {
var mapOptions = {
```



```

    center: new google.maps.LatLng(51.5, -0.12),
    zoom: 10,
    mapTypeId: google.maps.MapTypeId.HYBRID
}
var map = new google.maps.Map(document.getElementById("map"), mapOptions);
}
</script>
<script src="https://maps.googleapis.com/maps/api/js?key=AlzaSyBu-
916DdpKAjTmJNlgngS6HL_kDIKU0aU&callback=myMap"></script>
</body>
</html>

```

mapOptions: It is a variable which defines the properties for the map.

center: It specifies where to center the map (using latitude and longitude coordinates).

zoom: It specifies the zoom level for the map (try to experiment with the zoom level).

mapTypeId: It specifies the map type to display. The following map types are supported: ROADMAP, SATELLITE, HYBRID, and TERRAIN.

var map=new google.maps.Map(document.getElementById("map"), mapOptions): It creates a new map inside the

element with id="map", using the parameters that are passed (mapOptions).

HTML Multiple Maps

You can use different map types in a single example.

Example:

```

<!DOCTYPE html>
<html>
<body>
<div id="googleMap1" style="width:400px;height:300px;"></div>
<br>
<div id="googleMap2" style="width:400px;height:300px;"></div>

```


<div id="googleMap3" style="width:400px;height:300px;"></div>

<div id="googleMap4" style="width:400px;height:300px;"></div>

<script>

```
function myMap() {
```

```
  var mapOptions1 = {
```

```
    center: new google.maps.LatLng(51.508742,-0.120850),
```

```
    zoom:9,
```

```
    mapTypeId: google.maps.MapTypeId.ROADMAP
```

```
  };
```

```
  var mapOptions2 = {
```

```
    center: new google.maps.LatLng(51.508742,-0.120850),
```

```
    zoom:9,
```

```
    mapTypeId: google.maps.MapTypeId.SATELLITE
```

```
  };
```

```
  var mapOptions3 = {
```

```
    center: new google.maps.LatLng(51.508742,-0.120850),
```

```
    zoom:9,
```

```
    mapTypeId: google.maps.MapTypeId.HYBRID
```

```
  };
```

```
  var mapOptions4 = {
```

```
    center: new google.maps.LatLng(51.508742,-0.120850),
```

```
    zoom:9,
```

```
    mapTypeId: google.maps.MapTypeId.TERRAIN
```

```
  };
```

```
  var map1 = new google.maps.Map(document.getElementById("googleMap1"),mapOptions1);
```

```

var map2 = new google.maps.Map(document.getElementById("googleMap2"),mapOptions2);

var map3 = new google.maps.Map(document.getElementById("googleMap3"),mapOptions3);

var map4 = new google.maps.Map(document.getElementById("googleMap4"),mapOptions4);
}
</script>

<script src="https://maps.googleapis.com/maps/api/js?key=AlzaSyBu-916DdpKAjTmJNlgnngS6HL_kDIKU0aU&callback=myMap"></script>

</body>
</html>

```

HTML Multimedia

Multimedia on the web is sound, music, videos, movies, and animations.

HTML Video

The HTML <video> element is used to show a video on a web page.

| Example 1 | Example 2 |
|--|--|
| <pre> <!DOCTYPE html> <html> <body> <video width="400" controls> <source src="mov_bbb.mp4" type="video/mp4"> <source src="mov_bbb.ogv" type="video/ogg"> Your browser does not support HTML video. </video> <p> Video courtesy of Big Buck Bunny. </p></body> </html> </pre> | <pre> <!DOCTYPE html> <html> <body> <video width="320" height="240" autoplay muted> <source src="movie.mp4" type="video/mp4"> <source src="movie.ogv" type="video/ogg"> Your browser does not support the video tag. </video> </body> </html> </pre> |

HTML Audio

The HTML `<audio>` element is used to play an audio file on a web page.

| Example 1 | Example 2 |
|--|---|
| <pre><!DOCTYPE html> <html> <body> <audio controls> <source src="horse.ogg" type="audio/ogg"> <source src="horse.mp3" type="audio/mpeg"> Your browser does not support the audio element. </audio> </body> </html></pre> | <pre><!DOCTYPE html> <html> <body> <audio controls autoplay muted> <source src="horse.ogg" type="audio/ogg"> <source src="horse.mp3" type="audio/mpeg"> Your browser does not support the audio element. </audio> </body> </html></pre> |

HTML Plug-ins

Plug-ins are computer programs that extend the standard functionality of the browser.

Plug-ins were designed to be used for many different purposes:

- To run Java applets
- To run Microsoft ActiveX controls
- To display Flash movies
- To display maps
- To scan for viruses
- To verify a bank id

| | |
|--|--|
| <p>The <code><object></code> Element</p> <p>The <code><object></code> element is supported by all browsers.</p> <p>The <code><object></code> element defines an embedded object within an HTML document.</p> <p>It was designed to embed plug-ins (like Java applets, PDF readers, and Flash Players) in web pages, but can also be used to include HTML in HTML:</p> | <p>The <code><embed></code> Element</p> <p>The <code><embed></code> element is supported in all major browsers.</p> <p>The <code><embed></code> element also defines an embedded object within an HTML document.</p> <p>Web browsers have supported the <code><embed></code> element for a long time. However, it has not been a part of the HTML specification before HTML5.</p> |
|--|--|

| | |
|--|--|
| <pre><!DOCTYPE html> <html> <body> <object width="100%" height="500px" data="snippet.html"></object> </body> </html></pre> | <pre><!DOCTYPE html> <html> <body> <embed src="audi.jpeg"> </body> </html></pre> |
|--|--|

HTML YouTube Video

The easiest way to play videos in HTML, is to use YouTube.

YouTube Video Id

YouTube will display an id (like tgbNymZ7vqY), when you save (or play) a video.

You can use this id, and refer to your video in the HTML code.

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 | Example 2 |
|---|---|
| <pre><!DOCTYPE html> <html> <body> <iframe width="420" height="345" src="https://www.youtube.com/embed/t gbNymZ7vqY"> </iframe> </body> </html></pre> | <pre><!DOCTYPE html> <html> <body> <iframe width="420" height="345" src="https://www.youtube.com/embed/tgb NymZ7vqY?autoplay=1&mute=1"> </iframe> </body> </html></pre> |
| Example 3 | Example 4 |
| <pre><!DOCTYPE html></pre> | <pre><!DOCTYPE html></pre> |

| | |
|--|--|
| <pre><html> <body> <iframe width="420" height="345" src="https://www.youtube.com/embed/t gbNymZ7vqY?playlist=tgbNymZ7vqY&loo p=1"> </iframe> </body> </html></pre> | <pre><html> <body> <iframe width="420" height="345" src="https://www.youtube.com/embed/tgb NymZ7vqY?controls=0"> </iframe> </body> </html></pre> |
|--|--|

Coding Quotes For Students

CODING LIKE POETRY SHOULD BE SHORT AND CONCISE. — SARDAR AZEEM

IT'S NOT A BUG; IT'S AN UNDOCUMENTED FEATURE. — SARDAR AZEEM

FIRST, SOLVE THE PROBLEM. THEN, WRITE THE CODE. — SARDAR AZEEM

CODE IS LIKE HUMOR. WHEN YOU HAVE TO EXPLAIN IT, IT'S BAD. — SARDAR AZEEM

MAKE IT WORK, MAKE IT RIGHT, MAKE IT FAST. — SARDAR AZEEM

CLEAN CODE ALWAYS LOOKS LIKE IT WAS WRITTEN BY SOMEONE WHO CARES. — SARDAR AZEEM

OF COURSE, BAD CODE CAN BE CLEANED UP. BUT IT'S VERY EXPENSIVE." — SARDAR AZEEM

CSS

What is CSS?

CSS stands for Cascading style sheets. It describes to the user how to display HTML elements on the screen in a proper format. CSS is the language that is used to style HTML documents. In simple words, cascading style sheets are a language used to simplify the process of making a webpage.

CSS is used to handle some parts of the webpage. With the help of CSS, we can control the color of text and style of fonts, and we can control the spacing between the paragraph and many more things. CSS is easy to understand but provides strong control on the HTML documents. CSS is combined with HTML.

Advantages of CSS

- **Faster page speed:** It has a faster page speed than other code's page speeds. With the help of the CSS rule, we can apply it to all occurrences of certain tags in HTML documents.
- **Better user experience:** CSS makes a webpage very attractive to the eyes. Also, CSS makes it user-friendly. When the button or text is in a proper format, it improves the user experience.
- **Quicker Development time:** With the help of CSS, we can specify the format and style the multiple pages into one code string. In cascading style sheet, we can make a duplicate copy of several website pages.
If we make a webpage, it has the same formatting, looks, and feel, so with the help of the CSS rule for one page, and it is sufficient for all the pages.
- **Easy Formatting changes:** In CSS, if we need to make changes in the format, it is very easy; we only need to change the one-page format it will automatically apply to the other pages of CSS.
There is no need to correct individual pages in a CSS style sheet. If we fix a CSS style sheet, it will automatically update the other CSS style sheet.
- **Compatibility:** Compatibility is very important in today's age. If we create any webpage, it should be very responsive and user-friendly. CSS is used with HTML to make webpage design responsive.

CSS Syntax

The CSS provides the style to the HTML element, which the browser interprets. After being interpreted by the browser, the CSS style property will be applied to all the elements of the HTML. We can provide style property to the HTML element in three parts. These three parts are as follows.

1. Selector

It is an HTML tag. All the style properties of the CSS will be applied to the selector. The selector tag like `<h1>` or `<table>` etc.

2. Property

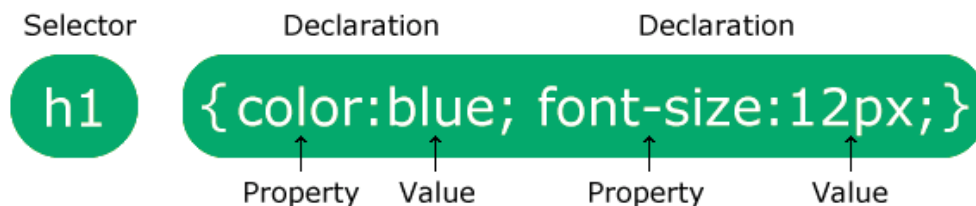
It is a type of attribute that is present in HTML tags. All the attributes of the HTML will be converted to the CSS properties. The CSS properties like color, border, etc.

3. Value

In HTML, these are assigned to the properties. For example, the color property can have a value of either red or #F1F1F1, etc.

Syntax:

```
selector { property: value }
```



| Example 1 | Example 2 |
|---|--|
| <pre><!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device- width, initial-scale=1.0"> <title>Document</title> <!--Application Of CSS -- > <style> body { font-family: Arial, sans-serif; margin: 0;</pre> | <pre><!DOCTYPE html> <html> <head> <style> body { background-color: lightblue; } h1 { color: white; text-align: center; }</pre> |


```
padding: 0;
background-color: #f2f2f2;
}

#header {
background-color: #333;
color: white;
text-align: center;
padding: 20px;
}

h1 {
font-size: 36px;
margin-bottom: 20px;
}

p {
font-size: 18px;
margin-bottom: 10px;
}

.button {
background-color: #ff0000;
color: white;
padding: 10px 20px;
border: none;
text-align: center;
text-decoration: none;
display: inline-block;
font-size: 16px;
border-radius: 5px;
cursor: pointer;
}

.button:hover {
background-color: #990000;
}
</style>
</head>
<body>
<div id="header">
<h1>Lets Start Learning CSS From
Experts!</h1>
```

```
p {
font-family: verdana;
font-size: 20px;
}
</style>
</head>
<body>

<h1>My First CSS Example</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

```
<p>Welcome to The World Of Styling a
Website:</p>
<button class="button">Click Me!</button>
</div>
</body>
</html>
```

Types of CSS (Where To Write CSS)

There are three types of CSS:

1. Inline CSS

Inline CSS is used to style the elements of HTML documents. It is used in HTML to style the attributes without using the selectors. It is challenging to manage the inline function in websites compared to other types. It is very helpful in HTML in some situations.

Example of inline CSS:

```
<p style="color: orange; font-size: 25px;">Here is my first paragraph.</p>
```

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>
```

2. Internal CSS

Internal CSS is used to design the style single page effectively. It is more time-consuming because we can only work on one page or we need to style each web page. In internal CSS, we style a single webpage uniquely.

Syntax:

```
<style>
```

```
--- required styles--
```

</style>

Example
<pre><!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></pre>

3. External CSS

External CSS is used to link all webpage with an external file. CSS, which can be created in a text file. It is more efficient for styling an extensive webpage. It also increases the readability of the CSS files.

1. Create CSS File and save it by "name.css".
2. Create html file and link CSS by link tag as follows.

Syntax:

```
<head>
  <link rel="stylesheet" href="nameOfTheSheet.css">
</head>
```

CSS File Code	HTML File Code
<pre>body { background-color: lightblue; }</pre>	<pre><!DOCTYPE html> <html> <head></pre>

<pre>h1 { color: navy; margin-left: 20px; }</pre>	<pre><link rel="stylesheet" href="mystyle.css"> </head> <body> <h1>This is a heading</h1> <p>This is a paragraph.</p> </body> </html></pre>
---	---

CSS Selector

CSS selectors are used to select the content you want to style. Selectors are the part of CSS rule set. CSS selectors select HTML elements according to its id, class, type, attribute etc.

There are several different types of selectors in CSS.

1. CSS Element Selector
2. CSS Id Selector
3. CSS Class Selector
4. CSS Universal Selector
5. CSS Group Selector

1) CSS Element Selector

The element selector selects the HTML element by name. e.g. (p,h1,body)

Example
<pre><!DOCTYPE html> <html> <head> <style> p{ text-align: center; color: blue; } </style> </head></pre>

```
<body>
<p>This style will be applied on every paragraph.</p>
<p id="para1">Me too!</p>
<p>And me!</p>
</body>
</html>
```

2) CSS Id Selector

The id selector selects the id attribute of an HTML element to select a specific element. An id is always unique within the page so it is chosen to select a single, unique element.

It is written with the hash character (#), followed by the id of the element.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
#para1 {
    text-align: center;
    color: blue;
}
</style>
</head>
<body>
<p id="para1">I am Called By ID</p>
<p>This paragraph will not be affected. </p>
</body>
</html>
```

3) CSS Class Selector

The class selector selects HTML elements with a specific class attribute. It is used with a period character. (full stop symbol) followed by the class name.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
.center {
  text-align: center;
  color: blue;
}
</style>
</head>
<body>
<h1 class="center">This heading is blue and center-aligned.</h1>
<p class="center">This paragraph is blue and center-aligned.</p>
</body>
</html>
```

CSS Class Selector for specific element

If you want to specify that only one specific HTML element should be affected then you should use the element name with class selector.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
p.center {
```

```
    text-align: center;
    color: blue;
}
</style>
</head>
<body>
<h1 class="center">This heading is not affected</h1>
<p class="center">This paragraph is blue and center-aligned.</p>
</body>
</html>
```

4) CSS Universal Selector

The universal selector is used as a wildcard character (* Estaric). It selects all the elements on the pages.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
* {
  color: green;
  font-size: 20px;
}
</style>
</head>
<body>
<h2>This is heading</h2>
<p>This style will be applied on every paragraph.</p>
<p id="para1">Me too!</p>
```

```
<p>And me!</p>
</body>
</html>
```

5) CSS Group Selector

The grouping selector is used to select all the elements with the same style definitions.

Grouping selector is used to minimize the code. Commas are used to separate each selector in grouping.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
h1, h2, p {
  text-align: center;
  color: blue;
}
</style>
</head>
<body>
<h1>Hello Pictacademy.com</h1>
<h2> Hello Pictacademy.com (In smaller font) </h2>
<p>This is a paragraph Hello Pictacademy.com. </p>
</body>
</html>
```

CSS Comments

Comments are used to explain the code, and may help when you edit the source code at a later date.

Comments are ignored by browsers.

A CSS comment is placed inside the <style> element, and starts with /* and ends with */:

Example

```
/* This is a single-line comment */
```

```
p {  
  color: red;  
}
```

```
/* This is a
```

```
Multi-line
```

```
comment */
```

Example

```
<!DOCTYPE html>  
<html>  
<head>  
<style>  
p {  
  color: red; /* Set text color to red */  
}  
</style>  
</head>  
<body>  
  
<h2>My Heading</h2>  
  
<!-- These paragraphs will be red -->  
<p>Hello World!</p>  
<p>This paragraph is styled with CSS.</p>  
<p>HTML and CSS comments are not shown in the output.</p>  
  
</body>  
</html>
```

CSS Colors

Colors are specified using predefined color names, or RGB, HEX, HSL, RGBA, HSLA values.

CSS Color Names

In CSS, a color can be specified by using a predefined color name:

```
color: "cyan";
```

background-color: Dodger Blue;

Example Colors By Name

```
<!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>
```

Example 2 Colors By Name (Text Color)

```
<!DOCTYPE html>
<html>
<body>
<h3 style="color:Tomato;">Hello World</h3>
<p style="color:DodgerBlue;">Lorem ipsum dolor sit amet, consectetur adipiscing elit,
sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat
volutpat.</p>
<p style="color:MediumSeaGreen;">Ut wisi enim ad minim veniam, quis nostrud exerci
tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.</p>
</body>
```

```
</html>
```

Example 3 Colors by Name (Border Color)

```
<!DOCTYPE html>  
<html>  
<body>  
<h1 style="border: 2px solid Tomato;">Hello World</h1>  
<h1 style="border: 2px solid DodgerBlue;">Hello World</h1>  
<h1 style="border: 2px solid Violet;">Hello World</h1>  
</body>  
</html>
```

CSS RGB Colors

An RGB color value represents RED, GREEN, and BLUE light sources.

RGB Value

Syntax → `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)`.

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:

Syntax → `rgba(red, green, blue, alpha)`

The alpha parameter is a number between 0.0 (fully transparent) and 1.0

Example Simple RGB Colors	Example Using RGBA Colors
<pre><!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></pre>	<pre><!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></pre>
CSS HEX Colors	

A hexadecimal color is specified with: #RRGGBB, where the RR (red), GG (green) and BB (blue) hexadecimal integers specify the components of the color.

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.

The 3-digit hex code is a shorthand for some 6-digit hex codes.

The 3-digit hex code has the following form:

#rgb

Where r, g, and b represent the red, green, and blue components with values between 0 and f.

The 3-digit hex code can only be used when both the values (RR, GG, and BB) are the same for each component. So, if we have #ff00cc, it can be written like this: #f0c.

Example 1 HEX Colors	Example 2 HEX Colors
<pre><!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></pre>	<pre><!DOCTYPE html> <html> <head> <style> body { background-color: #fc9; /* same as #ffcc99 */ } h1 { color: #f0f; /* same as #ff00ff */ } p { color: #b58; /* same as #bb5588 */ } </style> </head> <body> <h1>CSS 3-digit Hex Code</h1> <p>This is a paragraph.</p> </body> </html></pre>

CSS HSL Colors

HSL stands for hue, saturation, and lightness.

HSL Value

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

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 Using HSL Colors	Example Using HSLA Colors
<pre><!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></pre>	<pre><!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></pre>

CSS Background

CSS background property is used to define the background effects on element. There are 5 CSS background properties that affects the HTML elements:

1. background-color
2. background-image

3. background-repeat
4. background-attachment
5. background-position

1) CSS background-color

The background-color property is used to specify the background color of the element.

Example 1	Example 2
<pre> <!DOCTYPE html> <html> <head> <style> h1 { background-color: green; } div { background-color: lightblue; } p { background-color: yellow; } </style> </head> <body> <h1>CSS background-color example!</h1> <div> This is a text inside a div element. <p>This paragraph has its own background color.</p> We are still in the div element. </div> </body> </html> </pre>	<pre> <!DOCTYPE html> <html> <head> <style> body { background-color: lightblue; } h2,p{ background-color: #b0d4de; } </style> </head> <body> <h2>My first CSS page.</h2> <p>Hello Javatpoint. This is an example of CSS background-color.</p> </body> </html> </pre>

Opacity / Transparency

The opacity property specifies the opacity/transparency of an element. It can take a value from 0.0 - 1.0. The lower value, the more transparent:

Example 1	Example 2
<pre> <!DOCTYPE html> </pre>	<pre> <!DOCTYPE html> </pre>

```

<html>
<head>
<style>
div {
  background-color: green;
}

div.first {
  opacity: 0.1;
}

div.second {
  opacity: 0.3;
}

div.third {
  opacity: 0.6;
}
</style>
</head>
<body>
<div class="first">
  <h1>opacity 0.1</h1>
</div>

<div class="second">
  <h1>opacity 0.3</h1>
</div>

<div class="third">
  <h1>opacity 0.6</h1>
</div>

<div>
  <h1>opacity 1 (default)</h1>
</div>

</body>
</html>

```

```

<html>
<head>
<style>
div {
  background: rgb(0, 128, 0);
}

div.first {
  background: rgba(0, 128, 0, 0.1);
}

div.second {
  background: rgba(0, 128, 0, 0.3);
}

div.third {
  background: rgba(0, 128, 0, 0.6);
}
</style>
</head>
<body>

<h1>Transparent Boxes 2</h1>

<p>Result with opacity:</p>

<div style="opacity:0.1;">
  <h1>10% opacity</h1>
</div>

<div style="opacity:0.3;">
  <h1>30% opacity</h1>
</div>

<div style="opacity:0.6;">
  <h1>60% opacity</h1>
</div>

<div>
  <h1>opacity 1</h1>
</div>

<p>Result with rgba():</p>

```


	<pre> <div class="first"> <h1>10% opacity</h1> </div> <div class="second"> <h1>30% opacity</h1> </div> <div class="third"> <h1>60% opacity</h1> </div> <div> <h1>default</h1> </div> </body> </html> </pre>
--	---

2) CSS background-image

The background-image property is used to set an image as a background of an element. By default the image covers the entire element. You can set the background image for a page like this.

Example 1	Example 2
<pre> <!DOCTYPE html> <html> <head> <style> body { background-image: url("paper1.gif"); margin-left:100px; } </style> </head> <body> <h1>Hello pictacademy.com</h1> </body> </html> </pre>	<pre> <!DOCTYPE html> <html> <head> <style> body { background-image: url("paper.gif"); } </style> </head> <body> <h1>Hello World!</h1> <p>This page has an image as the background!</p> </body> </html> </pre>

3) CSS background-repeat

By default, the background-image property repeats the background image horizontally and vertically. Some images are repeated only horizontally or vertically.

The background looks better if the image repeated horizontally only.

Example 1	Example 2
<pre><!DOCTYPE html> <html> <head> <style> body { background-image: url("gradient- bg.png"); background-repeat: repeat-x; } </style> </head> <body> <h1>Hello LearnFromTheExperts.com</h1> </body> </html></pre>	<pre><!DOCTYPE html> <html> <head> <style> body { background-image: url("gradient- bg.png"); background-repeat: repeat-y; } </style> </head> <body> <h1>Hello Learnfromtheexperts.com</h1> </body> </html></pre>
Example 3	Example 4
<pre><!DOCTYPE html> <html> <head> <style> body { background-image: url("gradient_bg.png"); background-repeat: repeat-x; } </style> </head> <body> <h1>Hello World!</h1> <p>Here, a background image is repeated only horizontally!</p> </body> </html></pre>	<pre><!DOCTYPE html> <html> <head> <style> body { background-image: url("img_tree.png"); background-repeat: no-repeat; } </style> </head> <body> <h1>Hello World!</h1> <p>W3Schools background image example.</p> <p>The background image only shows once, but it is disturbing the reader!</p> </body> </html></pre>

4) CSS background-attachment

The background-attachment property is used to specify if the background image is fixed or scroll with the rest of the page in browser window. If you set fixed the background image then the image will not move during scrolling in the browser.

Example 1	Example 2
<pre data-bbox="203 451 787 976"><!DOCTYPE html> <html> <head> <style> body { background-image: url("img_tree.png"); background-repeat: no-repeat; background-position: right top; margin-right: 200px; background-attachment: fixed; } </style> </head> <body></pre> <p data-bbox="203 1018 787 1081"><h1>The background-attachment Property</h1></p> <p data-bbox="203 1134 787 1281"><p>The background-attachment property specifies whether the background image should scroll or be fixed (will not scroll with the rest of the page).</p></p> <p data-bbox="203 1323 787 1428"><p>Tip: If you do not see any scrollbars, try to resize the browser window.</p></p> <p data-bbox="203 1470 787 1543"><p>The background-image is fixed. Try to scroll down the page.</p></p> <p data-bbox="203 1554 787 1627"><p>The background-image is fixed. Try to scroll down the page.</p></p> <p data-bbox="203 1638 787 1711"><p>The background-image is fixed. Try to scroll down the page.</p></p> <p data-bbox="203 1722 787 1795"><p>The background-image is fixed. Try to scroll down the page.</p></p> <p data-bbox="203 1806 787 1879"><p>The background-image is fixed. Try to scroll down the page.</p></p>	<pre data-bbox="820 451 1404 976"><!DOCTYPE html> <html> <head> <style> body { background-image: url("img_tree.png"); background-repeat: no-repeat; background-position: right top; margin-right: 200px; background-attachment: scroll; } </style> </head> <body></pre> <p data-bbox="820 1018 1404 1081"><h1>The background-attachment Property</h1></p> <p data-bbox="820 1134 1404 1281"><p>The background-attachment property specifies whether the background image should scroll or be fixed (will not scroll with the rest of the page).</p></p> <p data-bbox="820 1323 1404 1428"><p>Tip: If you do not see any scrollbars, try to resize the browser window.</p></p> <p data-bbox="820 1470 1404 1543"><p>The background-image scrolls. Try to scroll down the page.</p></p> <p data-bbox="820 1554 1404 1627"><p>The background-image scrolls. Try to scroll down the page.</p></p> <p data-bbox="820 1638 1404 1711"><p>The background-image scrolls. Try to scroll down the page.</p></p> <p data-bbox="820 1722 1404 1795"><p>The background-image scrolls. Try to scroll down the page.</p></p> <p data-bbox="820 1806 1404 1879"><p>The background-image scrolls. Try to scroll down the page.</p></p>

3. bottom
4. left
5. right

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
background: white url('good-morning.jpg');
background-repeat: no-repeat;
background-attachment: fixed;
background-position: center;
}
</style>
</head>
<body>
<p>This is a fixed background-image. Scroll down the page.</p>
<p>This is a fixed background-image. Scroll down the page.</p>
<p>This is a fixed background-image. Scroll down the page.</p>
<p>This is a fixed background-image. Scroll down the page.</p>
<p>This is a fixed background-image. Scroll down the page.</p>
<p>This is a fixed background-image. Scroll down the page.</p>
<p>This is a fixed background-image. Scroll down the page.</p>
<p>This is a fixed background-image. Scroll down the page.</p>
<p>If you do not see any scrollbars, Resize the browser window.</p>
</body>
```

```
</html>
```

CSS background - Shorthand property

To shorten the code, it is also possible to specify all the background properties in one single property. This is called a shorthand property.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background: #ffffff url("img_tree.png") no-repeat right top;
  margin-right: 200px;
}
</style>
</head>
<body>
<h1>The background Property</h1>
<p>The background property is a shorthand property for specifying all the background
properties in one declaration.</p>
<p>Here, the background image is only shown once, and it is also positioned in the top-
right corner.</p>
<p>We have also added a right margin, so that the text will not write over the
background image.</p>
</body>
</html>
```

CSS Border

CSS border is a key property used to characterize and style the borders around HTML components. Borders assume a vital part in website composition, assisting with

making separation, emphasis, and stylish allure. In CSS, you can utilize a few border-related properties to control the style, variety, size, and shape of these borders.

CSS Border Properties

The CSS border properties are utilized to determine the style, variety, width, and ebb and flow of the borders of a component. These properties include:

1. border-style
2. border-color
3. border-width
4. border-radius

1) CSS border-style

The Border style property is used to specify the border type which you want to display on the web page.

Value	Description
none	It doesn't define any border.
dotted	It is used to define a dotted border.
dashed	It is used to define a dashed border.
solid	It is used to define a solid border.
double	It defines two borders with the same border-width value.
groove	It defines a 3d grooved border. effect is generated according to border-color value.
ridge	It defines a 3d ridged border. effect is generated according to border-color value.
inset	It defines a 3d inset border. effect is generated according to border-color value.
outset	It defines a 3d outset border. effect is generated according to border-color value.

Example 1	Example 2
<pre><!DOCTYPE html> <html> <head> <style> .border-example { width: 150px; height: 30px; margin: 10px; padding: 10px; } .dotted {</pre>	<pre><!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;}</pre>

<pre> border: 2px dotted #FFA500; } .dashed { border: 2px dashed #008000; } .solid { border: 2px solid #000; } .double { border: 4px double #FF0000; } .groove { border: 3px groove #3333FF; } .ridge { border: 3px ridge #660066; } .inset { border: 3px inset #006600; } .outset { border: 3px outset #990000; } </style> </head> <body> <div class = "border- example dotted"> Dotted Border </div> <div class = "border- example dashed"> Dashed Border </div> <div class = "border- example solid"> Solid Border </div> <div class = "border- example double"> Double Border </div> <div class = "border- example groove"> Groove Border </div> </pre>	<pre> 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> </pre>
--	--

<pre> <div class = "border- example ridge"> Ridge Border </div> <div class "border- example inset"> Inset Border </div> <div class = "border- example outset"> Outset Border </div> </body> </html> </pre>	
--	--

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:

Example 1	Example 2
<pre> <!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; } </pre>	<pre> <!DOCTYPE html> <html> <head> <style> p.one { border-style: solid; border-width: 5px 20px; /* 5px top and bottom, 20px on the sides */ } p.two { border-style: solid; border-width: 20px 5px; /* 20px top and bottom, 5px on the sides */ } p.three { border-style: solid; border-width: 25px 10px 4px 35px; /* 25px top, 10px right, 4px bottom and 35px left */ } </style> </head> <body> <h2>The border-width Property</h2> </pre>

<pre> p.six { border-style: double; border-width: thick; } </style> </head> <body> <h2>The border-width Property</h2> <p>This property specifies the width of the four borders:</p> <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> <p>Note: The "border-width" property does not work if it is used alone. Always specify the "border-style" property to set the borders first.</p> </body> </html> </pre>	<pre> <p>The border-width property can have from one to four values (for the top border, right border, bottom border, and the left border):</p> <p class="one">Some text.</p> <p class="two">Some text.</p> <p class="three">Some text.</p> </body> </html> </pre>
--	--

CSS Border Color

The border-color property is used to set the color of the four borders.

- 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 1	Example 2
<pre> <!DOCTYPE html> <html> <head> <style> p.one { border-style: solid; border-color: red; </pre>	<pre> <!DOCTYPE html> <html> <head> <style> p.one { border-style: solid; </pre>

<pre> } p.two { border-style: solid; border-color: green; } p.three { border-style: dotted; border-color: blue; } </style> </head> <body> <h2>The border-color Property</h2> <p>This property specifies the color of the four borders:</p> <p class="one">A solid red border</p> <p class="two">A solid green border</p> <p class="three">A dotted blue border</p> <p>Note: The "border-color" property does not work if it is used alone. Use the "border-style" property to set the borders first.</p> </body> </html> </pre>	<pre> border-color: red green blue yellow; /* red top, green right, blue bottom and yellow left */ } </style> </head> <body> <h2>The border-color Property</h2> <p>The border-color property can have from one to four values (for the top border, right border, bottom border, and the left border):</p> <p class="one">A solid multicolor border</p> </body> </html> </pre>
--	--

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

Like you saw in the previous page, there are many properties to consider when dealing with borders.

To shorten the code, it is also possible to specify all the individual border properties in one property.

The border property is a shorthand property for the following individual border properties:

- border-width
- border-style (required)
- border-color

Example

```
<!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:

Example

```
<!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>
```

```
<h2>The border-radius Property</h2>
```

```
<p>This property is used to add rounded borders to an element:</p>
```

```
<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

Margins are used to create space around elements, outside of any defined borders.

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
margin: 70px;
border: 1px solid #4CAF50;
```

```
}
</style>
</head>
<body>
<h2>CSS Margins</h2>
<div>This element has a margin of 70px.</div>
</body>
</html>
```

```
<!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>
```

Margin - Shorthand Property

To shorten the code, it is possible to specify all the margin properties in one property.

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  border: 1px solid black;
  margin: 25px 50px 75px 100px;
  background-color: lightblue;
}
</style>
```

```

</head>
<body>
<h2>The margin shorthand property - 4 values</h2>
<div>This div element has a top margin of 25px, a right margin of 50px, a bottom margin of 75px, and a
left margin of 100px.</div>

<hr>
</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;
}
</style>
</head>
<body>

<h2>CSS Padding</h2>
<div>This element has a padding of
70px.</div>

</body>
</html>

```

```

<!DOCTYPE html>
<html>
<head>
<style>
div {
border: 1px solid black;
background-color: lightblue;
padding-top: 50px;
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>

```

```

<!DOCTYPE html>
<html>
<head>
<style>
div {
border: 1px solid black;
padding: 25px 50px 75px 100px;
background-color: lightblue;
}

```



```

}
</style>
</head>
<body>
<h2>The padding shorthand property - 4 values</h2>
<div>This div element has a top padding of 25px, a right padding of 50px, a bottom
padding of 75px, and a left padding of 100px.</div>
</body>
</html>

```

CSS Height, Width and Max-width

The CSS height and width properties are used to set the height and width of an element.

The CSS max-width property is used to set the maximum width of an element.

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

<pre> <!DOCTYPE html> <html> <head> <style> div { height: 50px; width: 100%; 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> </pre>	<pre> <!DOCTYPE html> <html> <head> <style> div { height: 200px; width: 50%; 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> </pre>
---	--

</html>	</html>
---------	---------

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).

<pre><!DOCTYPE html> <html> <head> <style> div { max-width: 500px; height: 100px; background-color: powderblue; } </style> </head></pre>	<pre><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></pre>
--	--

CSS Outline

An outline is a line drawn outside the element's border.

```
<!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>

```

<pre> <!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> </pre>	<pre> <!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> </pre>
--	---

<pre> <p class="outset">An outset outline. The effect depends on the outline-color value.</p> </body> </html> </pre>	<pre> <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> </pre>
<pre> <!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> </pre>	<pre> <!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> </pre>

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>
```

CSS 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 ``).

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>
```

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

Bootstrap Icons

To use the Bootstrap glyphs, 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">
```

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bo
```

```
otstrap.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>
```

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>
<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>
```

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

- <!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>This is a link</p>
- <p>Note: a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective.</p>
- <p>Note: 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: white;

```



```

color: black;
border: 2px solid green;
padding: 10px 20px;
text-align: center;
text-decoration: none;
display: inline-block;
}

a:hover, a:active {
  background-color: green;
  color: white;
}
</style>
</head>
<body>

<h2>Link Button</h2>

<a href="default.asp" target="_blank">This is a link</a>

</body>
</html>

```

HTML Lists and CSS List Properties

In HTML, there are two main types of lists:

- unordered lists () - the list items are marked with bullets
- ordered lists () - 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.

<!DOCTYPE html>	<!DOCTYPE html>
<html>	<html>
<head>	<head>

<pre> <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"> Coffee Tea Coca Cola <ul class="b"> Coffee Tea Coca Cola <p>Example of ordered lists:</p> <ol class="c"> Coffee Tea Coca Cola <ol class="d"> Coffee </pre>	<pre> <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> Coffee Tea Coca Cola </body> </html> </pre>
--	---

```
<li>Tea</li>
<li>Coca Cola</li>
</ol>
```

```
</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>

```

CSS Layout - The position Property

The position property specifies the type of positioning method used for an element.

There are five different position values:

- static
- relative
- fixed
- absolute
- sticky

Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the position property is set first. They also work differently depending on the position value.

```

<!DOCTYPE html>
<html>
<head>
<style>
div.static {
  position: static;
  border: 3px solid #73AD21;
}
</style>
</head>
<body>

```

```

<!DOCTYPE html>
<html>
<head>
<style>
div.relative {
  position: relative;
  left: 30px;
  border: 3px solid #73AD21;
}
</style>
</head>

```

<pre><h2>position: static;</h2></pre> <p><p>An element with position: static; is not positioned in any special way, it is always positioned according to the normal flow of the page:</p></p> <pre><div class="static"> This div element has position: static; </div></pre> <pre></body> </html></pre>	<pre><body></pre> <pre><h2>position: relative;</h2></pre> <p><p>An element with position: relative; is positioned relative to its normal position:</p></p> <pre><div class="relative"> This div element has position: relative; </div></pre> <pre></body> </html></pre>
<pre><!DOCTYPE html> <html> <head> <style> div.fixed { position: fixed; bottom: 0; right: 0; width: 300px; border: 3px solid #73AD21; } </style> </head> <body></pre> <pre><h2>position: fixed;</h2></pre> <p><p>An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled:</p></p> <pre><div class="fixed"> This div element has position: fixed; </div></pre> <pre></body> </html></pre>	<pre><!DOCTYPE html> <html> <head> <style> div.relative { position: relative; width: 400px; height: 200px; border: 3px solid #73AD21; } div.absolute { position: absolute; top: 80px; right: 0; width: 200px; height: 100px; border: 3px solid #73AD21; } </style> </head> <body></pre> <pre><h2>position: absolute;</h2></pre> <p><p>An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed):</p> </p>

	<pre> <div class="relative">This div element has position: relative; <div class="absolute">This div element has position: absolute;</div> </div> </body> </html> </pre>
<pre> <!DOCTYPE html> <html> <head> <style> div.sticky { position: -webkit-sticky; position: sticky; top: 0; padding: 5px; background-color: #cae8ca; border: 2px solid #4CAF50; } </style> </head> </pre>	<pre> <body> <p>Try to scroll inside this frame to understand how sticky positioning works.</p> <div class="sticky">I am sticky!</div> <div style="padding-bottom:2000px"> <p>In this example, the sticky element sticks to the top of the page (top: 0), when you reach its scroll position.</p> <p>Scroll back up to remove the stickiness.</p> <p>Some text to enable scrolling.. Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.</p> <p>Some text to enable scrolling.. Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.</p> </div> </body> </pre>

```
</html>
```

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:

<pre><!DOCTYPE html> <html> <head> <style> img { position: absolute; left: 0px; top: 0px; z-index: -1; } </style> </head> <body> <h1>This is a heading</h1> <p>Because the image has a z-index of - 1, it will be placed behind the text.</p> </body> </html></pre>	<pre><!DOCTYPE html> <html> <head> <style> .container { position: relative; } .black-box { position: relative; z-index: 1; border: 2px solid black; height: 100px; margin: 30px; } .gray-box { position: absolute; z-index: 3; /* gray box will be above both green and black box */ background: lightgray; height: 60px; width: 70%; left: 50px; top: 50px; } .green-box { position: absolute; z-index: 2; /* green box will be above black box */ background: lightgreen; width: 35%; left: 270px; top: -15px; height: 100px;</pre>
--	--


```
}
</style>
</head>
<body>

<h1>Z-index Example</h1>

<p>An element with greater stack order
is always above an element with a lower
stack order.</p>

<div class="container">
  <div class="black-box">Black box (z-
index: 1)</div>
  <div class="gray-box">Gray box (z-
index: 3)</div>
  <div class="green-box">Green box (z-
index: 2)</div>
</div>

</body>
</html>
```

The float Property

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

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

<pre> <!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> 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 </pre>	<pre> <!DOCTYPE html> <html> <head> <style> .div1 { float: left; padding: 10px; border: 3px solid #73AD21; } .div2 { padding: 10px; border: 3px solid red; } .div3 { float: left; padding: 10px; border: 3px solid #73AD21; } .div4 { padding: 10px; border: 3px solid red; clear: left; } </style> </head> <body> </pre>
---	--

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>

JavaScript

What is JavaScript

JavaScript (js) is a light-weight object-oriented programming language which is used by several websites for scripting the webpages. It is an interpreted, full-fledged programming language that enables dynamic interactivity on websites when applied to an HTML document. It was introduced in the year 1995 for adding programs to the webpages in the Netscape Navigator browser. Since then, it has been adopted by all other graphical web browsers. With JavaScript, users can build modern web applications to interact directly without reloading the page every time. The traditional website uses js to provide several forms of interactivity and simplicity.

Although, JavaScript has no connectivity with Java programming language. The name was suggested and provided in the times when Java was gaining popularity in the market. In addition to web browsers, databases such as CouchDB and MongoDB uses JavaScript as their scripting and query language.

Features of JavaScript

There are following features of JavaScript:

1. All popular web browsers support JavaScript as they provide built-in execution environments.
2. JavaScript follows the syntax and structure of the C programming language. Thus, it is a structured programming language.
3. JavaScript is a weakly typed language, where certain types are implicitly cast (depending on the operation).
4. JavaScript is an object-oriented programming language that uses prototypes rather than using classes for inheritance.
5. It is a light-weighted and interpreted language.
6. It is a case-sensitive language.
7. JavaScript is supportable in several operating systems including, Windows, macOS, etc.
8. It provides good control to the users over the web browsers.

History of JavaScript

In 1993, Mosaic, the first popular web browser, came into existence. In the year 1994, Netscape was founded by Marc Andreessen. He realized that the web needed to become more dynamic. Thus, a 'glue language' was believed to be provided to HTML to make web designing easy for designers and part-time programmers. Consequently, in 1995, the company recruited Brendan Eich intending to implement and embed Scheme programming language to the browser. But, before Brendan could start, the company merged with Sun Microsystems for adding Java into its Navigator so that it could compete with Microsoft over the web technologies and platforms. Now, two languages were there: Java and the scripting language. Further, Netscape decided to give a similar name to the scripting language as Java's. It led to 'Javascript'. Finally, in May 1995, Marc Andreessen coined the first code of Javascript named 'Mocha'. Later, the marketing team replaced the name with 'LiveScript'. But, due to trademark reasons and certain other reasons, in December 1995, the language was finally renamed to 'JavaScript'. From then, JavaScript came into existence.

Application of JavaScript

JavaScript is used to create interactive websites. It is mainly used for:

- Client-side validation,
- Dynamic drop-down menus,
- Displaying date and time,
- Displaying pop-up windows and dialog boxes (like an alert dialog box, confirm dialog box and prompt dialog box),
- Displaying clocks etc.

JavaScript in <head> or <body>

You can place any number of scripts in an HTML document.

Scripts can be placed in the `<body>`, or in the `<head>` section of an HTML page, or in both.

JavaScript in <head>

In this example, a JavaScript `function` is placed in the `<head>` section of an HTML page.

```
<!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.

```
<!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 file

We can create external JavaScript file and embed it in many html page.

It provides code re usability because single JavaScript file can be used in several html pages.

An external JavaScript file must be saved by .js extension. It is recommended to embed all JavaScript files into a single file. It increases the speed of the webpage.

message.js

1. function msg(){
2. alert("Hello Javatpoint");
3. }

index.html

1. <html>
2. <head>
3. <script type="text/javascript" src="message.js"></script>
4. </head>
5. <body>
6. <p>Welcome to JavaScript</p>
7. <form>
8. <input type="button" value="click" onclick="msg()"/>
9. </form>
10. </body>
11. </html>

Advantages of External JavaScript

There will be following benefits if a user creates an external javascript:

1. It helps in the reusability of code in more than one HTML file.
2. It allows easy code readability.
3. It is time-efficient as web browsers cache the external js files, which further reduces the page loading time.
4. It enables both web designers and coders to work with html and js files parallelly and separately, i.e., without facing any code conflicts.
5. The length of the code reduces as only we need to specify the location of the js file.

Disadvantages of External JavaScript

There are the following disadvantages of external files:

1. The stealer may download the coder's code using the url of the js file.
2. If two js files are dependent on one another, then a failure in one file may affect the execution of the other dependent file.
3. The web browser needs to make an additional http request to get the js code.
4. A tiny to a large change in the js code may cause unexpected results in all its dependent files.
5. We need to check each file that depends on the commonly created external javascript file.
6. If it is a few lines of code, then better to implement the internal javascript code.

JavaScript Output

JavaScript can "display" data in different ways:

- Writing into an HTML element, using innerHTML.
- Writing into the HTML output using document.write().
- Writing into an alert box, using window.alert().
- Writing into the browser console, using console.log().

<pre><!DOCTYPE html> <html> <body> <h1>My First Web Page</h1> <p>My First Paragraph</p> <p id="demo"></p> <script> document.getElementById("demo").innerHTML = 5 + 6; </script> </body> </html></pre>	<pre><!DOCTYPE html> <html> <body> <h1>My First Web Page</h1> <p>My first paragraph.</p> <script> document.write(5 + 6); </script> </body> </html></pre>
<pre><!DOCTYPE html> <html> <body> <h1>My First Web Page</h1></pre>	<pre><!DOCTYPE html> <html> <body> <script></pre>

<pre><p>My first paragraph.</p> <script> window.alert(5 + 6); </script> </body> </html></pre>	<pre>console.log(5 + 6); </script> </body> </html></pre>
JavaScript Comment	

The JavaScript comments are meaningful way to deliver message. It is used to add information about the code, warnings or suggestions so that end user can easily interpret the code.

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

```
<html>
<body>
<script>
var a=10;
var b=20;
var c=a+b;//It adds values of a and b variable
document.write(c);//prints sum of 10 and 20
</script>
</body>
</html>
```

JavaScript Multi line Comment

It can be used to add single as well as multi line comments. So, it is more convenient.

```

<html>
<body>
<script>
/* It is multi line comment.
It will not be displayed */
document.write("example of javascript multiline comment");
</script>
</body>
</html>

```

JavaScript Variables

Variables are Containers for Storing Data

JavaScript Variables can be declared in 4 ways:

- Automatically
- Using var
- Using let
- Using const

<ul style="list-style-type: none"> • <!DOCTYPE html> • <html> • <body> • <h1>JavaScript Variables</h1> • • <p>In this example, x, y, and z are undeclared.</p> • <p>They are automatically declared when first used.</p> • • <p id="demo"></p> • • <script> • x = 5; • y = 6; • z = x + y; • document.getElementById("demo").innerHTML = 	<pre> <!DOCTYPE html> <html> <body> <h1>JavaScript Variables</h1> <p>In this example, x, y, and z are variables.</p> <p id="demo"></p> <script> var x = 5; var y = 6; var z = x + y; document.getElementById("demo").innerHTML = "The value of z is: " + z; </script> </pre>
---	--

<ul style="list-style-type: none"> • "The value of z is: " + z; • </script> • • </body> • </html> 	</body> </html>
<!DOCTYPE html> <html> <body> <h1>JavaScript Variables</h1> <p>In this example, x, y, and z are variables.</p> <p id="demo"></p> <script> let x = 5; let y = 6; let z = x + y; document.getElementById("demo").innerHTML = "The value of z is: " + z; </script> </body> </html>	<!DOCTYPE html> <html> <body> <h1>JavaScript Variables</h1> <p>In this example, x, y, and z are variables.</p> <p id="demo"></p> <script> const x = 5; const y = 6; const z = x + y; document.getElementById("demo").innerHTML = "The value of z is: " + z; </script> </body> </html>

JavaScript Data Types

JavaScript provides different data types to hold different types of values. There are two types of data types in JavaScript.

1. Primitive data type
2. Non-primitive (reference) data type

JavaScript is a dynamic type language; means you don't need to specify type of the variable because it is dynamically used by JavaScript engine. You need to use var here to specify the data type. It can hold any type of values such as numbers, strings etc.

For example:

1. var a=40;//holding number
2. var b="Rahul";//holding string

JavaScript primitive data types

There are five types of primitive data types in JavaScript. They are as follows:

Data Type	Description
String	represents sequence of characters e.g. "hello"
Number	represents numeric values e.g. 100
Boolean	represents boolean value either false or true
Undefined	represents undefined value
Null	represents null i.e. no value at all

JavaScript non-primitive data types

Data Type	Description
Object	represents instance through which we can access members
Array	represents group of similar values
RegExp	represents regular expression

JavaScript Operators

JavaScript operators are symbols that are used to perform operations on operands. For example:

1. `var sum=10+20;`

Here, + is the arithmetic operator and = is the assignment operator.

There are following types of operators in JavaScript.

1. Arithmetic Operators
2. Comparison (Relational) Operators
3. Bitwise Operators

4. Logical Operators
5. Assignment Operators
6. Special Operators

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic operations on the operands. The following operators are known as JavaScript arithmetic operators.

Operator	Description	Example
+	Addition	10+20 = 30
-	Subtraction	20-10 = 10
*	Multiplication	10*20 = 200
/	Division	20/10 = 2
%	Modulus (Remainder)	20%10 = 0
++	Increment	var a=10; a++; Now a = 11
--	Decrement	var a=10; a--; Now a = 9

JavaScript Comparison Operators

The JavaScript comparison operator compares the two operands. The comparison operators are as follows:

Operator	Description	Example
==	Is equal to	10==20 = false
===	Identical (equal and of same type)	10===20 = false
!=	Not equal to	10!=20 = true

!==	Not Identical	20!==20 = false
>	Greater than	20>10 = true
>=	Greater than or equal to	20>=10 = true
<	Less than	20<10 = false
<=	Less than or equal to	20<=10 = false

JavaScript Bitwise Operators

The bitwise operators perform bitwise operations on operands. The bitwise operators are as follows:

Operator	Description	Example
&	Bitwise AND	(10==20 & 20==33) = false
	Bitwise OR	(10==20 20==33) = false
^	Bitwise XOR	(10==20 ^ 20==33) = false
~	Bitwise NOT	(~10) = -10
<<	Bitwise Left Shift	(10<<2) = 40
>>	Bitwise Right Shift	(10>>2) = 2
>>>	Bitwise Right Shift with Zero	(10>>>2) = 2

JavaScript Logical Operators

The following operators are known as JavaScript logical operators.

Operator	Description	Example
----------	-------------	---------

&&	Logical AND	(10==20 && 20==33) = false
	Logical OR	(10==20 20==33) = false
!	Logical Not	!(10==20) = true

JavaScript Assignment Operators

The following operators are known as JavaScript assignment operators.

Operator	Description	Example
=	Assign	10+10 = 20
+=	Add and assign	var a=10; a+=20; Now a = 30
-=	Subtract and assign	var a=20; a-=10; Now a = 10
=	Multiply and assign	var a=10; a=20; Now a = 200
/=	Divide and assign	var a=10; a/=2; Now a = 5
%=	Modulus and assign	var a=10; a%=2; Now a = 0

JavaScript Special Operators

The following operators are known as JavaScript special operators.

Operator	Description
(?:)	Conditional Operator returns value based on the condition. It is like if-else.
,	Comma Operator allows multiple expressions to be evaluated as single statement.
delete	Delete Operator deletes a property from the object.

in	In Operator checks if object has the given property
instance of	checks if the object is an instance of given type
new	creates an instance (object)
type of	checks the type of object.
void	it discards the expression's return value.
yield	checks what is returned in a generator by the generator's iterator.

JavaScript If-else

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
}
<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 or 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  
}  
<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  
}
```

Let's see the simple example of if else if statement in javascript.

```
<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 Switch

The JavaScript switch statement is used *to execute one code from multiple expressions*. It is just like else if statement that we have learned in previous page. But it is convenient than *if..else..if* because it can be used with numbers, characters etc.

The signature of JavaScript switch statement is given below.

1. switch(expression){
2. case value1:
3. code to be executed;
4. break;
5. case value2:
6. code to be executed;
7. break;
8.
- 9.
10. default:
11. code to be executed if above values are not matched;
12. }

Let's see the simple example of switch statement in javascript.

1. <script>
2. var grade='B';
3. var result;
4. switch(grade){
5. case 'A':
6. result="A Grade";
7. break;
8. case 'B':
9. result="B Grade";
10. break;
11. case 'C':
12. result="C Grade";
13. break;
14. default:

15. result="No Grade";
16. }
17. document.write(result);
18. </script>

JavaScript Loops

The JavaScript loops are used *to iterate the piece of code* using for, while, do while or for-in loops. It makes the code compact. It is mostly used in array.

There are four types of loops in JavaScript.

1. for loop
2. while loop
3. do-while loop
4. for-in loop

1) JavaScript For loop

The JavaScript for loop *iterates the elements for the fixed number of times*. It should be used if number of iteration is known. The syntax of for loop is given below.

1. for (initialization; condition; increment)
2. {
3. code to be executed
4. }

Example

1. <script>
2. for (i=1; i<=5; i++)
3. {
4. document.write(i + "
")
5. }
6. </script>

2) JavaScript while loop

The JavaScript while loop *iterates the elements for the infinite number of times*. It should be used if number of iteration is not known. The syntax of while loop is given below.

1. while (condition)
2. {
3. code to be executed
4. }

Let's see the simple example of while loop in javascript.

1. <script>
2. var i=11;
3. while (i<=15)
4. {
5. document.write(i + "
");
6. i++;
7. }
8. </script>

3) JavaScript do while loop

The JavaScript do while loop *iterates the elements for the infinite number of times* like while loop. But, code is *executed at least* once whether condition is true or false. The syntax of do while loop is given below.

1. do{
2. code to be executed
3. }while (condition);

Let's see the simple example of do while loop in javascript.

1. <script>
2. var i=21;
3. do{
4. document.write(i + "
");

5. `i++;`
6. `}while (i<=25);`
7. `</script>`

JavaScript Functions

JavaScript functions are used to perform operations. We can call JavaScript function many times to reuse the code.

Advantage of JavaScript function

There are mainly two advantages of JavaScript functions.

1. **Code reusability:** We can call a function several times so it save coding.
2. **Less coding:** It makes our program compact. We don't need to write many lines of code each time to perform a common task.

JavaScript Function Syntax

The syntax of declaring function is given below.

1. `function functionName([arg1, arg2, ...argN]){`
2. `//code to be executed`
3. `}`

Let's see the simple example of function in JavaScript that does not has arguments.

1. `<script>`
2. `function msg(){`
3. `alert("hello! this is message");`
4. `}`
5. `</script>`
6. `<input type="button" onclick="msg()" value="call function"/>`

JavaScript Function Arguments

We can call function by passing arguments. Let's see the example of function that has one argument.

1. `<script>`

2. `function getcube(number){`
3. `alert(number*number*number);`
4. `}`
5. `</script>`
6. `<form>`
7. `<input type="button" value="click" onclick="getcube(4)"/>`
8. `</form>`

Function with Return Value

We can call function that returns a value and use it in our program. Let's see the example of function that returns value.

1. `<script>`
2. `function getInfo(){`
3. `return "hello javatpoint! How r u?";`
4. `}`
5. `</script>`
6. `<script>`
7. `document.write(getInfo());`
8. `</script>`

JavaScript Function Object

In JavaScript, the purpose of Function constructor is to create a new Function object. It executes the code globally. However, if we call the constructor directly, a function is created dynamically but in an unsecured way.

Syntax

1. `new Function ([arg1[, arg2[,argn]],] functionBody)`

Parameter

`arg1, arg2, , argn` - It represents the argument used by function.

`functionBody` - It represents the function definition.

JavaScript Function Methods

Let's see function methods with description.

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 a form of a string.

JavaScript Function Object Examples

Example 1

Let's see an example to display the sum of given numbers.

1. `<script>`
2. `var add=new Function("num1","num2","return num1+num2");`
3. `document.writeln(add(2,5));`
4. `</script>`

Example 2

Let's see an example to display the power of provided value.

1. `<script>`
2. `var pow=new Function("num1","num2","return Math.pow(num1,num2)");`
3. `document.writeln(pow(2,3));`
4. `</script>`

JavaScript Objects

A JavaScript object is an entity having state and behavior (properties and method). For example: car, pen, bike, chair, glass, keyboard, monitor etc.

JavaScript is an object-based language. Everything is an object in JavaScript.

JavaScript is template based not class based. Here, we don't create class to get the object. But, we directly create objects.

Creating Objects in JavaScript

There are 3 ways to create objects.

1. By object literal
2. By creating instance of Object directly (using new keyword)
3. By using an object constructor (using new keyword)

1) JavaScript Object by object literal

The syntax of creating object using object literal is given below:

1. `object={property1:value1,property2:value2.....propertyN:valueN}`

As you can see, property and value is separated by : (colon).

Let's see the simple example of creating object in JavaScript.

1. `<script>`
2. `emp={id:102,name:"Shyam Kumar",salary:40000}`
3. `document.write(emp.id+" "+emp.name+" "+emp.salary);`
4. `</script>`

2) By creating instance of Object

The syntax of creating object directly is given below:

1. `var objectname=new Object();`

Let's see the example of creating object directly.

1. `<script>`

2. `var emp=new Object();`
3. `emp.id=101;`
4. `emp.name="Ravi Malik";`
5. `emp.salary=50000;`
6. `document.write(emp.id+" "+emp.name+" "+emp.salary);`
7. `</script>`

Test it Now

Output of the above example

101 Ravi 50000

3) By using an Object constructor

Here, you need to create function with arguments. Each argument value can be assigned in the current object by using this keyword.

The `this` keyword refers to the current object.

The example of creating object by object constructor is given below.

1. `<script>`
2. `function emp(id,name,salary){`
3. `this.id=id;`
4. `this.name=name;`
5. `this.salary=salary;`
6. `}`
7. `e=new emp(103,"Vimal Jaiswal",30000);`

8. `document.write(e.id+" "+e.name+" "+e.salary);`
9. `</script>`

Defining method in JavaScript Object

We can define method in JavaScript object. But before defining method, we need to add property in the function with same name as method.

The example of defining method in object is given below.

1. <script>
2. function emp(id,name,salary){
3. this.id=id;
4. this.name=name;
5. this.salary=salary;
- 6.
7. this.changeSalary=changeSalary;
8. function changeSalary(otherSalary){
9. this.salary=otherSalary;
10. }
11. }
12. e=new emp(103,"Sonoo Jaiswal",30000);
13. document.write(e.id+" "+e.name+" "+e.salary);
14. e.changeSalary(45000);
15. document.write("
"+e.id+" "+e.name+" "+e.salary);
16. </script>

S.No	Methods	Description
1	Object.assign()	This method is used to copy enumerable and own properties from a source object to a target object
2	Object.create()	This method is used to create a new object with the specified prototype object and properties.

3	<code>Object.defineProperty()</code>	This method is used to describe some behavioral attributes of the property.
4	<code>Object.defineProperties()</code>	This method is used to create or configure multiple object properties.
5	<code>Object.entries()</code>	This method returns an array with arrays of the key, value pairs.
6	<code>Object.freeze()</code>	This method prevents existing properties from being removed.
7	<code>Object.getOwnPropertyDescriptor()</code>	This method returns a property descriptor for the specified property of the specified object.
8	<code>Object.getOwnPropertyDescriptors()</code>	This method returns all own property descriptors of a given object.
9	<code>Object.getOwnPropertyNames()</code>	This method returns an array of all properties (enumerable or not) found.
10	<code>Object.getOwnPropertySymbols()</code>	This method returns an array of all own symbol key properties.
11	<code>Object.getPrototypeOf()</code>	This method returns the prototype of the specified object.
12	<code>Object.is()</code>	This method determines whether two values are the same value.

13	<code>Object.isExtensible()</code>	This method determines if an object is extensible
14	<code>Object.isFrozen()</code>	This method determines if an object was frozen.
15	<code>Object.isSealed()</code>	This method determines if an object is sealed.
16	<code>Object.keys()</code>	This method returns an array of a given object's own property names.
17	<code>Object.preventExtensions()</code>	This method is used to prevent any extensions of an object.
18	<code>Object.seal()</code>	This method prevents new properties from being added and marks all existing properties as non-configurable.
19	<code>Object.setPrototypeOf()</code>	This method sets the prototype of a specified object to another object.
20	<code>Object.values()</code>	This method returns an array of values.