Web Development (Full Stack)

Date: 07-05-2023

1. **What is WEB?**

The World Wide Web, often referred to simply as "the web," is a system of interconnected with documents and resources that are accessed over the internet using a web browser (web is a subset of internet). It is a platform for sharing information and resources in a global, decentralized network.

Web is a collection of web pages, where web pages is also collection of information or documents, etc.

The key innovation of the web was the use of hyperlinks, or clickable links that allow users to navigate between pages and resources. This made it possible to create a vast network of interconnected documents that could be easily browsed and searched.

The basic building block of the web is the web page, which is a document that is written in HTML (Hypertext Markup Language), a language used to create content for the web. Web pages can contain text, images, videos, and other multimedia content, as well as interactive elements like forms and buttons.

Web pages are hosted on web servers, which are computers that store and serve web content to users who request it. When a user enters the URL of a web page into their web browser, the browser sends a request to the web server, which then sends back the requested page.

One of the most important aspects of the web is its ability to connect people and resources from all over the world. The web has made it possible to share information, collaborate on projects, and communicate with others in real-time, regardless of where they are located.

In addition to its social and collaborative aspects, the web has also had a profound impact on business and commerce. The rise of e-commerce and online marketplaces has transformed the way we shop and do business, making it possible to buy and sell goods and services from anywhere in the world.

Overall, the web is an incredibly powerful tool that has transformed the way we live, work, and communicate. It has made the world more connected and accessible, and has opened up new opportunities for collaboration, creativity, and innovation.

2. **What is Internet?**

The internet is a global network of computers and servers that are connected together and use standardized communication protocols to share information and resources. It allows users all over the world to communicate, share information, and access a wide range of services and applications.

It is networks of networks. Internet is global system where billions of systems are interconnected with each other.

The internet was developed in the 1960s as a project of the US Department of Defense. The original goal was to create a decentralized network that could withstand a nuclear attack, but it quickly evolved into a more general-purpose system for sharing information and resources.

The internet consists of a vast network of computers and servers that are connected together using a variety of technologies, including wired and wireless connections. These connections are made possible by a variety of networking protocols, including TCP/IP, HTTP, FTP, and others.

One of the key features of the internet is its ability to connect disparate systems and networks together, allowing users to communicate and share information across different platforms and devices. This has led to the development of a wide range of applications and services, including email, social media, video conferencing, online gaming, and more.

Today, billions of people around the world use the internet every day for a wide range of purposes. It has become an essential tool for communication, commerce, and entertainment, and has transformed the way we live, work, and interact with each other.

3. What is Web Development?

Web development refers to the process of creating and maintaining websites and web applications. It involves a wide range of activities, including web design, web content development, client-side scripting, server-side scripting, and database management.

Web development typically starts with the creation of a website's visual design or user interface (UI), which is then translated into a set of HTML, CSS, and JavaScript files that define the website's structure and appearance. Web designers and front-end developers work together to create these files, which are then uploaded to a web server.

Once the website's front-end has been created, back-end developers work on creating the server-side code that handles requests from users and interacts with databases to retrieve and store data. This code is typically written in programming languages like PHP, Ruby, Python, or Java.

Database management is another important aspect of web development. Web developers use databases to store and manage data related to a website, such as user account information, product listings, and other content.

Overall, web development is a complex and constantly evolving field that requires a wide range of skills and expertise. As the web continues to grow and evolve, web developers will continue to play a critical role in shaping the way we interact with information and each other online.

4. What is Web Site & Web App?

A website is a collection of web pages that are hosted on a web server and can be accessed over the internet using a web browser. Websites typically provide information or services to users, such as news articles, product listings, or online shopping. Websites can be static, with content that does not change very often, or dynamic, with content that is updated frequently.

A web application (or web app) is a software program that runs in a web browser and provides interactive functionality to users. Web apps are typically more complex than simple websites and often require a server-side component to handle user requests and data storage. Examples of web applications include social media sites, online banking systems, and e-commerce platforms.

The main difference between a website and a web app is their level of interactivity and functionality. While a website typically provides information or static content to users, a web app allows users to interact with the content and perform tasks, such as making a purchase or filling out a form. Web apps may also be more personalized, allowing users to customize their experience based on their preferences or behaviour.

Overall, websites and web apps both play important roles in the online world, and the choice between the two depends on the specific needs of the user or business. While a website may be sufficient for providing information or basic functionality, a web app may be necessary for more complex or interactive applications.

5. **What is Front-End & Back-End?**

Front-end and back-end are two terms used in web development to describe different parts of a web application or website.

Front-end (or client-side) refers to the part of a web application or website that users interact with directly. This includes the visual elements of a website, such as the layout, colours, and typography, as well as the interactive components, such as buttons, forms, and menus. Front-end developers use languages such as HTML, CSS, and JavaScript to create the user interface and experience.

Back-end (or server-side) refers to the part of a web application or website that users do not directly interact with, but that provides functionality and data storage for the front-end. This includes the server-side code, databases, and APIs (Application Programming Interfaces) that handle user requests and data management. Back-end developers use languages such as Python, PHP, Ruby, and Java to create the server-side functionality.

In simple terms, front-end developers create the visual and interactive elements that users see and interact with on a website, while back-end developers create the underlying systems and logic that handle user requests, store data, and communicate with external services.

It's worth noting that front-end and back-end development are often closely linked and may overlap in some areas, such as creating APIs or integrating third-party services into a website. Many web developers work on both front-end and back-end development, while others specialize in one area or the other.

6. **Architecture of WebApp.**

The architecture of a web application refers to the way in which its various components are structured and interact with each other to provide its functionality. While there are many different approaches to web application architecture, some common patterns and components include:

- **Client-side:** This is the part of the web application that runs on the user's device, typically in a web browser. It consists of HTML, CSS, and JavaScript code that creates the user interface and handles user interactions.
- **Server-side:** This is the part of the web application that runs on the web server, typically in a programming language like PHP, Python, Ruby, or Java. It handles user requests, interacts with databases, and performs other server-side tasks.
- **Database:** This is the component of the web application that stores and manages data used by the application. This can include user account information, product listings, and other content.
- **Application programming interfaces (APIs):** These are sets of protocols and tools for building and integrating web applications. APIs allow different parts of the application to communicate with each other and with external services.
- **Middleware:** This is software that connects different parts of the web application and provides additional functionality, such as authentication, logging, or caching.

- **Content management system (CMS):** A CMS is a software application that allows users to create, manage, and publish digital content, often used for websites and web applications that require frequent updates.

- **Cloud infrastructure:** Cloud computing services provide scalable, on-demand access to resources like computing power, storage, and databases, making it easier to deploy and scale web applications.

Overall, the architecture of a web application can vary depending on its specific requirements and the technologies used. However, regardless of the specific approach, the goal of web application architecture is to create a scalable, secure, and maintainable application that provides a high-quality user experience.

7. What is Network? Different types of Networks.

A network is a collection of interconnected devices, such as computers, servers, routers, switches, and other devices, that can communicate with each other and share resources. Networks can be local, connecting devices within a specific physical location, such as a home or office, or global, connecting devices across the world.

A computer network is a group of computers linked to each other that enables the computer to communicate with another computer and share their resources, data, and applications.

Networks are used for a variety of purposes, including:

- **Communication:** Networks allow people to communicate with each other using email, instant messaging, voice over IP (VoIP), and other technologies.

- **Resource sharing:** Networks allow devices to share resources, such as printers, scanners, and storage devices, which can be accessed by multiple users.

- **Information sharing:** Networks allow people to share information and collaborate on projects, such as shared files, documents, and databases.

- **Internet access:** Networks provide access to the internet, which allows users to access websites, online services, and other resources.

Different Types of Networks:

1. **Local area network (LAN):** A LAN is a network that connects devices within a specific physical location, such as a home, office, or school.

2. **Metropolitan Area Network (MAN):** MAN is a network that connects devices across a larger geographical area, such as across cities or states etc.

3. **Wide area network (WAN):** A WAN is a network that connects devices across a larger geographical area, such as across countries, zones, etc.

4. **Wireless network:** A wireless network uses radio waves to connect devices without the need for physical cables.

5. **Virtual private network (VPN):** A VPN is a secure network that allows users to connect to a private network over the internet.

6. **Personal Area Network (PAN):** It is used for connecting the computer devices for personal use.

https://www.javatpoint.com/types-of-computer-network

8. IP address.

An IP address, or Internet Protocol address, is a unique numerical identifier assigned to every device that is connected to a network that uses the Internet Protocol for communication. IP addresses are used to identify and communicate with devices on a network, allowing them to send and receive data.

IP addresses can be assigned statically, meaning they are manually assigned by a network administrator, or dynamically, meaning they are automatically assigned by a server using a protocol such as DHCP.

IP addresses are used to facilitate communication between devices on a network and are essential for internet communication. Every device that connects to the internet has a unique IP address that is used to identify and communicate with that device.

9. What is Browser & Browser Engine?

Browser:

A browser short for web browser, is a software application that is used to access and view web pages on the internet. Browsers allow users to enter a web address or search term and retrieve web pages from servers located anywhere in the world.

Browsers render web pages, which means they interpret and display the code used to create web pages, such as HTML, CSS, and JavaScript. Browsers display text, images, video, and other content on the web page, and provide features such as navigation buttons, address bars, bookmarks, and extensions to enhance the browsing experience.

Some popular web browsers include:

- Google Chrome: a widely used browser developed by Google.
- Mozilla Firefox: an open-source browser that is highly customizable.
- Apple Safari: a browser developed by Apple, primarily used on Apple devices.
- Microsoft Edge: a browser developed by Microsoft, designed to replace the older Internet Explorer browser.
- Opera: a browser known for its speed and advanced features, such as built-in VPN and ad blocker.

Overall, browsers play a critical role in accessing and interacting with web pages on the internet, allowing users to explore, learn, and connect with people and information from around the world.

Browser Engine:

A browser engine, also known as a rendering engine or layout engine, is the software component in a web browser that is responsible for rendering web pages by interpreting and displaying the code used to create them, such as HTML, CSS, and JavaScript. The engine reads the code and translates it into a visual representation that can be displayed on the screen.

Different web browsers use different rendering engines to display web pages. **For example**, Google Chrome and Microsoft Edge both use the Blink rendering engine, which is based on the WebKit engine

originally developed by Apple for its Safari browser. Mozilla Firefox uses the Gecko rendering engine, and Opera uses the Blink engine as well.

The browser engine is one of the key components of a web browser, and its efficiency and performance can significantly impact the browsing experience for users. Developers can optimize their code to work efficiently with a particular rendering engine, and different engines may handle certain features or elements of web pages differently, which can affect how they are displayed to users.

10. What are Client and Server? How they work. How many requests present in client and server.

In computer networking, a client is a computer program or device that requests services or resources from another computer program or device, called a server. The client and server work together to exchange data over a network.

When a client sends a request to a server, the server responds with the requested data. This exchange of data is called a request-response cycle. Clients and servers can communicate using different protocols, such as the Hypertext Transfer Protocol (HTTP) used for the web, File Transfer Protocol (FTP) used for file transfers, and Simple Mail Transfer Protocol (SMTP) used for email.

In the context of the web, a web browser is a client that sends requests to web servers, which respond with web pages that are displayed in the browser. The browser sends a request to the server for a particular web page, and the server responds with the HTML, CSS, and JavaScript code needed to display that page. The browser then renders the code to display the web page to the user.

Request:

- **GET:** this method helps to retrieves the information from given server.
- **PUT:** this method helps to replace the current representation of target resources.
- **POST:** this method helps to send the data to server (hiddenly).
- **DELETE:** this method helps to delete all the current representation of target resources.
- **CONNECT:** this method helps to establish a tunnel to the server, which is identified by a given URL.
- **OPTIONS:** this method helps to describe the options of communication of the target resource.

11. What is Domain? What is DNS (domain name system)? How it is work.

In the context of the internet, a domain is a unique name that is used to identify a website or a group of devices connected to the internet. A domain name is part of a larger system called the Domain Name System (DNS), which translates domain names into IP addresses that computers use to locate and communicate with each other on the internet. For example, in the domain name "google.com", "google" is the SLD and ".com" is the TLD.

Domain names are important because they provide a human-readable way to access websites and other resources on the internet, rather than having to remember a series of numbers (IP addresses). They are also used to establish an online identity and brand, and can be registered and owned by individuals, organizations, or businesses.

Domain Name System (DNS):

The DNS convert/turns down the domain names into IP addresses, which browsers use to load internet pages. Every device connected to the internet has its own IP address, which is used by others devices to locate the device.

A DNS server is a computer with a database containing the public IP addresses associate with the name of the websites an IP address brings a user to. DNS act like a phonebook of internet. Whenever people type the domain names, like www.google.com, www.facebook.com into the address bar of the web browser, the DSN finds the right IP address.

Once the DNS server finds the correct IP address, browsers take the address and use it to send data to content delivery network (CDN) edge servers or origin servers. Once this is done, the information on the website can be accessed by the users.

Date: 08-05-2023

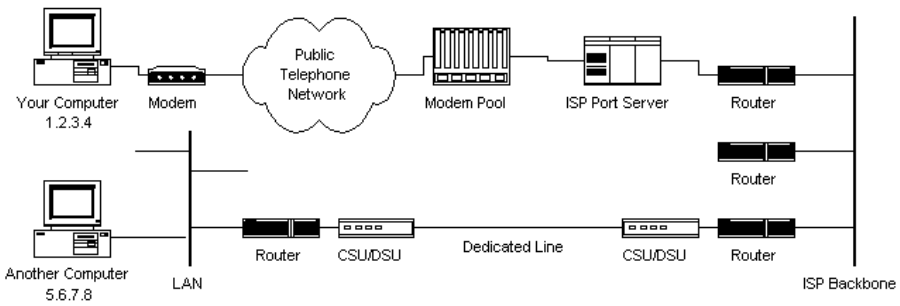
12. How internet works?

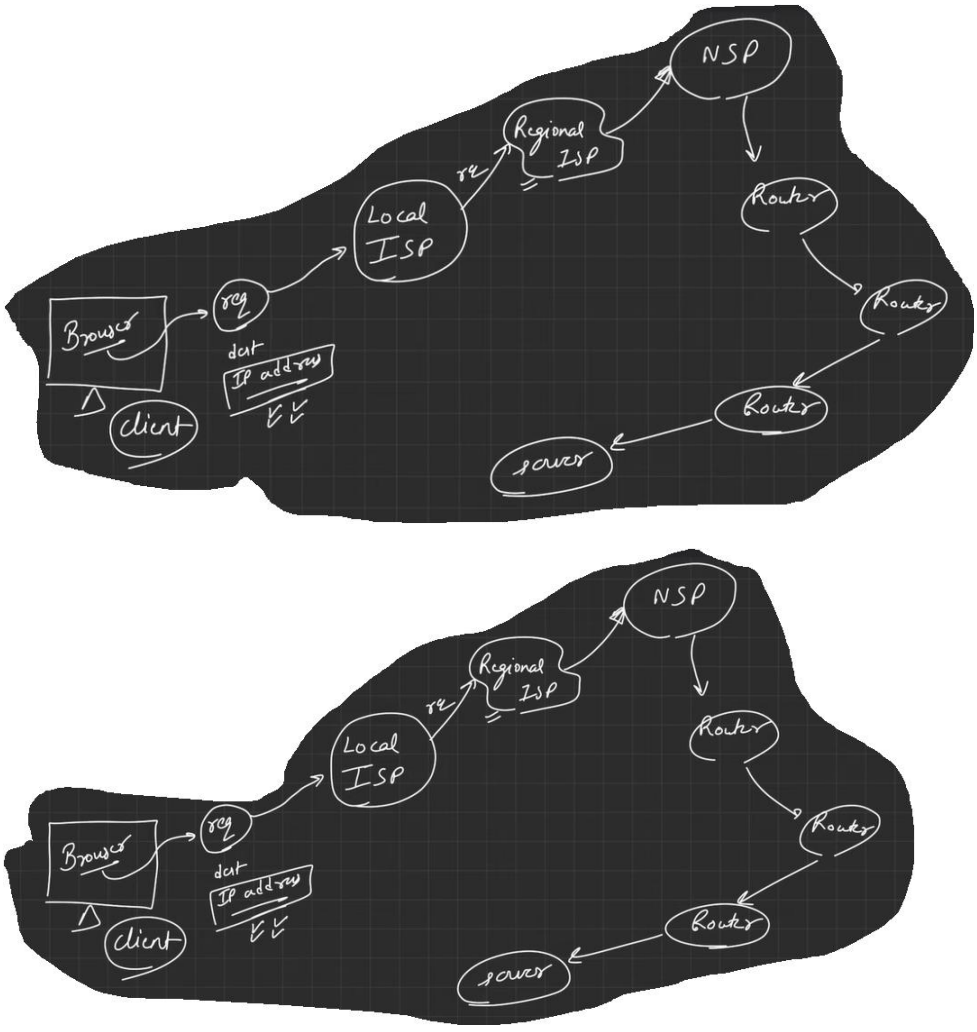
The internet is a vast network of interconnected computer networks that allows communication and data exchange between computers and devices around the world. It works by using a set of standardized protocols and technologies that enable devices to communicate with each other and share information.

Here's a simplified overview of how the internet works:

- Devices connected to the internet, such as computers, smartphones, and servers, send and receive data in the form of packets.
- These packets are sent over the internet using various types of connections, such as wired or wireless connections.
- The packets travel through various routers and switches, which direct them towards their destination.
- When a packet reaches its destination, it is reassembled and delivered to the intended device.
- To access resources on the internet, such as websites, devices use protocols such as HTTP or HTTPS to request and receive data from servers.
- Domain names are used to identify websites and other resources on the internet. These names are translated into IP addresses by the Domain Name System (DNS).
- The internet is maintained and managed by organizations such as Internet Service Providers (ISPs), network operators, and standards bodies, who work together to ensure the stability and security of the network.

The internet is constantly evolving, with new technologies and protocols being developed to improve its performance and capabilities. However, the basic principles of data exchange and communication remain the same, and the internet continues to be an essential part of modern society and the global economy.





13. What ISP, NSP, NAP?

ISP, NSP, and NAP are all related to the infrastructure of the internet and are important components of the network.

1. **ISP:** An Internet Service Provider (ISP) is a company that provides internet access to customers, usually through a wired or wireless connection. ISPs provide a range of services, including broadband internet, email, web hosting, and virtual private networks (VPNs). Examples of ISPs include Comcast, AT&T, and Verizon.

2. **NSP:** A Network Service Provider (NSP) is a company that provides internet connectivity and network services to other companies, ISPs, or organizations. NSPs operate high-speed backbone networks that connect different regions and countries around the world, and they provide the infrastructure for ISPs to connect to the internet. Examples of NSPs include Level 3 Communications, AT&T, and Sprint.

3. **NAP:** A Network Access Point (NAP) is a location where multiple NSPs connect their networks to exchange traffic and share resources. NAPs are typically large data centres with high-speed connections to the internet backbone, and they enable efficient and reliable communication between different networks. NAPs were originally developed in the early days of the internet as a way to facilitate network interconnection and are still used today, although many have been replaced by Internet Exchange Points (IXPs).

In summary, ISPs provide internet access to consumers, NSPs provide connectivity and network services to other companies and ISPs, and NAPs are locations where NSPs connect their networks to exchange traffic and share resources.

14. What is Protocol stack? HTTP, FTP, SMTP, etc.

A protocol stack, also known as a network stack, is a set of protocols and standards that work together to provide network communication functions. The protocols in a protocol stack are organized in layers, with each layer responsible for a specific aspect of network communication.

The most commonly used protocol stack is the OSI (Open Systems Interconnection) model, which defines a standard framework for network communication. The OSI model consists of seven layers, each of which represents a different aspect of network communication:

1. **Physical Layer:** This layer defines the physical characteristics of the network, such as the electrical and mechanical specifications of the cables and connectors.
2. **Data Link Layer:** This layer provides reliable communication over the physical layer by detecting and correcting errors in data transmission.
3. **Network Layer:** This layer provides routing and addressing functions, allowing data to be delivered between different networks.
4. **Transport Layer:** This layer provides end-to-end communication between applications, ensuring that data is delivered reliably and in the correct order.
5. **Session Layer:** This layer manages the communication sessions between applications, establishing and terminating connections.
6. **Presentation Layer:** This layer translates data between the application format and the network format, ensuring that data is presented in a standardized way.
7. **Application Layer:** This layer provides the interface between the network and the user, allowing applications to access network resources and services.

Each layer of the protocol stack communicates with the corresponding layer on the destination device, allowing data to be transmitted and received across the network. The use of a standardized protocol stack enables interoperability between different devices and networks, making it possible for devices from different manufacturers to communicate with each other.

15. Explain different types of networking devices.

There are several different types of networking devices, each with a specific function in a network. Here are some of the most common types of networking devices:

1. **Switches:** Switches are used to connect multiple devices on a network, allowing them to communicate with each other. They use MAC addresses to identify devices and direct data to the correct destination.
2. **Routers:** Routers are used to connect multiple networks together, allowing devices on different networks to communicate with each other. They use IP addresses to identify networks and direct data to the correct destination.

3. **Firewalls:** Firewalls are used to protect a network from unauthorized access and to monitor network traffic. They can be hardware or software-based and can block incoming and outgoing traffic based on specified rules.

4. **Hubs:** Hubs are similar to switches, but they are not as intelligent. They simply receive incoming data and broadcast it to all devices on the network.

5. **Modems:** Modems are used to convert digital signals into analog signals for transmission over telephone lines. They are used to connect to the internet through a telephone or cable line.

6. **Network interface cards (NICs):** NICs are used to connect devices to a network. They provide a physical connection to the network and allow devices to communicate with each other.

7. **Access points:** Access points are used to connect wireless devices to a network. They provide a wireless signal for devices to connect to and allow wireless communication between devices on the network.

8. **Repeaters:** Repeaters are used to amplify and regenerate signals to extend the range of a network. They receive signals and retransmit them at a higher power level to increase the distance that the signal can travel.

16. What is 1.2.3.4 syntax in IP address? what is 1, 2, 3, 4?

The 1.2.3.4 syntax is a representation of an IPv4 address, which is a 32-bit address used to identify devices on a network. The IPv4 address is typically represented in dotted decimal notation, with each of the four octets separated by a period.

In the example 1.2.3.4, the number 1 represents the first octet, the number 2 represents the second octet, the number 3 represents the third octet, and the number 4 represents the fourth octet. Each octet is a decimal number between 0 and 255, representing a portion of the 32-bit address.

The IP address is used to identify a specific device on a network and enable communication between devices. When a device sends a packet of data, it includes its own IP address as the source address and the IP address of the destination device as the destination address. The routers in the network use the IP address to route the packet to its intended destination.

IPv4 addresses are being gradually replaced by IPv6 addresses, which use a different syntax and are 128-bit addresses. The IPv6 address is represented in hexadecimal notation and is much larger than the IPv4 address space, allowing for a much larger number of devices to be connected to the internet.

17. What is IPv4. Why IPv6 is need.

IPv4 (Internet Protocol version 4) is a protocol used to assign unique addresses to devices on a network, allowing them to communicate with other devices on the internet. IPv4 uses 32-bit addresses, which are represented as a series of four decimal numbers separated by dots, for example, 192.168.1.1.

IPv6 (Internet Protocol version 6) is the successor to IPv4. It was developed to address the limitations of IPv4, which include a limited number of available addresses and a lack of support for new features required by modern networking technologies. IPv6 uses 128-bit addresses, which are represented as a series of eight groups of hexadecimal digits separated by colons, for example, 2001:0db8:85a3:0000:0000:8a2e:0370:7334.

The primary advantage of IPv6 is its larger address space, which provides a virtually unlimited number of unique addresses, allowing for the growth of the internet and the increasing number of devices that

require internet connectivity. IPv6 also includes several new features and improvements over IPv4, including better support for quality of service (QoS), improved security, and simplified network configuration.

While IPv6 has been around for many years, it is still in the process of being adopted by internet service providers (ISPs) and network operators. This is because transitioning from IPv4 to IPv6 can be complex and expensive, requiring upgrades to networking equipment and changes to network configurations. However, as the demand for internet connectivity continues to grow and the limitations of IPv4 become more apparent, the transition to IPv6 is expected to accelerate in the coming years.

18. **What is OSI & TCP model.**

    (Pass: got to note)

Day: 10-05-2023

Follow: MDN Website W3school Website

What is HTML, CSS, JavaScript?

HTML:

HTML stands for HyperText Markup Language. It is the standard markup language used to create web pages and applications that can be displayed in a web browser. HTML uses a combination of tags and attributes to define the structure, content, and layout of a web page.

It helps to labelling the website/ webpage, like which one is paragraph, image, header, footer, container, list etc. It helps to define structure of your website or webpage, it helps to provide the description like this one is heading, footer, container, list, etc.

HTML tags are used to markup different elements on a web page, such as headings, paragraphs, images, links, and lists. Each tag is enclosed in angle brackets, for example, <html>, <head>, <body>, <h1>, <p>, <img>, <a>, and so on.

Attributes provide additional information about an HTML element and are added to the opening tag using the format attribute="value". For example, the <img> tag might include attributes such as src="image.jpg" to specify the image file to be displayed, and alt="description" to provide a textual description of the image.

By combining HTML tags and attributes, web developers can create rich and interactive web pages that incorporate text, images, video, audio, and other multimedia content. The HTML code is then interpreted by web browsers to render the web page in a visually appealing and interactive way.

CSS:

CSS stands for Cascading Style Sheets. It is a language used for describing the presentation of web pages, including their layout, typography, color schemes, and other visual aspects. CSS is used in conjunction with HTML and JavaScript to create visually appealing and responsive web pages.

CSS works by defining styles for HTML elements using selectors and declarations. A selector is used to target one or more HTML elements, and a declaration specifies the style rules to be applied to those elements. For example, the following CSS rule sets the font size and color for all paragraph elements:

```
p{
        font-size: 20px;
        color: red;
}
```

CSS supports a wide range of styles and effects, including:

- Font styles: such as size, weight, color, and family
- Layout styles: such as padding, margin, and positioning
- Background styles: such as color, image, and gradient
- Border styles: such as width, style, and color
- Animation and transition effects: such as fade in/out, slide, and rotate

By separating the content and presentation of a web page, CSS makes it easier to create consistent and professional-looking websites. CSS also makes it easier to modify the visual style of a web page, since changes can be made to the CSS code without affecting the underlying HTML structure.

JavaScript:

JavaScript is a high-level programming language used to create interactive and dynamic web pages. It is often used in conjunction with HTML and CSS to create rich and engaging user experiences on the web.

JavaScript is a client-side scripting language, which means that it runs on the user's web browser rather than on a server. It is primarily used for front-end web development, such as creating animations, validating user input, and manipulating the DOM (Document Object Model) to dynamically update the content of a web page.

JavaScript is also used for back-end web development, using platforms such as Node.js to build server-side applications and APIs.

Some of the key features of JavaScript include:

- **Dynamic typing:** variables do not need to be declared with a specific data type.
- **Object-oriented programming:** JavaScript supports objects and classes, making it easier to organize and structure code.
- **Event-driven programming:** JavaScript can be used to respond to user events, such as clicking a button or scrolling a page.
- **Functional programming:** JavaScript supports functional programming techniques, such as higher-order functions and closures.

JavaScript is a versatile and widely-used language that is essential for modern web development. Its popularity and flexibility have led to the creation of many popular frameworks and libraries, such as React, Angular, and Vue.js, that simplify and streamline web development.

What is tag (opening tag & closing tag), elements (root & parament element), attributes?

Tag:

In HTML, a tag is a markup element that indicates or define the structure and content of a web page. A tag consists of an opening tag, which is used to begin the element, and a closing tag, which is used to end the element.

The opening tag is denoted by the tag name inside angle brackets, like this: **<tagname>**. The closing tag is denoted by the same tag name preceded by a forward slash, like this: **</tagname>**. For example, the following code creates a paragraph element with the opening tag ** ** and the closing tag ** **

HTML tags can be nested inside one another, which allows for complex page layouts and structures. In nested tags, the closing tag for each element should be placed in the reverse order of the opening tags.

Elements:

In HTML, elements are the building blocks of a web page, represented by tags and their contents. Each element has a starting tag, an ending tag, and content in between.

The root element of an HTML document is the **html** element. All other elements are nested inside the **html** element, and the **html** element itself is nested inside the **doctype** declaration at the beginning of the document.

The **html** element contains two child elements: the **head** element and the **body** element. The **head** element contains metadata about the page, such as the title and any links to external stylesheets or

scripts. The **body** element contains the visible content of the page, including text, images, and other HTML elements.

The parent element of an element is the element that encloses it. For example, in the following code, the **p** element is a child element of the **div** element and a sibling element of the **h1** element:

```
<div>
        <h1> Headings 1 </h1>
        <p> Paragraph of text </p>
</div>
```

In this example, the **div** element is the parent element of both the **h1** and **p** elements. The **div** element serves as the container for these child elements and provides a way to group them together as a unit within the page layout.

Attributes:

In HTML, attributes provide additional information about an element, such as its behaviour or appearance. Attributes are specified within the opening tag of an element, using the following syntax:

```
<tagname attribute1='value1' attribute2='value2'>
```

The **attribute1** and **attribute2** parts represent the attribute names, while the **value1** and **value2** parts represent their respective values. Attributes and their values are separated by an equal sign (**=**), and multiple attributes can be specified for a single element, separated by spaces.

For example, the **href** attribute of the **a** element specifies the URL that the hyperlink should point to:

```
<a href='https://example.com'> Example Website</a>
```

Similarly, the **src** attribute of the **img** element specifies the URL of the image file to display:

```
<img src='example.png' alt='img1'>
```

In these examples, **href** and **src** are the attribute names, and **"https://example.com"** and **"example.jpg"** are the attribute values.

Attributes can also be used to add styling to an element using CSS, or to specify the behaviour of a form element using JavaScript. Different HTML elements have different sets of attributes that are applicable to them, depending on their intended use and functionality.

Tell me basic structure of HTML5 (what is DOCTYPE, html, head, body tag and it's use).

- **Document Type Declaration:** This is the first line of the document and it tells the browser that the document is an HTML5 document. It looks like this:

```
<! DOCTYPE html>
```

- **HTML Tag:** This is the root element of the HTML5 document and it wraps around all other elements in the document. It looks like this:

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

- **Head Tag:** This contains metadata about the document such as the document title, author, and links to CSS stylesheets. It looks like this:

```
<head>
        <title> Document Title </title>
        <meta charset="UTF-8">
        <link rel='stylesheet' href='style.css'>
</head>
```

- **Body Tag:** This contains the content of the document such as text, images, videos, and other elements. It looks like this:

```
<body>
        <h1> Heading</h1>
        <p>Paragraph</p>
        <img src='img.png' alt='img1'>
</body>
```

Here's an example of a complete HTML5 document:

```
<! DOCTYPE html>
<html>
        <head>
                <title> Document Title </title>
                <meta charset=" UTF-8">
                <link rel='stylesheet' href='style.css'>
        </head>

        <body>
                <h1> Heading</h1>
                <p>Paragraph</p>
                <img src='img.png' alt='img1'>
        </body>

</html>
```

Day: 12-05-2023

What is Semantic, Structure & Formatting tags:

Tag basically means, that give you some kind of markup or some kind of definition about your contents.

- **Semantic tags:** These tags are used to describe the content that they surround, rather than how the content should look. They give meaning to the content, making it more understandable to both humans and machines. Some examples of semantic tags are: ** **, ****, **<ul>**, **<ol>**, **<li>**, **<table>**, **<tr>**, **<td>**, and **<th>**.

- **Structural tags:** These tags are used to define the structure of the document, such as the main sections and sub-sections. They don't necessarily give meaning to the content, but they help organize it. Some examples of structural tags are: **<header>**, **<nav>**, **<main>**, **<section>**, **<article>**, **<footer>**, and **<aside>**.

- **Formatting tags:** These tags are used to specify how the content should look, such as its font, color, and size. They are mainly used for styling purposes and should be used sparingly, as they can be replaced by CSS. Some examples of formatting tags are: **<b>**, **<i>**, **<u>**, **<strike>**, **<sub>**, **<sup>**, **<font>**, ** **, and **
**.

Can I use multiple h1 tag in a webpage?

Technically, you can use multiple **<h1>** tags in a webpage, but it is generally not recommended for best practices in web development and for the purpose of search engine optimization (SEO).

The **<h1>** tag is typically used to indicate the main heading of the page, and it should be used only once per page. The use of multiple **<h1>** tags can create confusion for search engines and screen readers, as they may not know which heading is the most important on the page.

Instead, you can use other semantic heading tags like **<h2>**, **<h3>**, **<h4>**, **<h5>**, and **<h6>** to structure the content of your page hierarchically, with **<h2>** indicating the secondary heading, **<h3>** indicating the third level heading, and so on. This helps search engines and screen readers understand the hierarchy and organization of your content.

How to send email, go to website, call a number using <a> tag?

 Send Email

 Go to YouTube

 Call me

What is Ordered list, Unordered list, Description list?

Ordered list: An ordered list is a list of items that are ordered in a numerical or alphabetical sequence. The items in an ordered list are typically numbered, and each item is marked with the **<li>** (list item) tag. The **<ol>** (ordered list) tag is used to indicate that a list is an ordered list. For example:

```
<ol>
        <li> Item 1 </li>
        <li> Item 2 </li>
        <li> Item 3 </li>
</ol>
```

Unordered list: An unordered list is a list of items that are not ordered in a sequence. The items in an unordered list are typically marked with a bullet point or other symbol, and each item is marked with the **<li>** tag. The **<ul>** (unordered list) tag is used to indicate that a list is an unordered list. For example:

```
<ul>
        <li> Dibyendu </li>
        <li> Siliguri </li>
        <li> Biswas </li>
</ul>
```

Description list: A description list is a list of terms and their corresponding definitions. The terms are typically marked with the **<dt>** (description term) tag, and the definitions are marked with the **<dd>** (description definition) tag. The **<dl>** (description list) tag is used to indicate that a list is a description list. For example:

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

In each case, the content of each list item is placed between the opening and closing **<li>**, **<dt>**, or **<dd>** tags, and the entire list is wrapped in the corresponding **<ol>**, **<ul>**, or **<dl>** tags.

Create a bookmark so that you can click this bookmark & reached in different segments.

Why we are generally not used <b>, <i> tag.

The **<b>** and **<i>** tags are often not used in modern HTML because they are considered to be presentational markup (define the visual appearance or style of content on a web page) rather than semantic markup (structure of the content on a web page). In other words, they are used to indicate how the text should be displayed (bold or italic), rather than what the text means or represents.

Instead, the recommended approach is to use CSS (Cascading Style Sheets) to style text for presentation. CSS provides a way to separate the presentation of a web page from its content, which allows for greater flexibility and consistency across different devices and screen sizes.

In terms of semantic markup, the **** and **** tags are often preferred over **<b>** and **<i>**, respectively. The **** tag is used to indicate that text is of strong importance, while the **** tag is used to indicate emphasis.

What is the used and importance of tag, tag and <i> tag, <u> tag, <s> tag, <sup> tag, <sub> tag, <pre> tag, <code> tag.

- **** tag: This tag is used to indicate that a portion of text should be given importance, often rendered as bold text. It's important to note that the **** tag is used to indicate the importance of the content, rather than its visual appearance.
- **** tag: This tag is used to indicate that a portion of text should be emphasized, often rendered as italicized text. Like the **** tag, the **** tag is used to indicate the meaning of the content rather than its visual appearance.
- **<i>** tag: This tag is used to indicate that a portion of text should be italicized, typically used for book titles, foreign words, or other instances where italics are used for emphasis rather than meaning.
- **<u>** tag: This tag is used to underline a portion of text. However, it's generally not recommended to use the **<u>** tag as it can be confusing to users who may mistake underlined text for a hyperlink.
- **<s>** tag: This tag is used to strike through a portion of text, typically used to indicate that content has been deleted or is no longer relevant.

- **<sup>** tag: This tag is used to create superscript text, typically used for mathematical equations, footnotes, or abbreviations.
- **<sub>** tag: This tag is used to create subscript text, typically used for chemical formulas or mathematical equations.
- **<pre>** tag: This tag is used to preserve whitespace and formatting within a block of text, typically used for displaying code snippets or other text with a fixed-width font.
- **<code>** tag: This tag is used to indicate a portion of text that contains code or programming language, typically rendered in a fixed-width font and with syntax highlighting to make it easier to read and understand.

Overall, these tags are important for providing meaning and context to the content on a web page, making it more accessible to users and search engines, and improving the overall readability and structure of the page.

Structural tags: header, nav, footer, main, article, aside, section tags.

In HTML5, there are several structural tags that are used to organize and structure the content of a webpage. These tags provide semantic meaning to the elements they contain, making it easier for search engines and other web technologies to understand the content of the page. Here is a brief explanation of each of these tags:

- **<header>**: This tag is used to define the header section of a webpage. Typically, it contains the site logo, navigation links, and other important information about the site.
- **<nav>**: This tag is used to define the navigation section of a webpage. It typically contains links to other pages on the site.
- **<footer>**: This tag is used to define the footer section of a webpage. It typically contains copyright information, contact details, and other important information about the site.
- **<main>**: This tag is used to define the main content area of a webpage. It should contain the primary content of the page.
- **<article>**: This tag is used to define a self-contained block of content, such as a blog post or news article.
- **<aside>**: This tag is used to define content that is related to the main content of the page, but not directly part of it. Typically, it contains sidebars, related links, or other supplementary content.
- **<section>**: This tag is used to define a section of a webpage. It can be used to group related content together.

These tags help to improve the structure of the webpage, making it easier to read and understand. They also provide important semantic meaning to the content, making it more accessible to search engines and other web technologies.

What is the difference between <article> tag and <section> tag.

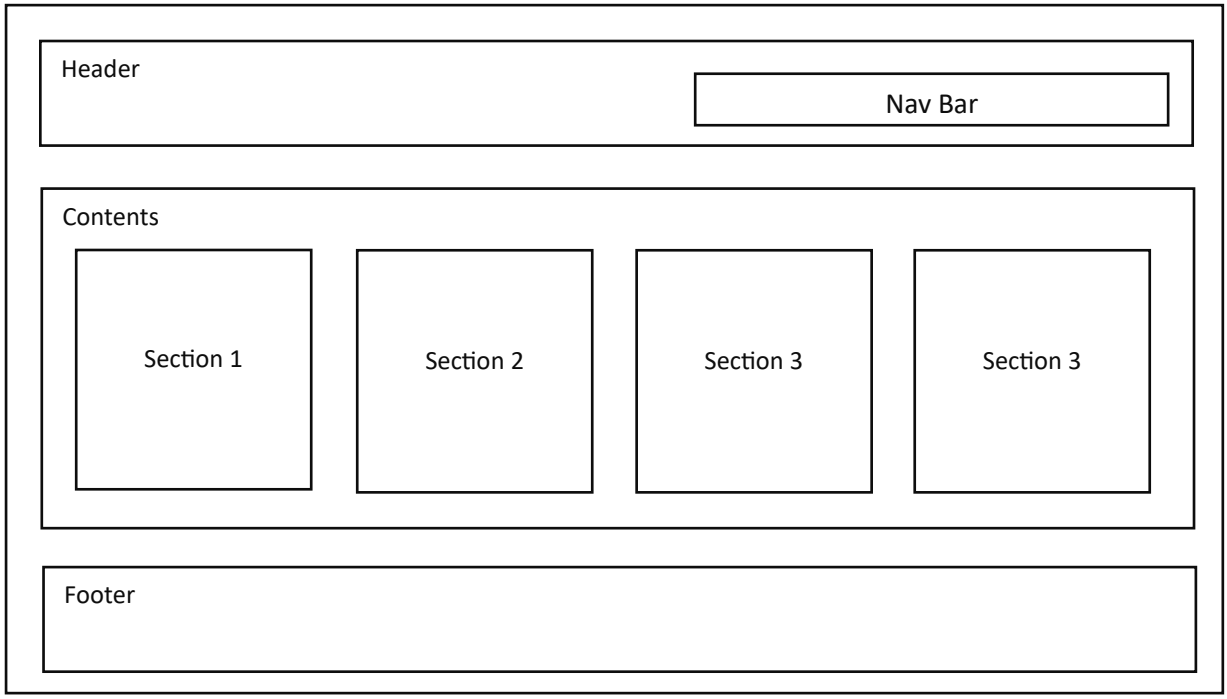
The **<article>** and **<section>** tags are both used for structuring content on a web page, but they have different meanings and use cases.

The **<article>** tag is used to define a self-contained piece of content that can be distributed and reused independently from the rest of the page. It typically represents a blog post, news article, forum post, or any other content that can stand alone. It can also include sub-sections with their own headings, images, and other elements. An **<article>** tag is often used to markup blog posts, news articles, and product descriptions.

The **<section>** tag is used to group related content together within a document, but it does not imply any semantic meaning or self-contained structure. It can be used to divide a long article into sections, or to group a set of related links or items. The content inside a **<section>** tag can include any type of HTML element, including headings, paragraphs, lists, and other sections. An **<section>** tag can be used to markup different sections of a web page such as the header, main content, and footer.

In summary, the main difference between the two tags is that **<article>** is used to mark-up self-contained content that can stand alone, while **<section>** is used to group related content together within a document.

Simple Structure of Web Page



What is Emmet & Emmet in VS-Code.

Emmet is a coding abbreviation tool that allows developers to write HTML and CSS code quickly and efficiently. It uses abbreviations and shortcuts to generate code snippets that can be expanded into complete HTML or CSS code blocks. Emmet can be used with a variety of code editors and IDEs, including Visual Studio Code, Sublime Text, and Atom, among others. Emmet can save developers significant amounts of time by allowing them to create complex code structures with a few simple keystrokes, making it an essential tool for web developers.

[got to code: Day 02 (12 May)]

What is “id” and “class” attribute?

Both **id** and **class** are HTML attributes used to apply styles and scripting to specific elements on a web page.

The **id** attribute is used to give an element a unique identifier. This identifier is used to target and manipulate the element using JavaScript and CSS. The **id** attribute value must be unique within the entire HTML document, and it should not contain any spaces.

The **class** attribute, on the other hand, is used to group elements that share common characteristics. Multiple elements can have the same class attribute value, and elements can have more than one class value. This allows developers to easily apply styles and scripting to multiple elements at once. The **class** attribute value can contain spaces, and multiple classes are separated by spaces.

What are Elements grouping with example.

Elements grouping in HTML refers to grouping multiple HTML elements/ tags together for applying common styling or functionality to them.

- **Using the div element:** The **div** element is used to create a generic container for other HTML elements. Multiple elements can be placed inside a **div** element, and styles or functionality can be applied to the **div** element and its children. For example:

```
<div>
        <h1> Headline 1 </h1>
        <p> paragraph 1 </p>
        <button> Click Me </button>
</div>
```

- **Using the ul and li elements:** The **ul** (unordered list) and **li** (list item) elements are used to create lists of items. Multiple list items can be grouped together using the **ul** element. For example:

```
<ul>
        <li> MAbc </li>
        <li> IAbc </li>
        <li> AbQ </li>
</ul>
```

What is “div” tag: .

In HTML, the ** ** tag is a generic container element that groups together other elements on a webpage. It is a block-level element that can be used to create sections or groupings of content on a webpage. The **div** tag itself does not have any inherent meaning or semantic value, but it provides a way to apply styles or manipulate the content within it as a group.

For example, you might use the **div** tag to group together related elements, such as a set of navigation links, a header or footer section, or a group of images. By assigning a **class** or **id** attribute to a **div**, you can target it with CSS and apply specific styles or layout to that section of content. The **div** tag is a versatile and commonly used element in web development, allowing for flexible and responsive designs.

What is container & wrapper?

A container refers to an HTML element that is used to group and organize other HTML elements together. It is usually used to apply styles or functionality to a group of elements, such as a section or a navigation bar. A container can be any HTML element, but commonly used container elements are and <section>.

```
<div>
        <section>
                <h1> Heading 1 </h1>
                <ul>
                        <li> list 101 </li>
                        <li> list 1 </li>
```

```
                    <li> list 20 </li>
                </ul>
            </section>
        </div>
```

What is Block (Block level content) & Inline element (Inline level content), and give me difference.

Block-level elements create a rectangular block on the page, and they take up the full width of their parent container by default. Examples of block-level elements include , <h1> to <h6>, <p>, <ul>, <ol>, <li>, <table>, and <form>. These elements are often used to group and structure content on a page.

```
<div style="background-color: blue;"> Div container </div>
<p style="background-color: blue;"> Paragraph 1 </p>
```

Inline elements, on the other hand, are used to apply styles and markup within a block-level element without creating a new line. They do not create a rectangular block on the page and only take up as much width as needed by their content. Examples of inline elements include <a>, , <img>, , , and <button>.

```
<span style="background-color: blue;"> My span 1 </span>

<a href="https://www.google.com" style="background-color: blue;"> Google </a>
```

The key difference between block-level and inline elements is their default display behaviour. Block-level elements create a new line before and after the element, while inline elements do not. Additionally, block-level elements can contain other block-level and inline elements, while inline elements can only contain other inline elements or text.

Is it possible to convert the Block-Element to Inline-Element or visa-versa.

Yes, it is possible using style, defining the display wither inline or block.

- **Block to Inline:**

```
<div style="background-color: blue; display: inline;"> Div container </div>
```

- **Inline to Block:**

```
<span style="background-color: blue; display: inline;"> My span 1 </span>
```

Day: 13-05-2023

What is table, row (tr), thead (th), tbody (td), tfoot in html? [Go to: **Day 03 (13 May)**]

What is rowspan, colspan.

- **rowspan:** it helps to expand the row (based on how many rows you want to expand).
- **colspan:** it helps to expand the column (based on how many columns you want to expand).

Day: 16-05-2023

What is Form?

An HTML form is a section of a document that collects user input. The input from the user is generally sent to a server (Web servers, Mail clients, etc). We use the HTML <form> element to create forms in HTML.

The <form> element can contain one or more of the following form elements:

- input element is used to create text fields, password fields, checkboxes, radio buttons, and other types of input elements.
- textarea element is used to create a text area.
- select element is used to create a drop-down list.
- button element is used to create a button.
- label element is used to associate a label with an input element.
- output element is used to display the value of an input element.

"id" attribute of <input> tag and "for" attribute in <label> tag must be same, because of mapping or make a link. E.g.:

```
<label for="username">Enter Username</label>
<input type="text" id="username" placeholder="username">
```

<form> tag: all elements and attributes for each tag: MDN Form
<fieldset> tag: It helps to group some tags or some selected tags or some elements.
legend: it helps to define the section in the fieldset.

Is it possible to select multiple options using <select> & <option> tag?

Yes, it is possible to select multiple options using the **<select>** and **<option>** tags in HTML. To enable multiple selections, you can add the **multiple** attributes to the **<select>** tag. Additionally, you can add the **multiple** attributes to each **<option>** tag that you want to allow users to select.

Here's an example:

```
<select multiple>
    <option value="option1">Option 1</option>
    <option value="option2">Option 2</option>
    <option value="option3">Option 3</option>
</select>
```

In the above example, the **multiple** attributes is added to the **<select>** tag, which allows the user to select multiple options. The user can hold down the Ctrl (or Command on Mac) key while clicking on the options to select multiple options.

When the form is submitted, the selected options can be accessed using server-side processing or JavaScript to retrieve the values of the selected options.

It's important to note that the appearance and behaviours of the multiple select feature may vary across different browsers and operating systems.

Day: 18-05-2023

Html form: pass

Day: 25-05-2023 to 26-05-2023

<img src='file_img.png' alt='image1' loading='lazy' >

loading='lazy' attribute in image tag: loading lazy is very useful because it can help in optimizing the loading time of a web page. When you include the attribute "loading=lazy" in an image tag, it tells the browser to delay the loading of the image until it is about to come into the user's viewport, or until the user starts scrolling towards it. This means that images located further down the page or outside the initial viewport are not loaded immediately when the page loads. Instead, they are loaded dynamically as the user scrolls or interacts with the page.

<video src='' controls autoplay muted type='video/mp4' >

- **controls:** it helps to run the video/ show the video player.
- **autoplay:** it helps to autoplay the video.
- **muted:** it helps to mute the video.
- **type:** define type of video.
- **loop:** run the video continuously.

How to increase the loading time of web page?

The loading time of videos when using the **`<iframe>`** tag can be influenced by several factors:

- **Video Size and Format:** The file size and format of the video can impact loading time. Large video files or videos in high-resolution formats may take longer to load, especially if the user has a slow internet connection.

- **Network Connection:** The speed and stability of the user's internet connection can affect video loading times. If the network connection is slow or unstable, it may result in buffering or delays when loading the video.

- **Server Performance:** The performance of the server hosting the video content can also impact loading times. If the server is experiencing high traffic or has limited resources, it may take longer to deliver the video content to the user's browser.

- **Browser Compatibility:** Different browsers may handle video loading differently. Some browsers may have more efficient video playback mechanisms or better support for certain video formats, resulting in faster loading times.

- **Other Factors:** Other factors such as the presence of ad blockers, browser extensions, or heavy scripting on the webpage can also affect the loading time of videos.

To improve video loading times, you can consider optimizing the video files for web playback, compressing the videos to reduce file size, using video formats that are widely supported, and ensuring a stable and fast internet connection. Additionally, using a dedicated video hosting service or content delivery network (CDN) can help improve the delivery speed of videos by distributing them across multiple servers globally.

Day: 28-05-2023

<head>:

<title>:

<meta>: it helps to optimize the search engine.

favicon: it helps to show the icon/image/logo in your title document.

code: <link rel='shortcut icon' href='logo.jpg' type='image/x-icon'

and make sure that this logo.jpg file is present inside the root, means where index.html or others html files are present.

File Structure:

```
MyProject
|
|--------- index.html
|--------- about.html
|--------- logo.png
|
|--------- CSS
|          |------ index.css
|          |------ about.css
|
|--------- JS
|          |------ myfile.js
|
|--------- Img
|          |------ img1.jpg
|          |------ img2.jpg
|          |------ img3.png
|
```

- <meta charset="UTF-8">: helps to encoding the character, so that you can see it in digital format. Here you can add multiple different encoding format. UTF-8 supports multiple character like Chines, Arabic, French etc.

- <meta http-equiv="X-UA-Compatible" content="IE=edge">: it helps to render the web pages using latest rendering engine.

- <meta name="viewport" content="width=device-width, initial-scale=1.0">:

  viewport means a window (in a browser) where your content is showing/ display,

  initial-scale=1.0 means window size is normal,

  content='width=device-width**'** means it helps to manage your content based on your devices like: laptop, mobile, tab, etc.

How to add 'google map' in html.

CSS

Day: 30-05-2023

What is DOM? And DOM manipulation?

Difference between "id" and "class" in html.

Suppose you design a tag using two same class names in css, then second class helps to override the design of the first-class design.

```
<div class=" first second"> </div>

<style>
        first {
                background-color: blue;
        }
        second {
                background-color: red;
        }
</style>
```

In this div tag, your final background-color is red, because it overrides the first background-color.

- m0+p0 = {margin:0; padding:0}
- MDN CSS Building Blocks: https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks
- View: Toggle Word Wrap: do in your vs code, so that contents are visible properly.

CSS Box-Model.

Margin Collapse:

Box Sizing: determine what out of padding and border is included in elements width and height can be content-box (include only content in width/height) or border-box (the content width and height includes content+padding+border).

- display: none; → It means the element is removed from the document flow. Its space is not blocked.
- visibility: hidden; → It means the element is hidden but its space is reserved.

Generic family: Broad class of similar fonts e.g.: serif, sans-serif. When I say fruits, it can be any fruits (e.g.: apple, mango, etc.,). When we say "serif" then it can be any "serif font."

Generic family is bigger class or broad group of fonts where font-family is sub-classes.

There are more units for describing size other than "px". There is **em**, **vh**, **vw**, **rem**, **% (percentage)**, etc.

What wrong with pixel?

Pixels (px) are relative to the viewing device. For a device with size 1920 x 1080, 1px is the 1 unit out of 1920/1080.

Relative lengths: These units are relative to the other length property. Following are some of the most commonly used relative lengths.

- **em:** relative to the parent font size
- **rem:** relative to the root font size (<html> tag). It is used when you want to resize the entire page with particular element.
- **vw:** relative to 1% viewport width
- **vh:** relative to 1% viewport height
- **% (percentage):** relative to the parent element.
- **min-height**
- **min-width**
- **max-height**
- **max-width**

Position property: Used to manipulate the location of an element following are the possible values:

- **static:** The default value, top|bottom|left|right|z-index has no effects. It follows the document flow.
- **relative:** top|bottom|left|right|z-index will now work. Otherwise, the element is in the flow of document like static.
- **absolute:** the element is removed from the flow and is relatively positioned to its first non-static ancestor top|bottom etc works.
- **fixed:** just like **absolute** except the element is positioned relative to the browser window.
- **sticky:** The element is positioned based on user's scroll position.

z-index: if you want to see any element in the top among some overlapping elements, then set the z-index value high so that you can see this element in top section. (The z-index property specifies the stack order of an element. It defines which layer will be above which in case of overlapping elements)

list-style: list_style_type list_style_position list_style_url

list-style: square inside url('img.png');

float: This property flows the element towards left or right.

clear: It is used to clear the float. This property specifies what elements can float beside a given element. (left or right or both)

FlexBox: Aims at providing a better way to layout, align and distribute space among items in a container.

```
.container {
        display: flex;
}
```

flex-direction: this property defines the direction towards which items are laid/ render. Can be row (default), row-reverse, column, column-reverse.

Flex property for parent (flex container):

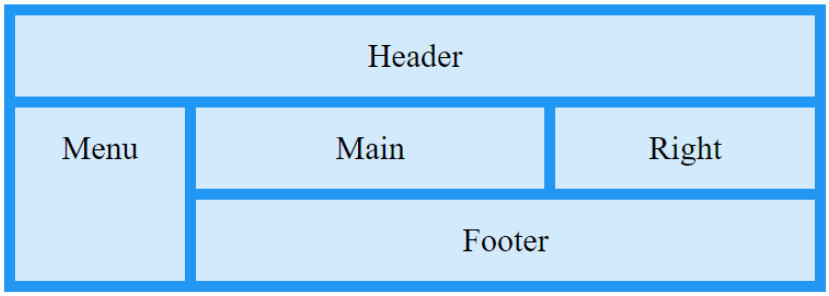
- **flex-wrap:** can be wrap, nowrap, wrap-reverse. Wrap items as needed with this property. Make responsive the container, suppose you have 5 boxes in a particular container, and if you wrap the container then, it resized based on window and showing top to bottom (based on window size).
- **justify-content:** Defines alignment along main axis (horizontal). Best way defines the position (center, left, right, end, etc.) horizontal of container.
- **align-item:** Defines alignment along main axis (vertical).
- **align-content:** aligns a flex container's lines when there is extra space in the cross axis. Remove the extra spaces when you minimize the window viewport and show as a group.

Flex property for the children (flex items):

- **order:** control the order in which the items appear in the flex container. Change the positioning order of your child container in the parent container.
- **align-self:** allows default alignment to be overridden for the individual flex items. Change the horizontal or vertical position of a sub container.
- **flex-grow:** defines the ability for a flex item to grow. Increase the width or height of the container based on **flex-direction.**
- **flex-shrink:** specifies how much a flex item will shrink compare to the rest of the flex items.

CSS Grid:

The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.



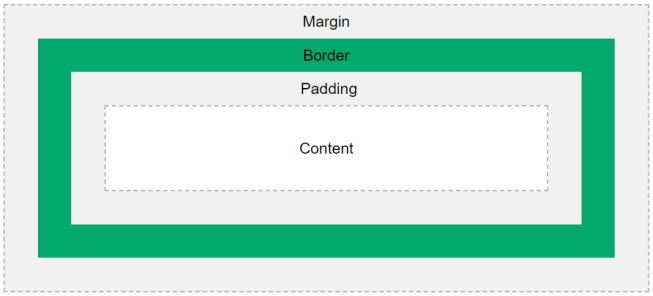
.container { display: grid; }

All the direct children automatically grid items.

- **grid-column-gap:** this property used to adjust the space between the columns of a css grid.
- **grid-row-gap:** this property used to adjust the space between the rows of a css grid.
- **grid-gap:** shorthand property for **grid-column-gap** and **grid-row-gap**.

Day: 31-05-2023

CSS Box Model:



margin: 10px; means, top 10px → right 10px → bottom 10px → left 10px

padding: 10px; means, top 10px → right 10px → bottom 10px → left 10px

top → right → bottom → left

- **padding:** it helps to separate the content from the border. Clears an area around the content. The padding is transparent.
- **margin:** Clears an area outside the border. The margin is transparent. It helps to separate the containers or elements among them.
- **border:** A border that goes around the padding and content.

box-sizing: border-box; It helps to manage the container by using the size (height & width) of padding and border. Define the size using border and padding.

box-sizing: content-box; Define the size without using border & padding.

CSS Float, Clear, Overflow:

overflow: The css overflow property controls what happens to content that is too big to fit into an area. Means, when your content is not fit into area then your content is overflow.

overflow properties:

- **visible:** default property.
- **hidden:** hide the overflow contents.
- **scroll:** showing all the content by scrolling the container. Vertical & Horizontal scroll bars are occurring.
- **auto:** showing all the content by scrolling the container. Either vertical or horizontal or both scroll bars are occurring based on the content.

float: The css property specifies how an element should float.

float values:

- **left**
- **right**
- **none**
- **inherit**

clear: It helps to clear the **float** property/value (left, right, both, none, etc.).

CSS Flex Box:

There are two types of Flex: one is **Flex Container** and another is **Flex Children.**

Flex Container: The flex container becomes flexible by setting the display property to flex.

Flex Container properties:

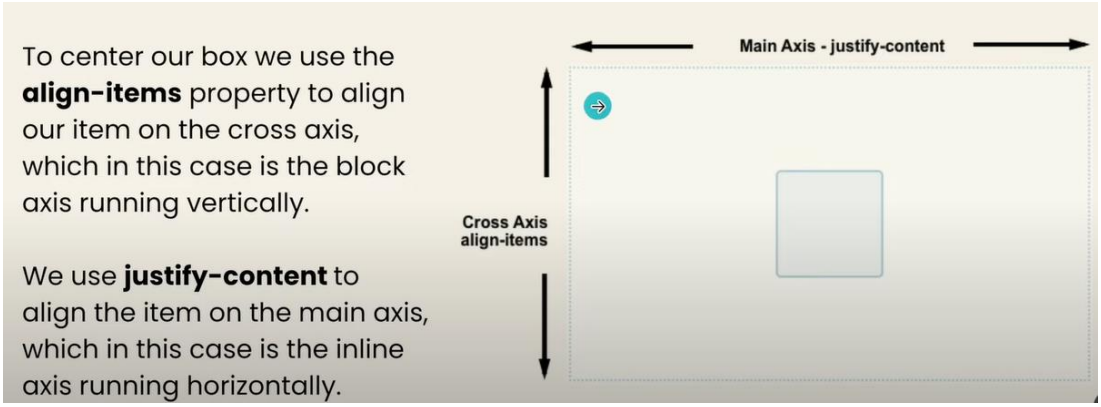
- **flex-direction:** change the box direction, like: row, column, row-reverse, column-reverse, etc.
- **flex-wrap:** wrap the content, mean when you minimize or change the window size then sub containers are placed column wise based on given condition (wrap, wrap-reverse, nowrap etc.).
- **flex-flow:** it helps to wrap the content (based on condition) and change the direction (based on condition). flex-flow: wrap row;
- **justify-content:** align the content/ container horizontally based on condition. By default, its row, if you want to change the direction then apply the **flex-direction column**, then apply **justify-content.**

    justify-content: space-around; → It helps to add some space between + left + right the sub container or contents based on available space present in parent container/ content.
    justify-content: space-between; → It helps to add some space between the sub container or contents (not left and right side) based on available space present in parent container/ content.

- **align-items:** align the content/ container vertically based on condition, by default, stretch.
- **align-content:** minimize/add the space between the contents or sub-containers.

    align-content: flex-start; → minimize the space & arrange between the sub-container or contents

- **gap:** it defines the gap (some pixel) between the siblings or sub containers for both row and column.
- **row-gap:** row wise gap (some pixel) between the sub container or siblings.
- **column-gap:** column wise gap (some pixel) between the sub container or siblings.



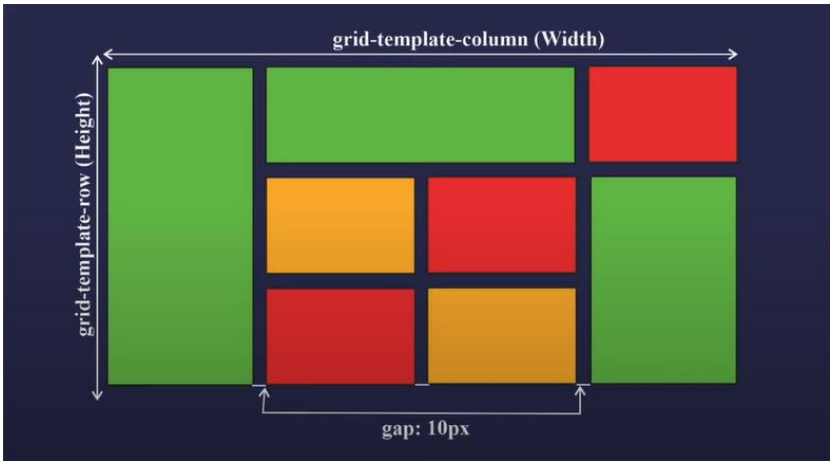
Flex Child properties: The direct child items of a flex container automatically becomes flexible (flex) items.

- **order:** change the order (left or right side) of child content or sub-container. Generally ordered the containers in html.
- **flex-grow:** if you apply flex-grow in a particular container, then it increases the width or height itself, by default value is zero (0).

- **flex-shrink:** shrink (minimize the height & width) the element/content/sub-container., by default value is 1 (one).
- **flex-basis:** It helps to define the initial width in the direction of main axis (horizontal) for a particular content/element/sub-container. Or it changes the default width of particular content/element/sub-container.
- **flex:** It is a shorthand property of **flex-grow, flex-basis, flex-shrink**.
- **align-self:** align the items cross-axis (vertically) for a particular content/element/sub-container.

CSS Grid Layout:

display: grid; → set the container property grid. By default, gird work top to bottom.



Grid Container: The grid container becomes by settings the display property to grid.

Properties:

- **gap:** gap between the child container, you can use **row-gap** or **column-gap**.
- **grid-template-columns:** define the width of the columns, like:

  grid-template-columns: 100px 200px 150px; →**100px** width those sub-containers that present in first column, **200px** width those sub-containers that present in second column, **300px** width those sub-containers that present in third column, and so on. And remaining space will be free if space is available.

  grid-template-columns: 1fr 1fr 1fr 1fr; → divide the width of the parent container into 4 equal sizes, and arrange all the elements based on that. And remaining space will be free if space is available.

  grid-template-columns: repeat(4, 1fr) → repeat 1fr for 4 times. Following the above descriptions.

  grid-template-columns: auto auto → divide the width of sub containers into two equal sizes (column wise) based on available width of the parent container.

- **grid-template-rows:** define the width of the rows, like: **grid-template-columns**.
- **grid-auto-rows:** define the height of those rows where you not mentioned any height.
- **justify-content:** align the content/ container horizontally based on condition. By default, its row, if you want to change the direction then apply the **flex-direction column**, then apply **justify-content.**

justify-content: space-around; → It helps to add some space between + left + right the sub container or contents based on available space present in parent container/ content.
justify-content: space-between; → It helps to add some space between the sub container or contents (not left and right side) based on available space present in parent container/ content.

justify-content: space-evenly; → It helps to add some space between + left + right the sub container or contents based on available space present in parent container/ content.

- **align-items:** align the content/ container vertically based on condition, by default, stretch.
- **align-content:** minimize/add the space between the contents or sub-containers.

align-content: flex-start; → minimize the space & arrange between the sub-container or contents

Grid Child: The direct child elements of a grid container automatically becomes grid items.

Properties:

- **justify-self:** it helps to align the particular item horizontally, by default the property is stretch.

Difference between **justify-content** & **justify-self:** → → **justify-content** helps to align the all items horizontally (by default the property is stretch), **justify-self** it helps to align the particular item horizontally, by default the property is stretch.

- **align-self:** it helps to align the items vertically, by default the property is stretch.

Difference between **align-content** & **align-self:** → → **align-content** helps to align the all items vertically (by default the property is stretch), **align-self** it helps to align the particular item vertically, by default the property is stretch.

- **grid-column:** increase the width of column from start to end: (1 / 3). Shorthand property of **grid-column-start** & **grid-column-end**.

grid-column-start: 1; grid-column-end:3; → increase the column width up to 1 to 2.

- **grid-row:** shorthand property of **grid-row-start** & **grid-row-end**. And it similar to the **grid-column**, but increase the length of row.

grid-row: 1/3; → start from 1st row and end to 3nd row index. (1 to 2 row expand)

grid-row: 1/ span 2; → start/ expand from 1st row to 2nd row.

- **grid-area:** shorthand property of **grid-row** & **grid-column**.

grid-area: 1/3/1/4; → **row** start from 1st to 3rd row and **column** start from 1st column to 4th column.

CSS Media Queries:

It helps to make responsive webpage. Syntax:

```
@media (width <=200px) {
        body {
                background-color: red;
        }
}

@media (width <=200px) and (height <= 200px) {
        body {
                background-color: yellow;
        }
}
```

Note: you can multiple media queries and also add some logical operator, like: and, or, etc.

CSS Variables:

It helps to access the specific property inside a particular container. Suppose, you define a variable inside "box1" container, then all the elements that present inside the "box1" container, those elements are access the variables, outside elements cannot access the variables. (If you define the variable local (inside container/element/tag/id/class) → then it will be local **or,** if you define the variable global (html: root) → then it will be global).

Example: --

Html:

```
<div class="box1>
        <h1> Welcome </h2>
        <p> some paragraph </p>
</div>
<div class="box2>
        <h1> Hello All </h2>
        <p> paragraph 2 </p>
</div>
```

CSS:

```
.box1{
        --my-var: yellow; /* "my-var" is variable name */
}
.box1 h1{
        background-color: var(my-var);
}

.box1 p{
        color: var(my-var);
}
```

```
.box2 p{
        background-color: var(my-var, green); /* here color will not become yellow
        but become green or fall in green */

        color: var(primary-text-color);
}


/* global variable */

:root{
        --primary-text-color: orange;
        --secondary-text-color: blue;
}
```

Pseudo

Pseudo Class:

A pseudo class is used to define a special state of an element. **Go to → w3school** (or) **MDN**

Some examples are:

- **:visited**
- **:link**
- **:hover**
- **:active**
- **:first-child**
- **:last-child**
- **:nth-child**

Pseudo Element:

A CSS pseudo element is used to style specified parts of an elements. (There is double colon in pseudo element)

Some examples are:

- **::first-letter**
- **::first-line**
- **::before** (add something before the element)
- **::after** (add something after the element)

Transform properties:

The transform property applies a 2D or 3D transformation to an element. This property allows you to rotate, scale, move, skew, etc., elements. transform: scale(20px);

- **scale:** it helps to increase or decrease the width & height based on current width & height.
- **scalex:** increase the width in x direction but not height.
- **scaley:** increase the height in y direction but not width.
- **scalez:** increase the are in z-direction.
- **skew:** skew in a particular degree in x & y direction.
- **translate:** helps to move the element in x or y direction based on some value.
- **rotate:** rotate the element/ container in a particular degree.

Transform Origin properties:

The transform-origin property allows you to change the position of transformed elements.

2D transformations can change the x-axis and y-axis of an element. 3D transformations can also change the z-axis of an element. transform-origin: top left;

- **left:**
- **center:**
- **right:**
- **length:**
- **%:**

Day: 04-06-2023

CSS depends on which 4 factors:

- **Specificity:** means priority of selectors Inline css > id > class > element
- **! important**
- **Order in which css declared**, the last one has higher priority
- **Inheritance** means some css properties will apply on all the elements inside the selector like color but some not like border

Suppose,

background-color: yellow !important;
It basically, indicate that forget all the background-color and use this one, because it is important.

External css: separation of concern principal is maintained.

Selector: it is a way to target an html element for designing or styling.

- **Tags or elements:** <p>, <h1>, <h2>, etc.
- **id selector:** specific "id" name.
- **class selector:** specific "class" name.
- **group selector:** select multiple "class names" or "id names" or both.

1 physical pixel: (1/96) inch = 0.26mm.

- **line-height:** distance between two lines. (Under font category)
- **font-family:** 'Gill Sans', 'Gill Sans MT', 'Trebuchet MS', 'sans-serif'; → → here I used different font families, it basically means, if the first one is available then used it, if first one is not available (for some reasons) then used second one, if second one is not available (for some reasons) then used third one, and so on. It's call **fall-back system**.

  You can use Google fonts.

Add color using different formats, like: **named, rgb(), hex(), hsl(), rgba().**

rgba (66, 176, 86, 0.45); → 0.45 stands for transparency.

#ffab23: The first two components (ff) represent **red**, second two components (ab) represent **blue** and last two components (23) represents **green** components and also another last two components represent **transparency/ opacity.**

hsl (hue, saturation, lightness):

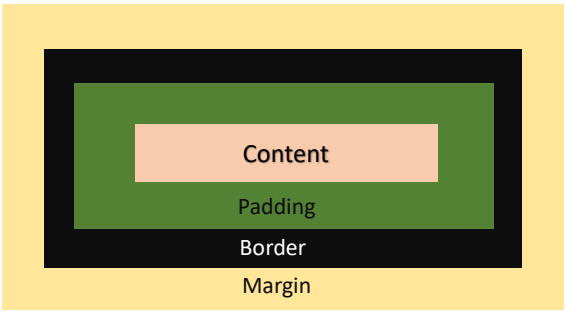
- **hue:** Hue is a degree on the color wheel from 0 to 360. **0 (or 360) is red, 120 is green, 240 is blue**.
- **saturation:** Saturation can be described as the intensity of a color. It is a percentage value from 0% to 100%.
- **lightness:** The lightness of a color can be describing as how much light want to give the color, where **0%** means **no light (dark)**, **50%** means **50%** light **(neither dark or light)**, **100%** means **full light.**

font-style: italic, normal, etc.
font-weight: bold, mention the thickness, etc.

border-top-left-radius: 5px; → add a radius 5px to **top left** corner.
border-top: 5px solid red; → add a radius 5px to **top portion**.
border-bottom: 5px solid red; → add a radius 5px to **bottom portion.**
border-left: 5px solid red; → add a radius 5px to **left portion.**
border-right: 5px solid red; → add a radius 5px to **right portion.**

Why height & width is not applied in tag: because **** tag is an **inline** element.

Box-Model: It helps to render the content inside the box whatever the height & width of the box.



- **padding:** gap between content and border.
- **border:**
- **margin:**

Universal selector (*):

```
*{
    /* Best Practice */
    padding: 0;
    margin: 0;
    box-sizing: border-box;
}
```