Front End Development Course Training Material

What Does Front Developer Do ?

A front-end developer is a type of software developer who specializes in creating and designing the user interface (UI) and user experience (UX) of websites and web applications. The primary responsibility of a front-end developer is to ensure that the visual and interactive aspects of a website or application are user-friendly, aesthetically pleasing, and functionally efficient.

A front-end developer builds the front-end portion of websites and web applications—the part users see and interact with. A front-end developer creates websites and applications using web languages such as HTML, CSS, and JavaScript that allow users to access and interact with the site or app.

Roles and Responsibilities :-

1. **User Interface Design**
2. **Web Development Languages**
3. **Responsive Design**
4. **Performance Optimization**
5. **Front end frameworks and Libraries**
6. **Version Control**

Introduction Web Programming

Web programming refers to the development of web applications and websites that are accessed over the internet . Basically it involves creating web pages , web applications and other online content that are displayed over web browsers . The programming languages that

are involved in web programming are Html , CSS , Javascript , Perl , Php etc . Each language has its own strength and weakness , the choice of language needed depends on the requirements of the project .

Key components of web programming

1. **Client side Technologies :-**

- **HTML :-** HTML stands for HyperText Markup Language. It is used to design web pages using a markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. A markup language is used to define the text document within a tag which defines the structure of web pages.
- **CSS :-** css is a stylesheet language used to design the webpage to make it attractive. The reason for using CSS is to simplify the process of making web pages presentable. CSS allows you to apply styles to web pages. More importantly, CSS enables you to do this independent of the HTML that makes up each web page.
- **Javascript :-** is the most powerful and versatile programming language used on the web. It is a lightweight, cross-platform, single-threaded and interpreted programming language. It is a commonly used programming language to create dynamic and interactive elements in web applications, easy to learn, compiled language.

2. **Server Side Technologies :-**

- **Server-Side Languages**:- Languages like PHP, Python, Ruby, Java, and Node js are used to handle server-side logic, process requests, and generate dynamic content.
- **Databases:-** Systems like Mysql, Postgres, Mysql or SQLite are used to store and retrieve data dynamically, enabling web applications to manage and manipulate information.
- **Server Environment:-** Software environments such as Apache, Nginx, or Microsoft IIS provide the infrastructure to host and serve web applications. They handle incoming requests, route them to the appropriate handlers, and send responses back to clients.

3. Frameworks :-

- **What is a Framework ?**

  Framework is a pre-written collection of standardized Html , CSS and Javascript code that developers can use to build web applications more efficiently .

  Basically it provides a set of tools , conventions , libraries for building user interfaces , handling events , managing application states , and interacting with APIs .

  Why do we need a Framework ?

They help developers to organize the code , reduce repetitive work , and achieve consistency across the project .

What is a Library ?

Library is a set of pre - written code used to perform a specific task , it looks after the details , offers functions .

Some popular Frameworks and libraries that a frontend developer must know are as follows

- **React :- React is a Javascript library , owned by facebook released in 2013 .**
- **Tailwind CSS :- Tailwind Css is a Css framework , it is highly customizable and provides a wide range of configurations .**
- **Angular :- Angular is based on Typescript ,it's a Javascript framework and superset of Javascript.**
- **Vue.js :- Vue.js is a Javascript framework , MVVM Model (** Model - View - ViewModel)
- **Bootstrap :- Bootstrap is an open source CSS framework , developed by Twitter .**

- **Material Design Lite :- It's a framework developed by Google , that is based on Material Design**
- **Material UI :- It is a design language developed by Google in 2014 , it uses more grid based layouts , responsive animations . (react library that implements google material design)**

While Material Design Icons offer a comprehensive set of guidelines and components that can be customized to fit specific needs, Material Design Lite provides a more limited range of customization options, focusing on simplicity and ease of use for web developers.

HTML Content :-

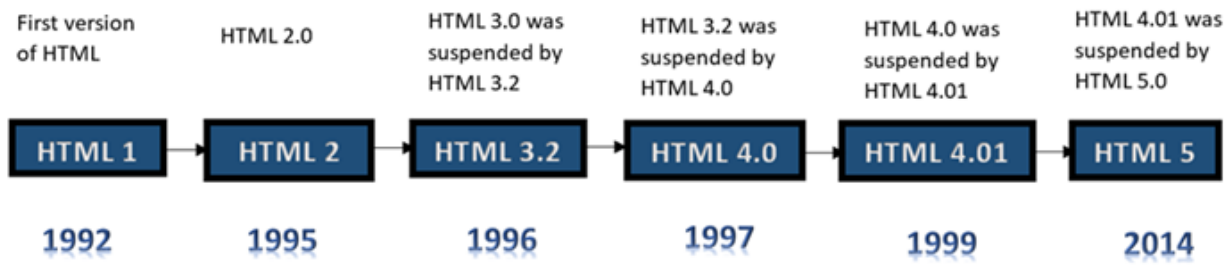
Introduction to web programming and html (history, syntax , concepts involved in html)
Html Basic formatting Tags (head , body , style , paragraph etc) , Atributes
Grouping and using span and div tags
Lists (ordered , unordered list)
Images , Hyperlink (anchor tags)
Tables
iFrames , intro to forms
Forms , Headers
Html Miscellaneous (Meta tags and depreciated tags of HTML along with XHTML and attributes)
Web browser
HTML 5 introduction
Semantics
Audio and video support
Canvas Elements

Geolocation API
Local storge
Responsive Images
Drag and Drop
Web workers , web sockets
Form Enhancements

HTML (Hypertext Markup language)

History

Tim Berners -Lee invented HTML in 1989 and officially introduced it to the world in the early 1990s. In late 1994, he founded the World Wide Web Consortium (W3C), which took over the HTML specifications. After several years of working on the specifications, the W3C switched its focus from HTML to XHTML.



HTML stands for **HyperText Markup Language**, it is a Standard Markup language for web pages. HTML is used to create content and structure of any web page. If you think of the human body as a

web page then HTML is the skeleton of the body. It is the building block of web pages .

HTML Basics: In the basic part we have covered all the fundamentals of HTML like - editors, basic tags, elements, attributes, heading, paragraph, formatting, etc..

HTML Tables: After getting the knowledge of fundamentals of HTML we should learn about tables. The primary tags used to create tables are <table>, <tr>, <td>, and <th>.

HTML Lists: The lists can be ordered or unordered depending on the requirement. In html we can create both ordered and unordered lists by using <ol> and <ul> tags and for the list item we can use <li> tag.

HTML Links: Links allow visitors to navigate between Web sites by clicking on words, phrases, and images. A link is specified using HTML tag <a>.

HTML Backgrounds: Background of a web page is a layer behind its content, which includes text, images, colors and various other elements. HTML allows you to change the background color of any elements within a document using bgcolor attribute.

HTML Colors: Colors are a way of specifying the appearance of web elements. Colors are very important aspects of web

design, as they not only enhance the visual appeal but also influence user behavior. They are also used to evoke emotions and highlight important content.

HTML Form: HTML forms are simple forms that have been used to collect data from the users. HTML form has interactive controls and various input types such as text, numbers, email, password, radio buttons, checkboxes, buttons, etc. There are lots of form elements in HTML, you can learn those from HTML - Forms.

HTML Media: Media is an important element in HTML, sometimes we want to include videos and audios into our websites, we can embed any media into our websites.

HTML Header: Header part of an HTML document is represented by the <head> tag. It serves as a container of various other important tags like <title>, <meta>, <link>, <base>, <style>, <script>, and <noscript> tags.

HTML Layout: Layouts specify the arrangement of components on an HTML web page. A good layout structure of the webpage is important to provide a user-friendly experience on our website. It takes considerable time to design a website's layout with a great look and feel.

HTML Graphics: HTML allows two types of graphics development in the document directly. SVG is an XML-based

markup language used for creating scalable 2D graphics and graphical applications, and Canvas gives you an easy and powerful way to draw graphics using JavaScript.

HTML Applications

As mentioned before, HTML is one of the most widely used languages on the web.

Web Page Development: HTML is used to create pages that are rendered over the web. Almost every page of the web has HTML tags in it to render its details in the browser.

Responsive UI: HTML pages now-a-days works well on all platforms, mobile, tabs, desktop or laptops owing to responsive design strategy.

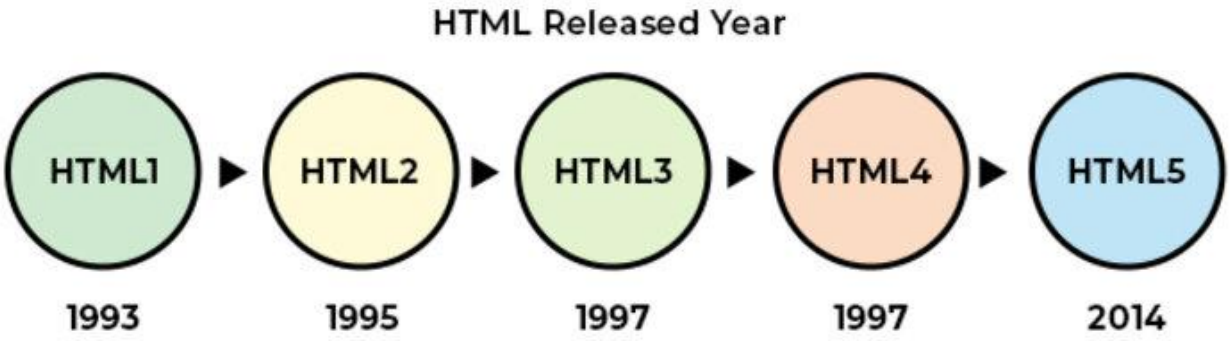
Game Development: HTML5 has native support for rich experience and is now useful in the gaming development area as well.

Mobile application development: HTML with CSS3 and Javascript can be used for developing cross-platform mobile applications.

Multimedia and video streaming: HTML5 offers support for multimedia elements like video and audio, which enables seamless media playback directly in web browsers.

Geolocation: Allows websites to request a user's geographic location. This is helpful for location-based applications and services.

HTML Versions



HTML Document Structure

HTML elements are hierarchical, meaning we can create elements inside of an element. But there are few rules to follow like the head can not be placed inside of a body like that so to know the basic structure of an HTML document please check the below example code.

```
<!-- HTML Version Declaration -->

<!DOCTYPE html>

<!-- HTML Root Element -->

<html>
```

```
<!-- HTML Head Section -->
<head>
    <!-- HTML Document Title -->
    <title>This is Title</title>
    <link>
    <meta>
</head>

<!-- HTML Body Section -->
<body>
    <!-- HTML Header Element -->
    <h1>This is Header</h1>
    <!-- HTML Paragrapgh Element -->
    <p>This is a Paragraph</p>
</body>
</html>
```

Role of Web Browsers in HTML

Web Browsers like Google Chrome, Mozilla Firefox, Microsoft Edge, Apple Safari, Opera, etc are able to show the output of HTML code, it will define the font size, weight, structure, etc based on tags and attributes.

-
- **HTML Tags , Elements and Attributes :-**
- **HTML Tags:** Tags are similar to keywords, which specify how a web browser will format and display content. A web browser can differentiate between simple content and HTML content with the use of tags.
- All HTML tags must be enclosed within < > these brackets.
- Every tag in HTML performs different tasks.
- If you have used an open tag <tag>, then you must use a close tag (except some tags)

  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.

Tags in HTML:-

- HTML Meta Tags -> (DOCTYPE, title, link, meta and style)

- HTML Text Tags -> <p>, <h1>, <h2>, <h3>, <h4>, <h5>, <h6>, , <pre>
- HTML Link Tags -> <a> , <base>
- HTML List Tags -> , <ul> ,
  HTML Attributes: Attributes are used to customize an element's behavior, special terms called HTML attributes are utilized inside the opening tag. An HTML element type can be modified via HTML attributes.
- **Syntax** :- name=”value”

  Ex:- src = “ “ , alt = “ ” , class = “ “ etc

  HTML Elements: Elements are building blocks of a web page. It consists of a start tag, an end tag, and the content between them.

Grouping Elements :- Arranging elements like (images , text , forms , etc) together and applying styling to it using CSS is specified as Grouping of Elements .

Grouping can be performed with the help of various tags such as , , <fieldset > , <section > , <footer> , <header> etc .

- ** :-** The div tag is known as the **Division tag**. The HTML tag is a block-level element used for grouping and structuring content. It provides a container to organize and style sections of a webpage, facilitating layout design and CSS

styling , this tag has both opening( ) and closing ( ) tags and it is mandatory to close the tag.

Ex :-

```
<!DOCTYPE html>

<html>


<head>
    <title>Div tag</title>


    <style>
        div {
            color: white;
            background-color: #009900;
            margin: 2px;
            font-size: 25px;
        }
    </style>
</head>


<body>
    <div> div tag
        <div>div 2</div>
    </div><br>
```

```
    <div> div tag </div><br>

    <div> div tag </div><br>

    <div> div tag </div><br>
</body>
</html>
```

- ** :-** The HTML span element is a **generic inline container** for inline elements and content. It used to group elements for styling purposes (by using the class or id attributes).

  A better way to use it when no other semantic element is available. The span tag is very similar to the div tag, but div is a block-level tag and span is an inline tag.

  Ex:-

```
<!DOCTYPE html>
<html>
<head>
    <title>span tag</title>
</head>
<body>
    <h2>Welcome To GFG</h2>
```

```
<p>
        <span style="background-color:lightgreen">
            Front End Training
        </span>
        starts from June and will end in september
        <span style="color:blue;">
            Classes will be conducted in
        </span> your own
        <span style="background-color:lightblue">
            shift timings
        </span>
        try to attend classes regularly
    </p>
</body>
</html>
```

Difference between and

1. is block level element , while < span > is inline level element .
2. accepts align attribute while < span > does not accept align attribute.
3. This tag should be used to wrap a section, for highlighting that section , < span > . This tag should be used to wrap any specific word that you want to highlight in your webpage.

Fieldset tag :- It is used to group the related elements in the form and it draws the box around the related elements . It can be used anywhere in html , but except button tag .

The main difference between div and fieldset tag is that div is used for layout structure in webpage , whereas fieldset is used for layout specifically for form fields .

Ex :-

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

<h1>The fieldset element</h1>

<form >
 <fieldset>
  <legend>Personal</legend>
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <label for="lname">Last name:</label>
  <input type="text" id="lname" name="lname"><br><br>
  <label for="email">Email:</label>
  <input type="email" id="email" name="email"><br><br>
  <label for="birthday">Birthday:</label>
  <input type="date" id="birthday" name="birthday"><br><br>
  <input type="submit" value="Submit">
 </fieldset>
</form>
```

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

Footer tag :- Footer tag is used to define a footer for a document or section . <footer> element contains information such as

- Copyright information
- Authorship information
- Contact information
- Back to top links
- Related documents

You can have several footer elements in one document

Section tag:- It defines a section in a document

Legend tag :- It defines a caption for Fieldset element

Lists :-

List is a record of related information , used to display the data or any information on web pages in **ordered** or **unordered** form .

Lists are of three types

- Ordered List
- Unordered List or Bulleted List
- Description List

Ordered List :- In ordered list all the items are marked by number by default sequentially . It starts <ol> tag and items inside it as written within <li> tags and every tag must be closed .

The elements within the list tag <ol> are referred to as items .

 → ordered list

 → items of list

Ex:-

```
<html>
    <head>
        <Title>Ordered List</Title>
    </head>
    <body>
        <ol>
            <li>Java</li>
            <li>Python</li>
            <li>Ruby</li>
            <li>React</li>
            <li>JavaScript</li>
            <li>Nodejs</li>

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

Unordered List:- The list items here are marked with Bullets . It starts with <ul> tag , each list item starts with <li> tag .

Ex:- Replace in the above code with below part

If you want different shape types then use

Style → `<ul style = "list-style-type:square">`

Type → `<ul type="circle">`

```
<ul>
            <li>Css</li>
            <li>Material UI</li>
            <li>Tailwind CSS</li>
            <li>Bootstrap</li>
            <li>Next.Js </li>
            <li>Nodejs</li>
</ul>
```

Description List :- Description List is a list of terms with a description of each term , <dl> tag defines description list , <dd> tag describes each term .

Ex:-

```
<dl>
            <h1>Description List</h1>
            <dt>Java</dt>
            <dd> - Programming Language </dd>
```

```
        <dt>Python</dt>
        <dd> - Programming Language </dd>
        <dt>JavaScript</dt>
        <dd> - Programming Language </dd>
        <dt>React</dt>
        <dd> - Framework </dd>
    </dl>
```

Replace the codes according to the list types

Task :-

1. **What are iFrames ?**
2. **Purpose of iFrames ?**
3. **In Which case we use iFrames ?**

Image tag :- Image tag is used to display the images in a webpage by linking them . It is defined by attributes like src , width , height , cross-origin alt etc and does not have any closing tag .

There are 2 ways of inserting an image into webpage

- By providing a full path or address (URL) to access an internet file.
- By providing the file path relative to the location of the current web page file.

  Syntax :-

```
<img src="url" alt="some_text" width="" height="">
```

Ex:-

```
<html>



<head>

    <title>Welcome To FrontEnd Development</title>

</head>

<body>

    <h2>NextJs Developer</h2>

<img src=”use your own images“  alt="Next js img" />

</body>

</html>
```

Adding image in animated format in HTML :-

To add an animated image in HTML, use the <img> tag with the src attribute pointing to a GIF file, providing engaging motion to enhance webpage content.

Ex:-

```
<html>
```

```
<body>

    <h3>Adding a gif file on a webpage</h3>

    <img
src="https://media.geeksforgeeks.org/wp-content/uploads/20210203164653/ezgifcomgifmaker2.gif"

         alt="smiley"

         style="width: 200px; height: 200px" />

</body>

</html>
```

Anchor tag :-

The **<a> tag** defines a hyperlink, which is used to link from one page to another. The most important attribute of the <a> element is the **href attribute**, which indicates the link's destination. This attribute determines where the user is directed upon clicking the link.

Syntax:- `<a href = "link"> Link Name </a>`

Ex:-

```
<html>
```

```
<body>

    <h2>

        Welcome to GeeksforGeeks

        HTML Tutorial

    </h2>

    <!-- Opening link in same tab  -->

    <a href="https://www.geeksforgeeks.org/html-tutorials/">

        <!-- Opening Link in new tab  -->

    <a href="https://www.geeksforgeeks.org" target="_blank">Visit
GeeksforGeeks</a>

        GeeksforGeeks HTML Tutorial

    </a>

</body>

</html>
```

Linking Image with anchor tag :-

When you click on the image you will be redirected to new page .

Ex:-

```
<html>
```

```
<body>

    <p>Click on the image to open web page.</p>



    <!-- anchor tag starts here -->

    <a href="https://www.geeksforgeeks.org/">

        <img src=”use img from google”  width="300" height="250" />

    </a>

    <!-- anchor tag ends here -->

</body>

</html>
```

Task - 2 ➖

1. **Create an unordered list inside ordered list , in unordered list the items must be marked by using square points .**
2. **Take 4 sections in a webpage and display images inside it .**

Tables :-

Tables are a way to represent the data in form of rows and columns in tabular format. We can store text and numerical information data in table. A table is a useful tool for quickly and easily finding

connections between different types of data. Tables are also used to create databases.

Tags used in HTML tables are as follows :-

1. **<table> :-** Defines structure for organizing data rows and columns within a webpage .
2. **<tr> :-** Represents a row within an HTML table, containing individual cells.
3. **<th> :-** Shows a table header cell that typically holds titles or headings.
4. **<td> :-** Represents a standard data cell, holding content or data.
5. **<captain> :-** Provides a title or description for the entire table.
6. **<thead> :-** Defines the header section of a table, often containing column labels.
7. **<tbody> :-** Represents the main content area of a table, separating it from the header or footer.
8. **<tfoot> :-** Specifies the footer section of a table, typically holding summaries or totals.
9. **<col> :-** Defines attributes for table columns that can be applied to multiple columns at once.
10. **<colgroup> :-** Groups together a set of columns in a table to which you can apply formatting or properties collectively.

Defining Tables in HTML:-

An HTML table is defined with the “table” tag. Each table row is defined with the “tr” tag. A table header is defined with the “th” tag. By default, table headings are bold and centered. A table data/cell is defined with the “td” tag.

Table Cells:- Table Cell are the building blocks for defining the Table. It is denoted with <td> as a start tag & </td> as a end tag.

Syntax:- <td> content </td>

Table Rows:- The rows can be formed with the help of combination of Table Cells. It is denoted by <tr> and </tr> tag as a start & end tags.

Syntax:-

<tr> Content...</tr>

Table Headers:- The Headers are generally use to provide the Heading. The Table Headers can also be used to add the heading to the Table. This contains the <th> & </th> tags.

Syntax:-

<th> Content...</th>

Ex:-

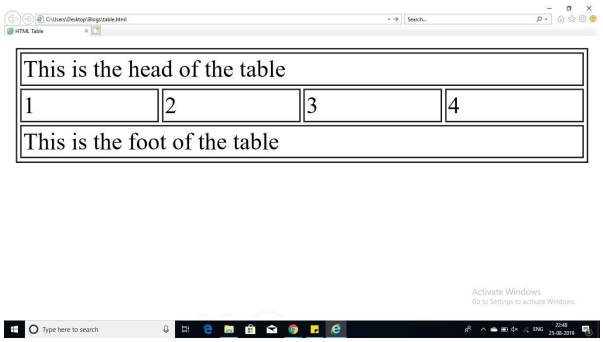
```
<!-- index.html -->

<!DOCTYPE html>

<html>



<head>

    <style>

        table,

        th,

        td {

            border: 1px solid black;

        }

    </style>

</head>



<body>
```

```
<table style="width:100%">
  <tr>
    <th>First Name</th>
    <th>Last Name</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Priya</td>
    <td>Sharma</td>
    <td>24</td>
  </tr>
  <tr>
    <td>Arun</td>
    <td>Singh</td>
    <td>32</td>
```

```
        </tr>
        <tr>
            <td>Sam</td>
            <td>Watson</td>
            <td>41</td>
        </tr>
  </table>
</body>
</html>
```

Task :-

Write the code for the following format.



Forms :-

Form is used to collect the inputs from the user via variety of interactive controls. These controls range from text fields, numeric inputs, email fields, passwords fields, check boxes, radio buttons, submit buttons . Html forms serves as versatile container to collect input from users and there by enhancing user interaction.

Syntax:-

<form>

// form elements

</form>

Form Elements :-

Form comprises several elements each justifies unique purpose. Among all the elements <input> element is versatile as it accepts input data from users of various types such as text, password, numbers, email etc.

Let us see the elements of forms

1. Label :- it defines the label for form elements
2. Input :- takes input from the user of various types such as text, numerical, password, email etc.
3. Button :- It defines a clickable button to control other elements or execute a functionality.
4. Select :- It is used to create a drop-down list.
5. Textarea:- It is used to input long text content.
6. Fieldset:- It is used to draw a box around other form elements and group the related data.
7. Legend :- It defines a caption for fieldset elements
8. Option :- It is used to define options in a drop-down list.
9. Optgroup :- It is used to define group-related options in a drop-down list.
10. Datalist:-It is used to specify predefined list options for input control.

Ex :-

```
<!DOCTYPE html>
```

```
<html lang="en">

<head>

<title>Html Forms</title>

</head>

<body>

  <h2>HTML Forms</h2>

  <form>

    <label for="username">Username:</label><br>

    <input type="text" id="username" name="username"><br><br>



    <label for="password">Password:</label><br>

    <input type="password" id="password"
name="password"><br><br>



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

  </form>
```

```
</body>

</html>
```

Ex :-

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Form</title>
    <style>
      body {
        display: flex;
        justify-content: center;
        align-items: center;
        height: 100vh;
        margin: 0;
        background-color: #f0f0f0;
      }
```

```
form {
    width: 400px;
    background-color: #fff;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 0 10px
        rgba(0, 0, 0, 0.1);
}

fieldset {
    border: 1px solid black;
    padding: 10px;
    margin: 0;
}

legend {
    font-weight: bold;
    margin-bottom: 10px;
```

```
}

label {
    display: block;
    margin-bottom: 5px;
}

input[type="text"],
input[type="email"],
input[type="password"],
textarea,
input[type="date"] {
    width: calc(100% - 20px);
    padding: 8px;
    margin-bottom: 10px;
    box-sizing: border-box;
    border: 1px solid #ccc;
    border-radius: 4px;
```

```
        }

        input[type="radio"] {
            margin-left: 20px;
        }

        input[type="submit"] {
            padding: 10px 20px;

            border-radius: 5px;
            cursor: pointer;
        }
    </style>
</head>
<body>
    <form>
        <fieldset>
            <legend>
```

```
    User personal information
</legend>
<label
    >Enter your full name</label
>
<input type="text" name="name" />
<label>Enter your email</label>
<input
    type="email"
    name="email"
/>
<label>Enter your password</label>
<input
    type="password"
    name="pass"
/>
<label
    >Confirm your password</label
```

```
>
<input
    type="password"
    name="confirmPass"
/>
<label>Enter your gender</label>
<input
    type="radio"
    name="gender"
    value="male"
/>Male
<input
    type="radio"
    name="gender"
    value="female"
/>Female
<input
    type="radio"
```

```
        name="gender"
        value="others"
      />Others
      <label
        >Enter your Date of
        Birth</label
      >
      <input type="date" name="dob" />
      <label>Enter your Address:</label>
      <textarea
        name="address"
      ></textarea>
      <input
        type="submit"
        value="submit"
      />
    </fieldset>
  </form>
```

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

Task :-

Design a form to place an order for a product, specify all the details of the product, and take a reference of any Ecommerce platform.

Nested table (sample code)

```
<body>
  <h4>How to create nested tables within tables in HTML?</h4>
  <table class="main-table">
    <tr>
      <td>
        Main table cell 1
        <table class="nested-table">
          <tr>
            <td>Nested table cell 1</td>
            <td>Nested table cell 2</td>
          </tr>
        </table>
```

```
        </td>
        <td>Main table cell 2</td>
      </tr>
    </table>
  </body>
```

Iframes :-

Iframes are used to display a webpage within a webpage .

It specifies inline frame

Syntax:-

```
<iframe src="url" title="description"></iframe>
```

- The **src** attribute specifies the **URL** of the document you want to embed.
- Iframes can include **videos**, **maps**, or **entire web pages** from other sources.

Ex:-

```
<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="demo_iframe.htm" height="200" width="300" title="Iframe Example"></iframe>

</body>

</html>
```

Ex :- Embed a map

```
<html>

    <body>

        <h1>Map</h1>

        <iframe src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d3807.01379096796!2d78.54706697414181!3d17.411125802122857!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3
```

```
m3!1m2!1s0x3bcb9963b2a5d0a7%3A0x1f10a3b1a2eef974!2sMantha%20Tech%20Solutions%20
Private%20Limited!5e0!3m2!1sen!2sin!4v1718270997552!5m2!1sen!2sin" width="600"
height="450" style="border:0;"

         allowfullscreen="" loading="lazy"
referrerpolicy="no-referrer-when-downgrade"></iframe>

    </body>

</html>
```

Ex:- Embed a video

```
<html>

    <body>

        <h1>Youtube Video </h1>

        <iframe width="560" height="315"
src="https://www.youtube.com/embed/SAqi7zmW1fY?si=hqPpezPKezbfKnac"

         title="YouTube video player" frameborder="0" allow="accelerometer;
autoplay; clipboard-write;

         encrypted-media; gyroscope; picture-in-picture; web-share"

         referrerpolicy="strict-origin-when-cross-origin"
allowfullscreen></iframe>
```

```
    </body>

</html>
```

Quotation Tag :-

Html quotation elements are used to insert quoted text in a webpage . It means some part of text that is different from the rest of the portion .

Some quotation elements elements are as follows

- <abbr> :- defines abbreviation or acronym .
- <address> :- defines contact info for the owner of the document.
- <bdo> :- defines text direction (right to left or left to right) .Bi-directional override
- <q> :- defines shortline quotation , enclosed with quotation marks .

Ex:-

```
<html>

    <body>

        <h1>Quotation tag </h1>

        <p>React is Javascript library </p> <br>
```

```
        <p>

            <bdo dir="rtl">

                Angular is Javascript Framework

            </bdo>

        </p>

        <br>

        <abbr title="Spring tool suite">STS</abbr>

        <blockquote>

            <p>

                Learning Management system app includes Java , Aws ,
Devops course

            </p>

        </blockquote>

       <p>The best platform to learn react is <cite>Geeks for
Geeks</cite> start your course today and excel in Front

    end development</p>
```

```
        <address>
            <p>
                Address:<br />
                H.No.- 3-8-876, Uppal
                Park,<br />
                Sector-142, Noida Uttar Pradesh -
                201305
            </p>
            <q>Advanced version of react is NextJs</q>
        </address>
    </body>
</html>
```

Marquee Tag :- Marquee tag creates scrolling text or image effect in a webpage . It allows content to move horizontally or vertically across the screen, providing a simple way to add dynamic

movement to elements. It includes attributes like direction to specify whether the content moves left, right, up, or down.

Marquee tag is deprecated in HTML 5

Syntax:-

```
<marquee>

 Content

</marquee>
```

Ex:-

```
<html>

    <body>

        <body>

            <div class="main">

                <marquee

                        direction="left" loop="">
```

```
                <div >

                    GeeksforGeeks

                </div>

                <div >

                    A computer science portal for geeks

                </div>

            </marquee>

        </div>

    </body>

</html>
```

Semantic Elements :- A semantic element clearly describes its meaning to both the browser and the developer.

Examples of non-semantic elements: and - Tells nothing about its content.

Examples of semantic elements: <form>, <table>, and <article> - Clearly defines its content.

Let us see some semantic elements used in a webpage

- <article>
- <aside>
- <details>
- <figcaption>
- <figure>
- <footer>
- <header>
- <main>
- <mark>
- <nav>
- <section>
- <summary>
- <time>

Ex :-

```
<html>

    <head>

        <style>

            aside {

                width: 50%;

                padding: 15px;

                margin: 15px;

                color: black;

                background-color: blanchedalmond;
```

```
                float: right;



            }
        </style>
    </head>
    <body>
        <article>
        <p>Used by some of the world's largest companies,
             Next.js enables you to create high-quality web
applications with the power of
              React components. Used by some of the world's largest
companies,
              Next.js enables you to create high-quality web
applications with the power of
               React components</p>
            </article>
        <aside>
            Used by some of the world's largest companies,
             Next.js enables you to create high-quality web
applications with the power of React components.
        </aside>
        <abbr title="World Wide Web consortium">W3C</abbr>
        <address>
            <p>Address <br>
            H.No. 5-8-887, VNC Colony <br>
```

```
        Uppal , Hyderabad</p>
        </address>

        <details>
            <summary>Components</summary>
            <p>
                Breaking down in to small modules are referred as
components
                Breaking down in to small modules are referred as
components
                Breaking down in to small modules are referred as
components

            </p>
        </details>
    </body>
</html>
```

Aside :- aside tag is used to display a content that is different or is separated from the main content .

Article :- article tag is used to define an article in a web page . In real time we use article tags to define blog posts , news articles etc .

Details :- It specifies additional details that the user can open and close on demand , basically details and summary tag are used together .

Summary :- Summary tag specifies heading for the details tag , it hides or displays text based on user demand .

As mentioned above we use summary tag for heading , when user clicks on that it hides and shows the details of that , for details we use details tag . This is mentioned as an example in the above code .

Address :- address tag is used to display the address when we use this tag , it shows the address in italic font by default .

Footer :- footer tag is used to define footer .

Footer tag is explained in the Grouping concept ,check the reference form there .

Task :-

Design a web page that embeds a map , video in four different sections

(take four sections , and embed a map in first , video in second follow the same pattern for next sections)

Design a webpage with following requirements

- **Navbar in header section**
- **Home , Contact , About options (pages)**
- **Design each page separately .**
- **In Home page display content along with images**
- **In Contact page display contact details along with location**
- **In About page display an article about the web page**
- **Also include footer section**

- **Display the list of items in the table or list according to the topics you choose in about page after article .**
- **Topics :- Training Institute for online courses ,**

Or

Groceries Delivery site

Choose any one topic according to your convenience

- **Also mention the price of course and grocery items if you are using table**

Note :- Remember to use all the concepts and tags of what we discussed so far . All the basics of HTML

Do the above task on Tuesday , and submit it on Wednesday

HTML 5

Overview :-

HTML5 provides details of all 40+ HTML tags including audio, video, header, footer, data, datalist, article etc. This HTML tutorial is designed for beginners and professionals.

HTML5 is the next version of HTML. Here, you will get some brand new features which will make HTML much easier. These new introducing

features make your website layout clearer to both website designers and users. There are some elements like <header>, <footer>, <nav> and <article> that define the layout of a website.

Why HTML5 ?

It is enriched with advanced features which makes it easy and interactive for designer/developer and users.It allows you to play a video and audio file.

It allows you to draw on a canvas.

It facilitates you to design better forms and build web applications that work offline.

It provides you advanced features for that you would normally have to write JavaScript to do.

The most important reason to use HTML 5 is, we believe it is not going anywhere. It will be here to serve for a long time according to W3C recommendation.

HTML5 Attributes :-

Some attributes are defined globally and can be used on any element, while others are defined for specific elements only. All attributes have a name and a value and look like as shown below in the example.

Ex :-

```
<div class = "example">...</div>
```

Attributes may only be specified within **start tags** and must never be used in **end tags**.

HTML5 attributes are case insensitive and may be written in all uppercase or mixed case, although the most common convention is to stick with lowercase.

Standard Attributes :-

- **align :-** horizontally aligned tags
- **background :-** places a background image behind an element
- **class :-** Classifies an element for use with cascading style sheets
- **Contenteditable :-** Specifies if the user can edit the content or not
- **draggable :-** Specifies whether or not a user is allowed to drag an element .
- **height :-** Specifies the height of a table , image or table cells .
- **hidden :-** Specifies whether the element should be visible or not
- **id :-** Names an element for use with CSS
- **Item :-** used to group elements
- **Itemprop :-** used to group items
- **spellcheck :-** Specifies if the element must have its spelling or grammar checked.
- **style :-** specifies an inline style for an element
- **valign :-** aligns elements vertically
- **width :-** specifies width of tables , table cells , images etc

Events :-

When users visit your website, they perform various activities such as clicking on text and images and links, hover over defined elements, etc. These are examples of what JavaScript calls **events**.

We can write our event handlers in Javascript or VBscript and you can specify these event handlers as a value of event tag attribute. The HTML5 specification defines various event attributes as listed below −

We can use the following set of attributes to trigger any **javascript** or **vbscript** code given as value, when there is any event that takes place for any HTML5 element.

Let us see some event attributes that are mostly used with HTML5

- **onoffline :-** triggers when document goes offline
- **onabort :-** triggers on an abort event
- **onchange :-** triggers when an element changes
- **onclick :-** triggers on a mouse click
- **ondblclick :-** triggers on mouse double
- **ondrag :-** triggers when an element is dragged
- **ondragend :-** triggers at the end of drag operation
- **ondragstart :-** triggers at the start of drag operation
- **ondragenter :-** Triggers when an element has been dragged to a valid drop target.
- **ondragleave :-** Triggers when an element leaves a valid drop target .
- **ondragover :-** Triggers when an element is being dragged over a valid drop target .
- **ondrop :-** Triggers when dragged element is being dropped .

- **ondurationchange :-** triggers when the length of the media is changed .
- **onemptied :-** Triggers when a media resource element suddenly becomes empty.
- **onended :-** triggers when media has reach end
- **onformchanges :-** triggers when a form changes
- **onforminput :-** triggers when a form gets user input
- **onmousedown :-** triggers when a mouse button is pressed
- **onmousemove :-** triggers when mouse pointer moves
- **onmouseout :-** triggers when the mouse pointer moves out of an element.
- **onmouseover :-** triggers when the mouse pointer moves over an element .

Ex :-

```
<html>

    <body ononline="offline()">

        <h1>Offline example</h1>

        <input type="text" value="hello" onchange="function1()">

        <button onclick="function2()">Submit</button>

         <p ondblclick="function3()">double click this paragraph to call a function</p>



         <p  draggable="true" ondrag="function4()" >drag this paragraph to call a function</p>
```

```
        <script>
            function function1() {
                alert("The input value has changed ");


            }
            function function2() {
                  alert("when you click on button function2 is called and
displayes alert message ");
            }
            function function3() {
                    alert("when you  double click on paragraph element
function3 is called and displayes alert message ");


            }
            function function4() {
                  alert("when you  drag  this  paragraph element function4
is called and displayes alert message ");


            }


          function offline(){
                alert("Browser works in offline mode ");
                console.log("online mode ");
```

```
            }
        </script>
    </body>
</html>
```

CSS (Cascading Style Sheets)

Introduction to css , Css Syntax
CSS fonts
CSS colors , css properties
Combinators
Comments , align
Borders , index , lists
Padding , margin
Animation
Media
CSS selectors
CSS text , Display
width , contents , overflow
CSS Rule Set
CSS transforms
Margin

Introduction :-

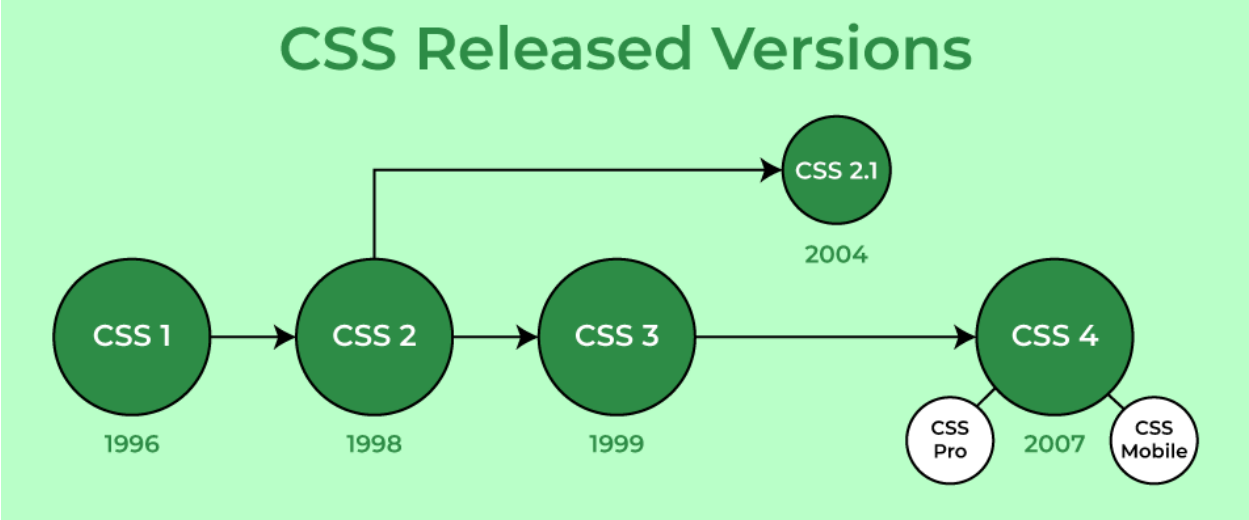
CSS (Cascading Style Sheets) is a language designed to simplify the process of making web pages presentable. It allows you to apply styles to HTML documents, describing how a web page should look by prescribing colors, fonts, spacing, and positioning. CSS provides developers and designers with powerful control over the presentation of HTML elements.

HTML uses tags and CSS uses rulesets. CSS styles are applied to the HTML element using selectors. CSS is easy to learn and understand, but it provides powerful control over the presentation of an HTML document.

Why CSS ?

- **Saves Time:** Write CSS once and reuse it across multiple HTML pages.
- **Easy Maintenance:** Change the style globally with a single modification.
- **Search Engine Friendly:** Clean coding technique that improves readability for search engines.
- **Superior Styles:** Offers a wider array of attributes compared to HTML.

Css Versions :-



Css Syntax :-

Css consists of style rules that are interpreted by the browser and applied to the corresponding elements .

What is a Style Rule Set ?

A Style Rule Set includes a selector and a declaration block

Selector :- targets the specific html element to apply the styles

Declaration :- Combination of a property and its corresponding value

Ex:-

<h1>Hello World</h1>

//Style

h1{ color: “white”;font-size:12px ; } -> style rule set

In above example

h1-> selector (that specifies element to which the style is to be applied)

{ color: “white”;font-size:12px ; } -> this is called as declaration block

It contains one or more declarations each separated by semicolons and it includes css properties in name value format , always css declarations are separated by semicolons and enclosed in curly braces .

Ex :- with css

```
<html>
    <head>
        <style>
            h1{
                text-align: center;
                border: 1px solid black;
                background-color:blue;
                color: aliceblue;
            }
        </style>
```

```
    </head>
    <body>
        <h1>Mantha Tech Solutions</h1>


    </body>
</html>
```

Types of CSS

1. **Inline CSS :-** Inline CSS involves applying styles directly to individual HTML elements using the style attribute. This method allows for specific styling of elements within the HTML document, overriding any external or internal styles.

Ex:-

```
<html>
    <head>
        <style>
            h1{
                text-align: center;
                border: 1px solid black;
```

```
                background-color:blue;
                color: aliceblue;
            }
        </style>
    </head>
    <body>
        <h1>Mantha Tech Solutions</h1>
        <p style="color:black;
                font-style: italic;
                text-align: center;
                font-size: 30px;
        ">We Provide Development and Services.</p>

    </body>
</html>
```

2. **Internal or Embedded CSS :- It** is defined within the HTML document's <style> element. It applies styles to specific HTML elements, The CSS rule set should be within the HTML file in the head section i.e. the CSS is embedded within the <style> tag inside the head section of the HTML file.

Ex:-

Same as previous one

3. **External CSS :-** It contains separate CSS files that contain only style properties with the help of tag attributes (For example class, id, heading, … etc). CSS property is written in a separate file with a .css extension and should be linked to the HTML document using a link tag. It means that, for each element, style can be set only once and will be applied across web pages.

Ex:-

```
<!DOCTYPE html>

<html>



<head>

    <title>External CSS</title>

    <link rel="stylesheet" href="style.css">

</head>



<body>

    <div class="main">

        <div class="sub">We Provide services such as web and mobile app Development</div>
```

```
        <div id="para">
            we follow all the standards of Agile
        </div>
    </div>
</body>


</html>
```

Style.css

```
body {
    background-color:antiquewhite
}

.main {
    text-align: center;
}

.sub {
```

```
    color:aqua;
    font-size: 50px;
    font-weight: bold;
}

#para {
    font-style: bold;
    font-size: 20px;
    color: aqua;
}
```

Important Points :-

- Inline CSS has the highest priority, then comes Internal/Embedded followed by External CSS which has the least priority. Multiple style sheets can be defined on one page. For an HTML tag, styles can be defined in multiple style types and follow the below order.
- As Inline has the highest priority, any styles that are defined in the internal and external style sheets are overridden by Inline styles.

- Internal or Embedded stands second in the priority list and overrides the styles in the external style sheet.
- External style sheets have the least priority. If there are no styles defined either in inline or internal style sheets then external style sheet rules are applied for the HTML tags.

Disadvantages of CSS :-

- **Lack of security :-** Css does not provide security , it is vulnerable to Cross - SIte Scripting attacks (XSS attacks) , be cautious while using css in real time
- **Performance :-** CSS impacts performance of web page badly. It takes more time to load the application or web page because we need to write long lines of codes if we onlyB css for styling .
- **Browser Compatibility :-** CSS renders differently on various browsers it leads to inconsistency in results and outputs , in such case developers need to write browser specific code .

Css Comments :-

Comments are essential for documenting code, providing clarity during development and maintenance. They begin with `/*` and end with `*/`, allowing for multiline or inline annotations. Browsers ignore comments, ensuring they don’t affect the rendering of web pages.

Syntax :-

```
/* content */
```

Comments can be **single-line** or **multi-line**. The **/* */** comment syntax can be used for both single and multiline comments. We may use **<!– –>** syntax for hiding in CSS for older browsers, but this is no longer recommended for use.

Measurement Units in CSS :-

CSS has several units for expressing a length. Many CSS properties take “length” values , such as width , margin, padding, font-size etc

Length is a number followed by a length unit such as 10px, 2em etc

A whitespace cannot appear in between number and unit , if the value is 0 unit can be omitted . For some css properties negative lengths are allowed . There are two types of length ,

1. **Absolute Length :-** The Absolute length units are fixed and a length expressed in any of these will appear as exactly that size .

   But these are not recommended to use on screen , bucz screen sizes vary so much these can be used if the output medium is known before such as print layout .

   Units are cm, mm , in , px* , pt , pc

   1pt = 1/72 inch

2. **Relative Length :-** Relative length units specify a length relative to another length property . These units scale better between different mediums .

   Units are

   em - > 2em means 2 times of the size of current font

   rem - > relative to the font size of the root element .

   % - > relative to the parent element

Difference :-

- Unlike absolute units , relative units are not fixed , their values are relative to another value .
- It means when that other value changes , the relative unit value also changes

CSS Colors :-

CSS Colors are an essential part of web design, providing the ability to bring your HTML elements to life. This feature allows developers to set the color of various HTML elements, including font color, background color, and more.

Color Format :-

Colors in css can be specified by following methods

- Hexadecimal colors

- Hexadecimal colors with transparency
- RGB colors
- RGBA colors
- HSL colors
- HSLA colors

Hexadecimal 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. All values must be between 00 and FF.

For example, the #0000ff value is rendered as blue, because the blue component is set to its highest value (ff) and the others are set to 00.

Hexadecimal color with transparency :-

A hexadecimal color is specified with: #RRGGBB. To add transparency, add two additional digits between 00 and FF.

Ex:- #00ff0080

RGB Colors :-

An RGB color value is specified with the rgb() function, which has the following syntax:

rgb(red, green, blue)

Each parameter (red, green, and blue) defines the intensity of the color and can be an integer between 0 and 255 or a percentage value (from 0% to 100%).

For example, the rgb(0,0,255) value is rendered as blue, because the blue parameter is set to its highest value (255) and the others are set to 0.

Also, the following values define equal color: rgb(0,0,255) and rgb(0%,0%,100%).

Ex :- rgb(0,0,255)

RGBA Colors :-

RGBA color values are an extension of RGB color values with an alpha channel - which specifies the opacity of the object.

An RGBA color is specified with the rgba() function, which has the following syntax:

rgba(red, green, blue, alpha)

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

Ex :- rgba(0,0,255,0.5)

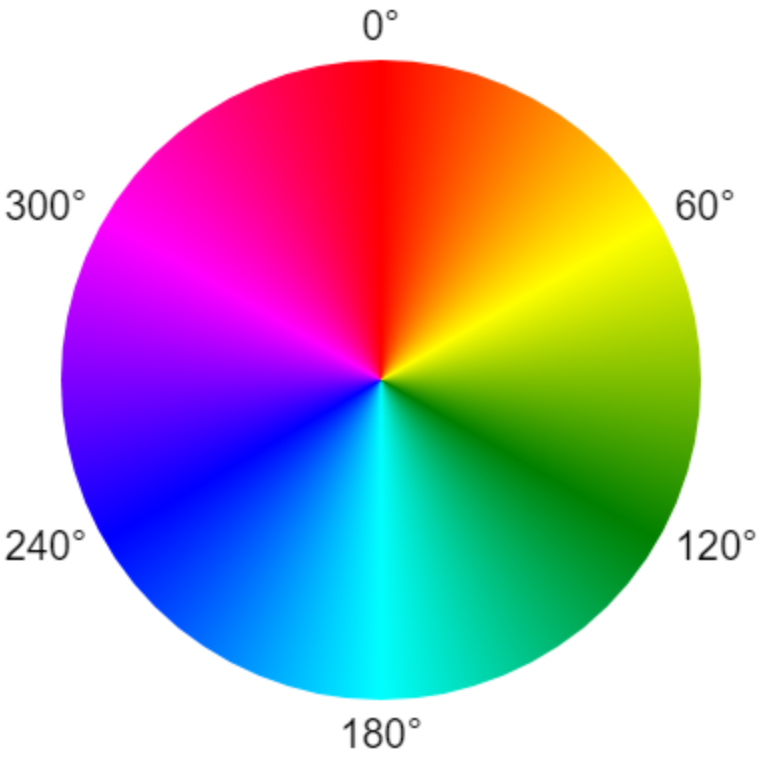
HSL colors :-

HSL stands for hue, saturation, and lightness - and represents a cylindrical-coordinate representation of colors.

An HSL color value is specified with the hsl() function, which has the following syntax:

hsl(hue, saturation, lightness)

Hue is a degree on the color wheel (from 0 to 360) - 0 (or 360) is red, 120 is green, 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, 100% is white.



Ex :-

```
background-color: hsl(0, 100%, 50%);
```

HSLA colors :-

HSLA is an extension of HSL color , and a means alpha parameter is added in this function , its specified as hsla() function .

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

Ex:-

```
background-color: hsla(0, 100%, 50%,0.3);
```

CSS Fonts :-

CSS fonts are used to style the text within HTML elements. The font-family property specifies the typeface, while font-size, font-weight, and font-style control the size, thickness, and style of the text. Combining these properties enhances the typography and readability of web content.

What is typography ?

Typography is the art of and technique of arranging type (text) in a way that is visually appealing and effective for communication .

It involves selecting and using typefaces (fonts) , font size , line spacing , and other text elements to create a clear and readable text .

The following are the types of CSS font properties :-

- CSS font-family property
- CSS font-style property
- CSS font-weight property
- CSS font-variant property
- CSS font-size property
- CSS font-stretch property
- CSS font-kerning property

CSS Margin property :-

The margin property sets the margins for an element, and is a shorthand property for the following properties:

- margin-top
- margin-right
- margin-bottom
- margin-left

If the margin property has four values:

- margin: 10px 5px 15px 20px;
    - top margin is 10px
    - right margin is 5px

    - bottom margin is 15px
    - left margin is 20px

If the margin property has three values:

- margin: 10px 5px 15px;
    - top margin is 10px
    - right and left margins are 5px
    - bottom margin is 15px

If the margin property has two values:

- margin: 10px 5px;
    - top and bottom margins are 10px
    - right and left margins are 5px

If the margin property has one value:

- margin: 10px;
    - all four margins are 10px

Note: Negative values are allowed.

Negative values in margin property :-

It is possible to give margins a negative value. This allows you to draw the element closer to its top or left neighbor , or draw its right and bottom neighbor closer to it .

margin - inline property :-

Css margin property specifies the margin at the start and end in inline direction , shorthand properties for this as as follows

- margin-inline-start
- margin-inline-end

Values for the margin-inline property can be set in different ways:

If the margin-inline property has two values:

- margin-inline: 10px 50px;
    - margin at start is 10px
    - margin at end is 50px

If the margin-inline property has one value:

- margin-inline: 10px;
    - margin at start and end is 10px

Ex:-

```
<html>
    <head>
        <style>
            #p1{
                font-size: 2em;
                color:aliceblue;
                background-color: hsla(0, 89%, 0%,1.0);
                margin:10px;
```

```
}
#p2{
    font-family:'Times New Roman', Times, serif;
    font-size: 70px;
    font-weight: bold;
    font-style: italic;
    font-variant: small-caps;
    font-stretch:condensed;
     margin:-15px;
     margin-left: 200px;

}
div{
    background-color: rgb(255,0,0);
    width:25%;
    height: 200px;
    float: left;

}
#div2{
    background-color: aqua;
    border: solid black 1px;
     margin-inline: 35px;
```

```
        }
    </style>
</head>
<body>
    <p id="p1">HSL Method of applying colors</p>

    <p id="p2">Font properties </p>

    <div >First section</div>

    <div id="div2">Second section</div>

    <div >Third section</div>

</body>
</html>
```

margin-block property :-

The `margin-block` property specifies the margin at the start and end in the block direction, and is a shorthand property for the following properties:

- `margin-block-start`

- `margin-block-end`

Values for the `margin-block` property can be set in different ways:

If the margin-block property has two values:

- margin-block: 10px 50px;
    - margin at start is 10px
    - margin at end is 50px

If the margin-block property has one value:

- margin-block: 10px;
    - margin at start and end is 10px

The CSS `margin-block` and `margin-inline` properties are very similar to CSS properties `margin-top`, `margin-bottom`, `margin-left` and `margin-right`, but the margin-block and margin-inline properties are dependent on block and inline directions.

Note: The related CSS property writing-mode defines block direction. This affects where the start and end of a block is and the result of the margin-block property. For pages in English, block direction is downward and inline direction is left to right.

Padding property :-

An element's padding is the space between its content and its border.

The padding property is a shorthand property for:

- padding-top
- padding-right

- padding-bottom
- padding-left

Note: Padding creates extra space within an element, while margin creates extra space around an element.

This property can have from one to four values.

If the padding property has four values:

- padding:10px 5px 15px 20px;
    - top padding is 10px
    - right padding is 5px
    - bottom padding is 15px
    - left padding is 20px

If the padding property has three values:

- padding:10px 5px 15px;
    - top padding is 10px
    - right and left padding are 5px
    - bottom padding is 15px

If the padding property has two values:

- padding:10px 5px;
    - top and bottom padding are 10px
    - right and left padding are 5px

If the padding property has one value:

- padding:10px;
    - all four paddings are 10px

Note: Negative values are not allowed.

padding-block property :-

An element's padding-block is the space from its border to its content in the block direction, and it is a shorthand property for the following properties:

- padding-block-start
- padding-block-end

Values for the padding-block property can be set in different ways:

If the padding-block property has two values:

- padding-block: 10px 50px;
    - padding at start is 10px
    - padding at end is 50px

If the padding-block property has one value:

- padding-block: 10px;
    - padding at start and end is 10px

padding-inline property :-

An element's padding-inline is the space from its border to its content in the inline direction, and it is a shorthand property for the following properties:

- padding-inline-start

- padding-inline-end

Values for the padding-inline property can be set in different ways:

If the padding-inline property has two values:

- padding-inline: 10px 50px;
    - padding at start is 10px
    - padding at end is 50px

If the padding-inline property has one value:

- padding-inline: 10px;
    - padding at start and end is 10px

Ex :-

```
<html>

    <head>

        <style>

            div{

                background-color: aqua;

                font-size: 2rem;

            }

            .div2{

                border: solid 1px black;

                 margin-block: 25px 50px;
```

```
                padding-inline-end: 70px;
                font-size: 2rem;
            }
        </style>
    </head>
    <body>
        <div >Section one</div>
        <div class="div2">Section two</div>
        <div>Section three</div>
    </body>
</html>
```

Border property :-

The border-style property sets the style of an element's four borders. This property can have from one to four values.

Examples:

- border-style: dotted solid double dashed;
    - top border is dotted
    - right border is solid
    - bottom border is double
    - left border is dashed

- border-style: dotted solid double;
    - top border is dotted
    - right and left borders are solid
    - bottom border is double

- border-style: dotted solid;
    - top and bottom borders are dotted
    - right and left borders are solid

- border-style: dotted;
    - all four borders are dotted

The CSS border properties allow you to specify the style, width, and color of an element's border.

border style :-

The border-style property specifies what kind of border to display.

The following values are allowed:

- dotted - Defines a dotted border
- dashed - Defines a dashed border
- solid - Defines a solid border
- double - Defines a double border
- groove - Defines a 3D grooved border. The effect depends on the border-color value
- ridge - Defines a 3D ridged border. The effect depends on the border-color value

- inset - Defines a 3D inset border. The effect depends on the border-color value
- outset - Defines a 3D outset border. The effect depends on the border-color value
- none - Defines no border
- hidden - Defines a hidden border

The border-style property can have from one to four values (for the top border, right border, bottom border, and the left border).

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 predefined values: thin, medium, or thick

border-color :-

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

The color can be set by:

- name - specify a color name, like "red"
- HEX - specify a HEX value, like "#ff0000"
- RGB - specify a RGB value, like "rgb(255,0,0)"
- HSL - specify a HSL value, like "hsl(0, 100%, 50%)"
- transparent

Note: If border-color is not set, it inherits the color of the element.

Background Property :-

The background property is a shorthand property for:

- background-color
- background-image
- background-position
- background-attachment
- background-repeat
- background-clip
- background-size
- background-origin

Ex:-

```
<!D<!DOCTYPE html>

<html>

<head>

<style>

body {


  background: lightblue url("image.png") no-repeat fixed  center  ;

}

</style>

</head>
```

```
<body>

<h1>The background Property</h1>

<p>This is some text</p>
<p>This is some text</p>
<p>This is some text</p>
<p>This is some text</p>
<p>This is some text</p>
<p>This is some text</p>
<p>This is some text</p>
<p>This is some text</p>
<p>This is some text</p>
<p>This is some text</p>
<p>This is some text</p>
<p>This is some text</p>
<p>This is some text</p>
<p>This is some text</p>
<p>This is some text</p>
<p>This is some text</p>
<p>This is some text</p>
<p>This is some text</p>
<p>This is some text</p>
```

```
<p>This is some text</p>

<p>This is some text</p>

<p>This is some text</p>

<p>This is some text</p>

<p>This is some text</p>

<p>This is some text</p>



</body>

</html>
```

Try also with other examples for this topic

Display Property :-

The display property specifies the display behavior (the type of rendering box) of an element.

The following are the short hand properties for display :-

- display-inline:- displays an element as inline element
- display-block:- displays am element as block element
- display-inline-block:- it considers element as inline element
- display-flex:- displays element as block level flex container
- display-grid :- displayes element as block - level grid container
- display-inline-flex:- displays an element as inline-flex level container
- display-inline-grid :- displays an element as inline-grid level container
- display-list-item:- it let's the element behave as list item

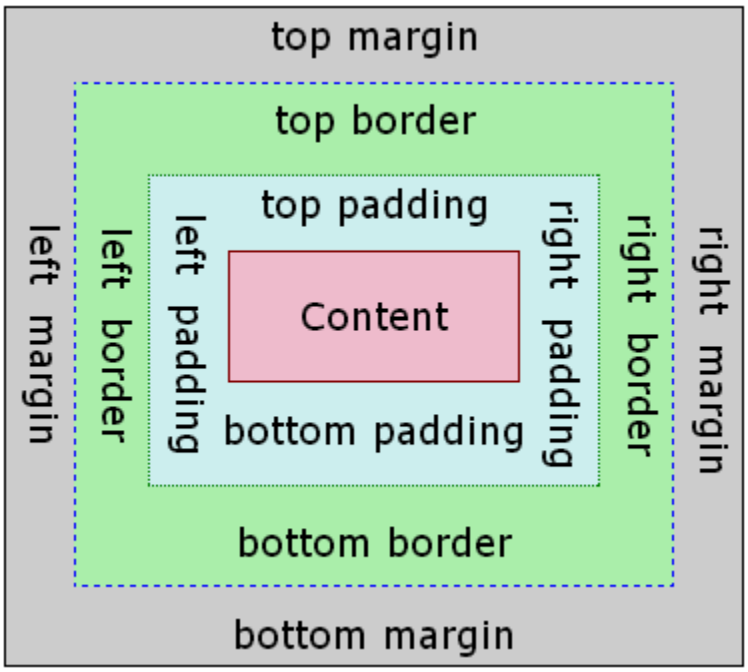
- display-run-in :- displays an element as either block or inline element
- display-inline-table:- displays element as inline level table

Try example with all the properties and apply inline table with table example

Box Model :-

In CSS, the term "box model" is used when talking about design and layout.

The CSS box model is essentially a box that wraps around every HTML element. It consists of: content, padding, borders and margins. The image below illustrates the box model



Explanation of the different parts:

- Content - The content of the box, where text and images appear

- Padding - Clears an area around the content. The padding is transparent
- Border - A border that goes around the padding and content
- Margin - Clears an area outside the border. The margin is transparent

The box model allows us to add a border around elements, and to define space between elements.

Example of box model (demonstration)

```
<!DOCTYPE html>

<html>

<head>

<style>

div {

  background-color: lightgrey;

  width: 300px;

  border: 15px solid green;

  padding: 50px;

  margin: 20px;

}

</style>
```

```
</head>

<body>


<h2>Demonstrating the Box Model</h2>


<p>The CSS box model is essentially a box that wraps around every HTML element. It consists of: borders, padding, margins, and the actual content.</p>


<div>This text is the content of the box. We have added a 50px padding, 20px margin and a 15px green border. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</div>


</body>

</html>
```

Setting Width and Height of Element :-

In order to set the width and height of an element correctly in all browsers, you need to know how the box model works.

When you set the width and height properties of an element with CSS, you just set the width and height of the content area. To calculate the total width and height of an element, you must also include the padding and borders.

Here is the calculation:

320px (width of content area)

+ 20px (left padding + right padding)

+ 10px (left border + right border)

= 350px (total width)

50px (height of content area)

+ 20px (top padding + bottom padding)

+ 10px (top border + bottom border)

= 80px (total height)

The total width of an element should be calculated like this:

Total element width = width + left padding + right padding + left border + right border

The total height of an element should be calculated like this:

Total element height = height + top padding + bottom padding + top border + bottom border

Css Float Property :-

The float property specifies whether an element should float to the left, right, or not at all.

Note: Absolutely positioned elements ignore the float property

```
<!DOCTYPE html>

<html>

<head>

<style>

img {

  float: right;

}

</style>

</head>

<body>

<h1>The float Property</h1>


<p>In this example, the image will float to the right in the text, and the text in the paragraph will wrap around the image.</p>


<p><img src="pineapple.jpg" alt="Pineapple"
style="width:170px;height:170px;margin-left:15px;">
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed 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>

Justify content property :-

The justify-content property aligns the flexible container's items when the items do not use all available space on the main-axis (horizontally).

Tip: Use the align-items property to align the items vertically

The `***justify-items***` property in CSS is important for aligning items within a grid container along the inline (row) axis. It allows you to control the alignment of grid items in a grid container when they do not explicitly position themselves using grid-area or similar properties.

Similar to `***align-items***` ***property*** for flex containers, ***justify-items*** enables you to align grid items horizontally within their grid areas.

CSS Outline property :-

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

An outline is a line that is drawn around elements, OUTSIDE the borders, to make the element "stand out".

CSS has the following outline properties:

- outline-style
- outline-color
- outline-width
- outline-offset
- outline

Note :-

Note: Outline differs from borders! Unlike borders, the outline is drawn outside the element's border, and may overlap other content. Also, the outline is NOT a part of the element's dimensions; the element's total width and height is not affected by the width of the outline

CSS Outline styles :-

The outline-style property specifies the style of the outline, and can have one of the following values:

- dotted - Defines a dotted outline
- dashed - Defines a dashed outline
- solid - Defines a solid outline
- double - Defines a double outline
- groove - Defines a 3D grooved outline
- ridge - Defines a 3D ridged outline
- inset - Defines a 3D inset outline
- outset - Defines a 3D outset outline
- none - Defines no outline
- hidden - Defines a hidden outline

Ex:-

```
<html>

    <head>

        <style>

            .dotted{

                outline: 5px double blue;

                color: green;

                text-align: center;

            }

            p{

                outline-style: groove;

                border-style: ridge;
```

```
            }
        </style>
    </head>
    <body>
        <h1>Outline property</h1>
        <p class="dotted">Computer Science and Engineering</p>



        <p >Computer Science and Engineering</p>


    </body>
</html>
```

Try with all properties

Overflow Property :-

The overflow property specifies what should happen if content overflows an element's box.

This property specifies whether to clip content or to add scrollbars when an element's content is too big to fit in a specified area.

Note: The overflow property only works for block elements with a specified height.

Ex:-

```
<html>
    <head>
```

```
<style>
    div.ex1{
        background-color: aqua;
        width: 110px;
        height: 110px;
        overflow: scroll;
        margin-left: 50px;
    }
    div.ex2{
        background-color:red;
        width: 110px;
        height: 110px;
        overflow:hidden ;
        margin-left: 50px;
    }
    div.ex3{
        background-color:greenyellow;
        width: 110px;
        height: 110px;
        overflow:auto;
        margin-left: 50px;
    }
```

```
            /* default it is visible */
            div.ex4{
                background-color:greenyellow;
                width: 110px;
                height: 110px;
                overflow:visible;
                margin-left: 50px;
            }




        </style>
    </head>
    <body>
        <h1>Overflow Property</h1>
        <p>Overflow property specifies whether to clip the content or to
add scrollbars , when an elements content
            is too big to fit in a specified container
        </p>

        <h2>overflow : scroll</h2>
```

```
        <div class="ex1">

            Overflow property specifies whether to clip the content or to
add scrollbars , when an elements content

            is too big to fit in a specified container

        </div>



        <h2>overflow : hidden</h2>

        <div class="ex2">

            Overflow property specifies whether to clip the content or to
add scrollbars , when an elements content

            is too big to fit in a specified container



        </div>

        <h2>overflow : auto</h2>

        <div class="ex3">

            Overflow property specifies whether to clip the content or to
add scrollbars , when an elements content

            is too big to fit in a specified container

        </div>



        <h2>overflow : clip</h2>

        <div class="ex4">
```

```
            Overflow property specifies whether to clip the content or to
add scrollbars , when an elements content

            is too big to fit in a specified container

        </div>

    </body>

</html>
```

Css Navigation Bar :-

A navigation bar needs standard HTML as a base.

In our examples we will build the navigation bar from a standard HTML list.

A navigation bar is basically a list of links, so using the <ul> and <li> elements makes perfect sense

Ex:- Vertical Navigation bar

```
<html>

    <head>

        <style>

            ul{

                list-style-type: none;

                margin: 0;

                padding: 0;

                width: 200px;
```

```
        }
        li a {
            display: block;
            width: 60px;
            padding: 8px 16px;
            background-color:beige;
            text-decoration: none;
        }
        li a:hover {
            background-color: black;
            color: aliceblue;
        }

    </style>
</head>
<body>
    <ul>
        <li><a href="#home">Home</a></li>
        <li><a href="#news">News</a></li>
        <li><a href="#contact">Contact</a></li>
        <li><a href="#about">About</a></li>
    </ul>
```

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

Css navigation Bar (horizontal)

There are two ways to create a horizontal navigation bar. Using inline or floating list items.One way to build a horizontal navigation bar is to specify the <li> elements as inline.

Ex:-

```
<html>
    <head>
        <style>
            body{
                overflow: scroll;
            }
            ul{
                list-style-type: none;
                margin: 0;
                padding: 0;
                overflow: hidden;
                background-color: aqua;
                border: 1px solid #e7e7e7;
            }
```

```
li a{
    text-decoration: none;
    display: block;
    color: white;
    text-align: center;
    padding: 14px 16px;
    float: left;
    position: sticky;
    position: -webkit-sticky;
    top: 0;

}
/* li a:hover{
    background-color:black;
} */
/* li a:hover:not(.active) {
background-color: #ddd;
 }
li a.active {
    color: white;
    background-color: #04AA6D;
}
```

```
 */


li a:hover {
  background-color: #111;
}


/* .active {
  background-color: #4CAF50;
} */
        </style>
    </head>
    <body>
        <ul>
            <li><a  href="#home">Home</a></li>
            <li><a href="#news">News</a></li>
            <li><a href="#contact">Contact</a></li>
            <li><a href="#about">About</a></li>
        </ul>
    </body>
</html>
```

CSS Scroll Property :-

This property is used for smooth animation of scroll position instead of a scroll jump. When the user clicks on links it smoothly performs its operation. It is used to visit one link to another link within a scrollable box.

Default Value: auto

Property:

- **smooth:** This property is used to specify the animation effect of the scroll between the elements within the scrollable box.
- **auto**: It is used to specify the straight jump scroll effect visit to one link to another link within a scrolling box.

Ex:-

```
<html>

    <head>

        <style>

            html {

                scroll-behavior: smooth;

            }



            #section1 {
```

```
            height: 800px;
            background-color: bisque;
        }
        #section2 {
            height: 600px;
            background-color:yellow;
        }

    </style>
</head>
<body>
    <div class="main" id="section1">
        <h2>Section 1</h2>
        <p> Click on the link to see the "smooth" scrolling effect</p>
        <a href="#section2">Click me to have smooth scroll to section
2 </a>

        <div class="main" id="section2">
            <h2>Section 2</h2>
```

```
                <a href="#section1">Click me to have smooth scroll to
section 1</a>



            </div>

        </div>

    </body>

</html>
```

Scroll-margin property :-

The scroll-margin property specifies the distance between the snap position and the container.

This means that when you stop scrolling, the scrolling will quickly adjust and stop at a specified distance between the snap position and the container.

The scroll-margin property is a shorthand property for the following properties:

- scroll-margin-top
- scroll-margin-bottom
- scroll-margin-left
- scroll-margin-right

Values for the scroll-margin property can be set in different ways:

If the scroll-margin property has four values:

- scroll-margin: 15px 30px 60px 90px;
    - top distance is 15px
    - right distance is 30px
    - bottom distance is 60px

    - left distance is 90px

If the scroll-margin property has three values:

- scroll-margin: 15px 30px 60px;
    - top distance is 15px
    - left and right distances are 30px
    - bottom distance is 60px

If the scroll-margin property has two values:

- scroll-margin: 15px 30px;
    - top and bottom distances are 15px
    - left and right distances are 30px

If the scroll-margin property has one value:

- scroll-margin: 10px;
    - all four distances are 10px

Ex :-

```
<html>

    <head>

        <style>

            .page{

                width: 270px;

                height: 278px;

                color: aliceblue;

                font-size: 50px;
```

```
            display: flex;
            align-items: center;
            justify-content: center;

        }
        .container{
            width: 300px;
            height: 300px;
            overflow-x: hidden;
            overflow-y: auto;

        }

    </style>
</head>
<body>
    <div class="container">
        <div class="page" style="background-color: aquamarine;
scroll-margin-top: 0px;">
            Scroll-margin
```

```
            </div>

            <div class="page" style="background-color:blueviolet;
scroll-margin-top: 20px;">

                Scroll-margin

            </div>

            <div class="page" style="background-color:brown;
scroll-margin-top: 40px;">

                Scroll-margin

            </div>

            <div class="page" style="background-color:blanchedalmond;
scroll-margin-top: 50px;">

                Scroll-margin

            </div>



        </div>

    </body>

</html>
```

CSS scroll-margin-block :-

The scroll-margin-block property specifies the distance in block direction, between the snap position and the container.

This means that when you stop scrolling, the scrolling will quickly adjust and stop at a specified distance in block direction, between the snap position and the container.

Values for the scroll-margin-block property can be set in different ways:

If the scroll-margin-block property has two values:

- scroll-margin-block: 10px 50px;
    - distance at start is 10px
    - distance at end is 50px

If the scroll-margin-block property has one value:

- scroll-margin-block: 10px;
    - distance at start and end is 10px

To see the effect from the scroll-margin-block property, the scroll-margin-block and scroll-snap-align properties must be set on the child elements, and the scroll-snap-type property must be set on the parent element.

Ex:-

CSS scroll-margin-inline:-

The scroll-margin-inline property specifies the distance in the inline direction, between the snap position and the container.

This means that when you stop scrolling, the scrolling will quickly adjust and stop at a specified distance in the inline direction, between the snap position and the container.

If the scroll-margin-inline property has two values:

- scroll-margin-inline: 20px 70px;
    - distance at start is 20px

    - distance at end is 70px

If the scroll-margin-inline property has one value:

- scroll-margin-inline: 20px;
    - distance at start and end is 20px

Ex:-

```
<html>
    <head>
        <style>
            .page{
                width: 278px;
                height: 296px;
                color: azure;
                font-size: 60px;
                display: flex;
                justify-content: center;
                align-items: center;
            }
            .container{
                width: 300px;
                height: 300px;
                overflow-x: hidden;
                overflow-y: auto;
```

```
            }



        </style>
    </head>
    <body>
        <div class="container">
            <div class="page" style="background-color: aqua;
scroll-margin-block: 90px;">
                Scroll margin block- 1
            </div>
            <div class="page" style="background-color:red;
scroll-margin-block: 90px;">
                Scroll margin block - 2
            </div>
            <div class="page" style="background-color:yellow;
scroll-margin-block: 90px;">
                Scroll margin block - 3
            </div>
            <div class="page" style="background-color:black;
scroll-margin-block: 90px;">
                Scroll margin block - 4
            </div>
        </div>
    </body>
</html>
```

Same example for block and inline just change the properties

JavaScript

Introduction :-

JavaScript is a *lightweight, cross-platform*, *single-threaded,* and *interpreted compiled* programming language. It is also known as the scripting language for web pages. It is well-known for the development of web pages, and many non-browser environments also use it.

JavaScript can be used for Client-side developments as well as Server-side developments. JavaScript is both an imperative and declarative type of language. JavaScript contains a standard library of objects, like Array , date and Math, and a core set of language elements like operators , control structures, and statements.

JavaScript is a high-level, interpreted programming language primarily used for making web pages interactive. It allows you to implement complex features on web pages, such as handling user interactions, manipulating the DOM (Document Object Model), and making asynchronous requests to servers.

Features :-

Client-Side Scripting: JavaScript code runs on the client-side (in the user's browser), allowing for dynamic content and interaction with the user interface. This enables web developers to create responsive and interactive websites.

Cross-platform Compatibility: JavaScript is supported by all major web browsers (Chrome, Firefox, Safari, Edge, etc.), making it a reliable choice for web development. It is also increasingly used beyond web browsers (e.g., server-side with Node.js, desktop applications with Electron.js).

Simple and Easy to Learn: JavaScript has a syntax similar to other programming languages like C and Java, making it relatively easy to pick up for developers familiar with these languages. Its dynamic typing and flexible syntax reduce the overhead of learning complex type systems.

Versatility: Apart from web development, JavaScript can be used for a variety of applications, including mobile app development (using frameworks like React Native), server-side scripting (with Node.js), and even for creating desktop applications.

Event-Driven and Asynchronous: JavaScript is inherently event-driven, meaning it can respond to user actions such as clicks, scrolls, and keyboard inputs. It also supports asynchronous programming, allowing tasks to run in the background without blocking the main program execution. This is crucial for handling tasks like fetching data from servers without freezing the user interface.

Rich Interfaces with DOM Manipulation: JavaScript allows developers to manipulate the Document Object Model (DOM) of a webpage, enabling dynamic changes to the content and structure of the page in response to user actions or other events.

Support for Libraries and Frameworks: JavaScript has a vast ecosystem of libraries and frameworks (e.g., React, Angular, Vue.js) that simplify complex tasks and provide pre-built solutions for common web development challenges. These tools enhance productivity and maintainability of JavaScript codebases.

Community and Resources: JavaScript benefits from a large and active community of developers, which contributes to the availability of tutorials, documentation, and open-source projects. This community support is invaluable for learning and troubleshooting JavaScript-related issues.

Security: While JavaScript runs on the client-side and can be manipulated by users, modern browsers have security measures in place to prevent malicious actions (like cross-site scripting attacks). Developers must follow best practices to ensure the security of their JavaScript applications.

Continuous Evolution: JavaScript evolves rapidly with new language features and updates (e.g., ECMAScript standards), ensuring that developers have access to modern programming capabilities and performance improvements.

Pros :-

Versatility: JavaScript can be used for both front-end (client-side) and back-end (server-side) development, thanks to frameworks like Node.js. This versatility allows developers to use a single language across different parts of their applications.

Ease of Learning: JavaScript has a relatively forgiving syntax compared to other programming languages, making it accessible for beginners. It doesn't require compilation and can be directly run in any web browser, simplifying the development and testing process.

Interactivity: JavaScript enables interactive web pages by allowing developers to respond to user actions in real-time without reloading the entire page. This improves user experience by creating dynamic and responsive interfaces.

Rich Interfaces: With JavaScript, developers can manipulate the DOM (Document Object Model) to dynamically change the content and styling of web pages. This ability to update parts of a page without full refreshes leads to smoother and more interactive user interfaces.

Large Ecosystem: JavaScript has a vast ecosystem of libraries (like jQuery, React, Angular, Vue.js) and frameworks that simplify complex tasks and

accelerate development. These tools provide pre-built components, extensive documentation, and strong community support.

Asynchronous Programming: JavaScript supports asynchronous programming, which allows tasks to run concurrently without blocking the main thread. This is crucial for handling operations like fetching data from servers or performing animations without freezing the user interface.

Performance: Modern JavaScript engines (like V8 in Chrome and Node.js) have significantly improved performance, making JavaScript a viable option for high-performance applications. Frameworks and tools built on JavaScript also optimize performance through efficient rendering and data handling.

Cross-platform Compatibility: JavaScript is supported by all major web browsers and can run on multiple platforms, including desktops, mobile devices, and servers. This cross-platform compatibility ensures consistent behavior across different environments.

Community and Support: JavaScript has a large and active community of developers who contribute to open-source projects, share knowledge through forums and tutorials, and provide assistance on platforms like Stack Overflow. This community support helps developers solve problems quickly and stay updated with best practices.

Continuous Improvement: JavaScript continues to evolve with regular updates to the ECMAScript standard (the specification that JavaScript follows). New features and improvements enhance language capabilities, improve performance, and address developer needs.

Cons:-

Browser Compatibility: Different web browsers may interpret JavaScript code differently, leading to inconsistencies in behavior and functionality across

platforms. Developers often need to write additional code or use polyfills to ensure compatibility with older browsers.

Security: Since JavaScript code runs on the client-side, it is inherently vulnerable to attacks like cross-site scripting (XSS). Developers need to implement security best practices, such as input validation and escaping output, to prevent malicious code execution.

Performance: While modern JavaScript engines have improved performance, JavaScript can still be slower compared to lower-level languages like C++ or Java. Intensive computations or complex applications may require careful optimization to maintain responsiveness.

Single-threaded Execution: JavaScript is single-threaded, meaning it can only execute one operation at a time on a single thread of execution. This can lead to blocking behavior if intensive operations are not managed properly, affecting responsiveness in some scenarios.

Lack of Static Typing: JavaScript is dynamically typed, which means variable types are determined at runtime rather than compile-time. This can lead to errors that might only surface during runtime, making it harder to catch bugs early in development.

Callback Hell: Asynchronous programming in JavaScript often relies on callbacks or nested callbacks (known as callback hell), which can make code difficult to read, maintain, and debug. Promises and async/await were introduced to alleviate this issue, but it still requires careful handling.

Scalability: Large-scale JavaScript applications can become difficult to manage and scale due to the language's flexibility and lack of strict structure. Developers may need to rely on design patterns, modularization, and frameworks to maintain code organization and scalability.

Tooling and Dependency Management: JavaScript's ecosystem, while extensive, can also be fragmented with a wide array of libraries, frameworks, and tools. Managing dependencies, ensuring compatibility, and staying up-to-date with the latest versions can be challenging.

Learning Curve: While JavaScript's syntax is relatively simple, mastering advanced concepts and best practices can take time and effort. Beginners may encounter challenges understanding concepts like closures, prototypal inheritance, and functional programming paradigms.

Evolution and Compatibility: JavaScript evolves rapidly with new features and updates to the ECMAScript standard. While this brings improvements, it can also introduce compatibility issues with older codebases or browsers that lag behind in implementing new features.

JavaScript Code Execution Phase :-

JavaScript is a *synchronous* (Moves to the next line only when the execution of the current line is completed) and *single-threaded* (Executes one command at a time in a specific order one after another serially) language. To know behind the scenes of how JavaScript code gets executed internally, we have to know something called **Execution Context** and its role in the execution of JavaScript code.

Execution Context: Everything in JavaScript is wrapped inside Execution Context, which is an abstract concept (can be treated as a container) that holds the whole information about the environment within which the current JavaScript code is being executed.

Now, an Execution Context has two components and JavaScript code gets executed in two phases.

- **Memory Allocation Phase:** In this phase, all the functions and variables of the JavaScript code get stored as a key-value pair inside

the memory component of the execution context. In the case of a function, JavaScript copied the whole function into the memory block but in the case of variables, it assigns *undefined* as a placeholder.

- **Code Execution Phase:** In this phase, the JavaScript code is executed one line at a time inside the Code Component (also known as the Thread of execution) of Execution Context.

Let's see the whole process through an example.

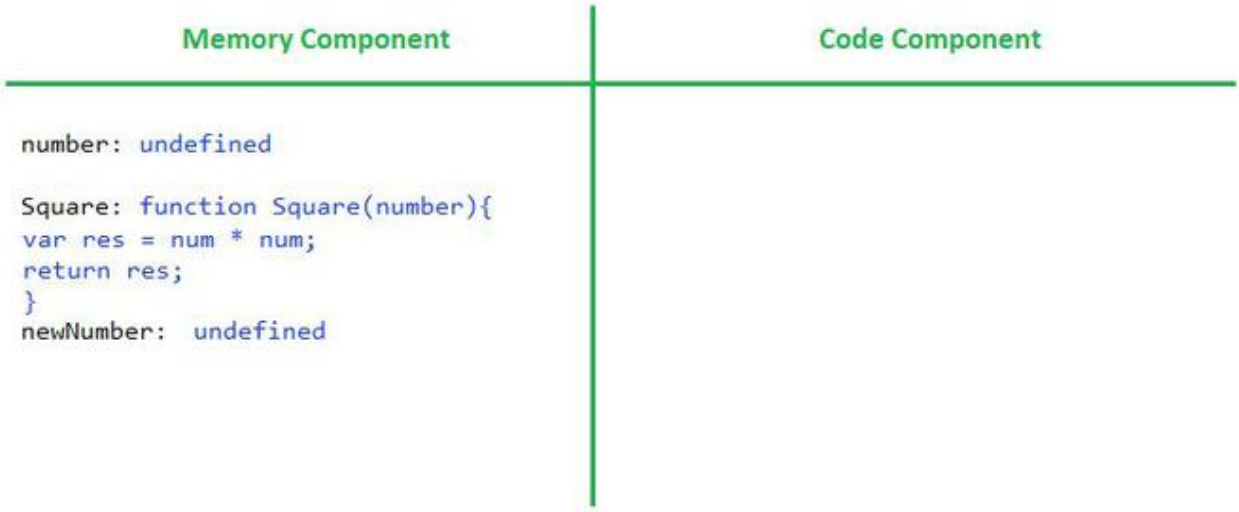
Ex:-

```
var number = 2;

function Square (n) {

    var res = n * n;

    return res;

}

var newNumber = Square(3);
```

Explanation :-

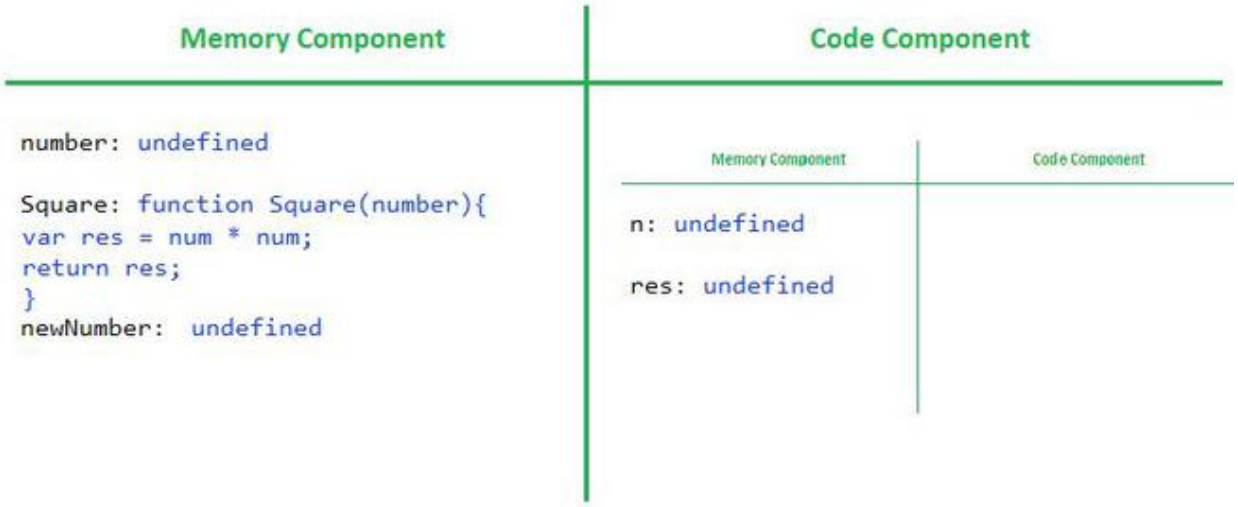
In the above JavaScript code, there are two variables named *number* and *newNumber* and one function named *Square* which is returning the square of the number. So when we run this program, Global Execution Context is created.

So, in the Memory Allocation phase, the memory will be allocated for these variables and functions like this.

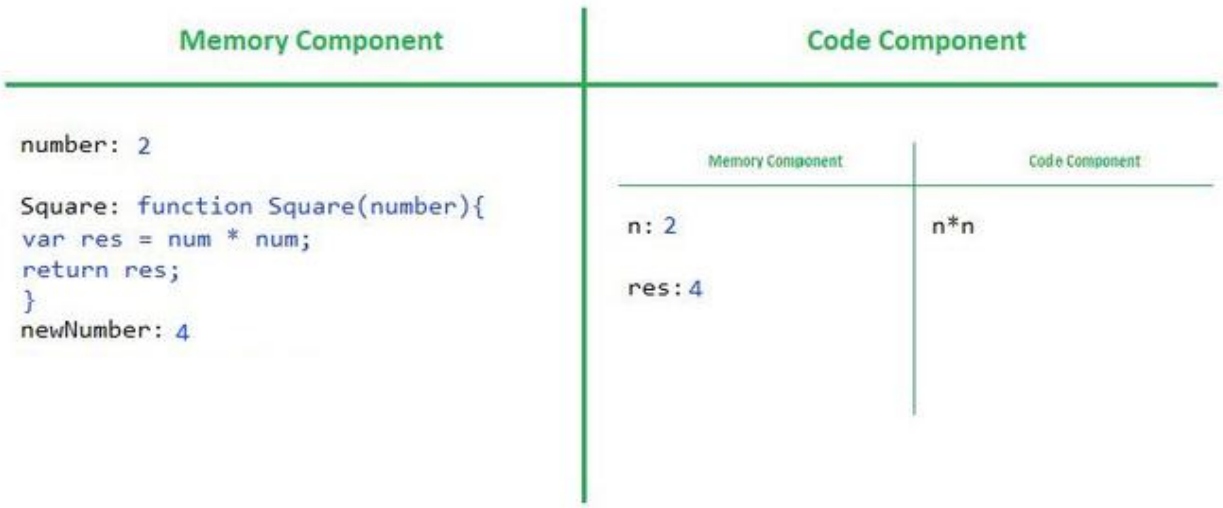


In the Code Execution Phase, JavaScript being a single thread language again runs through the code line by line and updates the values of function and variables which are stored in the Memory Allocation Phase in the Memory Component.

So in the code execution phase, whenever a new function is called, a new Execution Context is created. So, every time a function is invoked in the Code Component, a new Execution Context is created inside the previous global execution context.

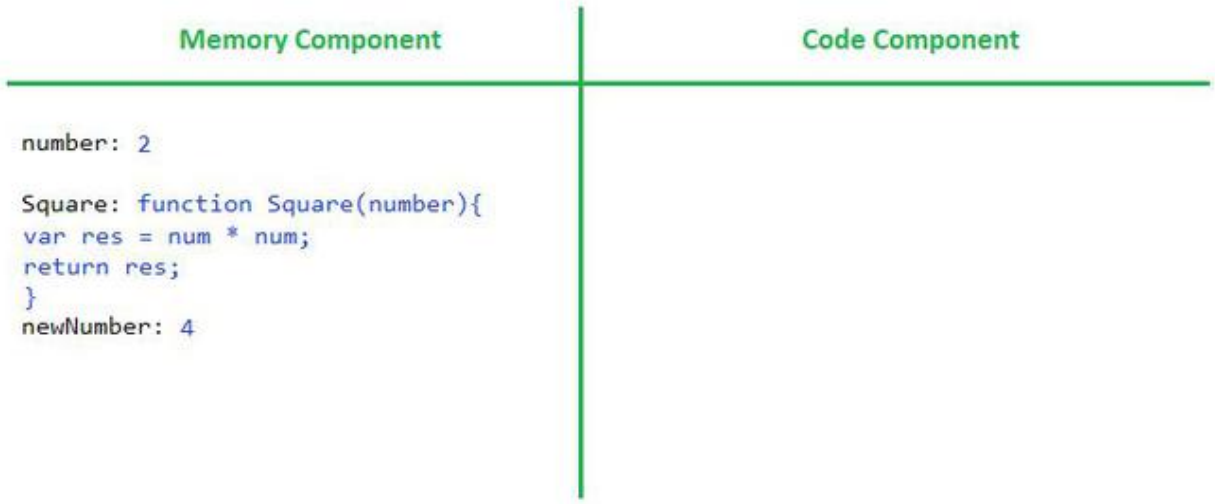


So again, before the memory allocation is completed in the Memory Component of the new Execution Context. Then, in the Code Execution Phase of the newly created Execution Context, the global Execution Context will look like the following.



As we can see, the values are assigned in the memory component after executing the code line by line, i.e. *number: 2, res: 4, newNumber: 4*.

After the *return* statement of the invoked function, the returned value is assigned in place of undefined in the memory allocation of the previous execution context. After returning the value, the new execution context (temporary) gets completely deleted. Whenever the execution encounters the return statement, It gives the control back to the execution context where the function was invoked.



After executing the first function call when we call the function again, JavaScript creates another temporary context where the same procedure repeats accordingly (memory execution and code execution). In the end, the global execution context gets deleted just like child execution contexts. The whole execution context for the instance of that function will be deleted

Call Stack: When a program starts execution JavaScript pushes the whole program as global context into a stack which is known as **Call Stack** and continues execution. Whenever JavaScript executes a new context and just follows the same process and pushes to the stack. When the context finishes, JavaScript just pops the top of the stack accordingly.

When JavaScript completes the execution of the entire code, the Global Execution Context gets deleted and popped out from the Call Stack making the Call stack empty.