1a)

Heading Tags:

HTML headings are titles or subtitles that you want to display on a webpage.

HTML headings are defined with the <h1> to <h6> tags.

<h1> defines the most important heading. <h6> defines the least important heading.

<h1>Heading 1</h1>  
<h2>Heading 2</h2>  
<h3>Heading 3</h3>  
<h4>Heading 4</h4>  
<h5>Heading 5</h5>  
<h6>Heading 6</h6>

Image tag:

The HTML <img> tag is used to embed an image in a web page.

Images are not technically inserted into a web page; images are linked to web pages. The <img> tag creates a holding space for the referenced image.

The <img> tag is empty, it contains attributes only, and does not have a closing tag.

The <img> tag has two required attributes:

* src - Specifies the path to the image
* alt - Specifies an alternate text for the image

Syntax:<img src="*url*" alt="alternatetext">

3.Anchor Tag:

The HTML <a> tag defines a hyperlink. It has the following syntax:

<a href="*url*">*link text*</a>

The most important attribute of the <a> element is the href attribute, which indicates the link's destination.

The *link text* is the part that will be visible to the reader.

Clicking on the link text, will send the reader to the specified URL address.

1b)

To add a background image on an HTML element, use the HTML style attribute and the CSS background-image property:

<body style="background-image: url('img\_girl.jpg');">

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

<video width="320" height="240" controls>  
  <source src="movie.mp4" type="video/mp4">  
  <source src="movie.ogg" type="video/ogg">  
Your browser does not support the video tag.  
</video>

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

<audio controls>  
  <source src="horse.ogg" type="audio/ogg">  
  <source src="horse.mp3" type="audio/mpeg">  
Your browser does not support the audio element.  
</audio>

2.)

HTML lists allow web developers to group a set of related items in lists.

An unordered HTML list:

* Item
* Item
* Item
* Item

An ordered HTML list:

1. First item
2. Second item
3. Third item
4. Fourth item

An unordered list starts with the <ul> tag. Each list item starts with the <li> tag.

The list items will be marked with bullets (small black circles) by default:

<ul>  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ul>

An ordered list starts with the <ol> tag. Each list item starts with the <li> tag.

The list items will be marked with numbers by default:

<ol>  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ol>

3.)

<p>H<sub>2</sub>O.</p>

<p>x<sup>2</sup>+y<sup>2</sup>.</p>

3.)

What is CSS?

* CSS stands for Cascading Style Sheets
* CSS describes how HTML elements are to be displayed on screen, paper, or in other media
* CSS saves a lot of work. It can control the layout of multiple web pages all at once
* External stylesheets are stored in CSS files

There are three ways of inserting a style sheet:

* External CSS
* Internal CSS
* Inline CSS

## External CSS

With an external style sheet, you can change the look of an entire website by changing just one file!

Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.

<!DOCTYPE html>  
<html>  
<head>  
<link rel="stylesheet" href="mystyle.css">  
</head>  
<body>  
  
<h1>This is a heading</h1>  
<p>This is a paragraph.</p>  
  
</body>  
</html>

## Internal CSS

An internal style sheet may be used if one single HTML page has a unique style.

The internal style is defined inside the <style> element, inside the head section.

Internal styles are defined within the <style> element, inside the <head> section of an HTML page:

<!DOCTYPE html>  
<html>  
<head>  
<style>  
body {  
  background-color: linen;  
}  
  
h1 {  
  color: maroon;  
  margin-left: 40px;  
}  
</style>  
</head>  
<body>  
  
<h1>This is a heading</h1>  
<p>This is a paragraph.</p>  
  
</body>  
</html>

## Inline CSS

An inline style may be used to apply a unique style for a single element.

To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

### **Example**

Inline styles are defined within the "style" attribute of the relevant element:

<!DOCTYPE html>  
<html>  
<body>  
  
<h1 style="color:blue;text-align:center;">This is a heading</h1>  
<p style="color:red;">This is a paragraph.</p>  
  
</body>  
</html>

3b)

The HTML id attribute is used to specify a unique id for an HTML element.

You cannot have more than one element with the same id in an HTML document.

The id attribute specifies a unique id for an HTML element. The value of the id attribute must be unique within the HTML document.

The id attribute is used to point to a specific style declaration in a style sheet. It is also used by JavaScript to access and manipulate the element with the specific id.

The syntax for id is: write a hash character (#), followed by an id name. Then, define the CSS properties within curly braces {}.

In the following example we have an <h1> element that points to the id name "myHeader". This <h1> element will be styled according to the #myHeader style definition in the head section:

<!DOCTYPE html>  
<html>  
<head>  
<style>  
#myHeader {  
  background-color: lightblue;  
  color: black;  
  padding: 40px;  
  text-align: center;  
}  
</style>  
</head>  
<body>  
  
<h1 id="myHeader">My Header</h1>  
  
</body>  
</html>

A class name can be used by multiple HTML elements, while an id name must only be used by one HTML element within the page:

The *.class* selector selects elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the name of the class.

You can also specify that only specific HTML elements should be affected by a class. To do this, start with the element name, then write the period (.) character, followed by the name of the class (look at Example 1 below).

<style>  
/\* Style the element with the id "myHeader" \*/  
**#myHeader** {  
  background-color: lightblue;  
  color: black;  
  padding: 40px;  
  text-align: center;  
}  
  
/\* Style all elements with the class name "city" \*/  
**.city**{  
  background-color: tomato;  
  color: white;  
  padding: 10px;  
}  
</style>  
  
<!-- An element with a unique id -->  
<h1 id="myHeader">My Cities</h1>  
  
<!-- Multiple elements with same class -->  
<h2 class="city">London</h2>  
<p>London is the capital of England.</p>  
  
<h2 class="city">Paris</h2>  
<p>Paris is the capital of France.</p>  
  
<h2 class="city">Tokyo</h2>  
<p>Tokyo is the capital of Japan.</p>

4)

<div class="full-width">

<img class="imgone" src="http://s23.postimg.org/b2h2rz457/image.png" />

<img class="imgtwo" src="http://s13.postimg.org/jlyzyofkn/image.png" />

</div>

.full-width {

position: relative;

width: 300px;

height: 200px;

}

img.imgone {

position:absolute;

Z index:1;

width: 300px;

height: 200px;

}

img.imgtwo {

position:absolute;

Z index:2;

width: 80px;

height: 80px;

top: 10px;

right: 10px;

}

4.b)

## The CSS 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: margins, borders, padding, and the actual content. 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.

<!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>

5.)

## JavaScript Function Syntax

A JavaScript function is defined with the function keyword, followed by a **name**, followed by parentheses **()**.

Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).

The parentheses may include parameter names separated by commas:  
**(parameter1, parameter2, ...)**

The code to be executed, by the function, is placed inside curly brackets: **{}**

function name(parameter1, parameter2, parameter3) {  
  // code to be executed  
}

Function **parameters** are listed inside the parentheses () in the function definition.

Function **arguments** are the **values** received by the function when it is invoked.

Inside the function, the arguments (the parameters) behave as local variables.

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Functions</h2>

<p>This example calls a function which performs a calculation and returns the result:</p>

<p id="demo"></p>

<script>

var x = myFunction(4, 3);

document.getElementById("demo").innerHTML = x;

function myFunction(a, b) {

return a \* b;

}

</script>

</body>

</html>

1. **<html>**
2. **<head>**
3. **<title>** Random Image Generator **</title>**
4. **</head>**
5. **<script>**
6. function getRandomImage() {
7. //declare an array to store the images
8. var randomImage = new Array();
10. //insert the URL of images in array
11. randomImage[0] = "https://images.pexels.com/photos/858115/pexels-photo-858115.jpeg?auto=compress&cs=tinysrgb&dpr=1&w=500";
12. randomImage[1] = "http://www.petsworld.in/blog/wp-content/uploads/2014/09/running-cute-puppies.jpg";
13. randomImage[2] = "https://images.pexels.com/photos/142497/pexels-photo-142497.jpeg?auto=compress&cs=tinysrgb&dpr=1&w=500";
14. randomImage[3] = "https://images.unsplash.com/photo-1543877087-ebf71fde2be1?ixlib=rb-1.2.1&ixid=eyJhcHBfaWQiOjEyMDd9&auto=format&fit=crop&w=500&q=60";
15. randomImage[4] = "https://wi.wallpapertip.com/wsimgs/156-1565522\_puppies-desktop-wallpaper-desktop-background-puppies.jpg";
16. randomImage[5] = "https://images.unsplash.com/photo-1501265976582-c1e1b0bbaf63?ixlib=rb-1.2.1&ixid=eyJhcHBfaWQiOjEyMDd9&auto=format&fit=crop&w=500&q=60";
18. //generate a number and provide to the image to generate randomly
19. var number = Math.floor(Math.random()\*randomImage.length);
21. //return the images generated by a random number
22. return document.getElementById("result").innerHTML = '<img src="'+randomImage[number]+'" **/>**';
23. }
25. **</script>**
26. **<body>**
27. **<center><h2** style="color:green"**>** Random Image Generator **</h2></center>**
28. **<h4>** Click the button to generate and display random images on the webpage **</h4>**
29. <!-- call user-defined getRandomImage function after 2 seconds -->
30. **<button** onclick = "setInterval(getRandomImage, 2000)"**>** Generate Image **</button>**
31. **<br>** **<br>**
32. **<span** id="result" align="center"**>** **</span>**
34. **</body>**
35. **</html>**

6.a) **Document Object Model is an API that represents and interacts with HTML or XML documents.**

**Why DOM is required?**

HTML is used to **structure**the web pages and Javascript is used to add **behavior**to our web pages. When an HTML file is loaded into the browser, the javascript can not understand the HTML document directly. So, a corresponding document is created(DOM). **DOM is basically the representation of the same HTML document but in a different format with the use of objects**. Javascript interprets DOM easily i.e javascript can not understand the tags(<h1>H</h1>) in HTML document but can understand object h1 in DOM. Now, Javascript can access each of the objects (h1, p, etc) by using different functions.

**Structure of DOM**: DOM can be thought of as a Tree or Forest.

* [**Window Object**](https://www.geeksforgeeks.org/properties-of-window-object/#:~:text=It%20represents%20an%20array%20that,frames%20of%20a%20given%20window.&text=It%20returns%20a%20reference%20to%20a%20DOMPoint%20object%2C%20which%20represents,point%20in%20a%20coordinate%20system.&text=It%20provides%20information%20of%20the%20URLs%20visited%20in%20the%20current%20window.&text=It%20represents%20the%20number%20of%20frames%20in%20the%20current%20window.)**:** Window Object is object of the browser which is always at top of the hierarchy.  It is like an API that is used to set and access all the properties and methods of the browser. It is automatically created by the browser.
* **Document object:** When an HTML document is loaded into a window, it becomes a document object. The ‘document’ object has various properties that refer to other objects which allow access to and modification of the content of the web page. If there is a need to access any element in an HTML page, we always start with accessing the ‘document’ object. Document object is property of window object.
* **Form Object:** It is represented by ***form*** tags.
* [**Link Object**](https://www.geeksforgeeks.org/html-dom-link-object/)**:** It is represented by ***link***tags.
* [**Anchor Object**](https://www.geeksforgeeks.org/html-dom-anchor-object/)**:** It is represented by ***a href*** tags.
* **Form Control Elements:**: Form can have many control elements such as text fields, buttons, radio buttons, checkboxes, etc.

**Methods of Document Object**:

* [**write**](https://www.geeksforgeeks.org/html-dom-write-method/)**(“string”):** Writes the given string on the document.
* [**getElementById()**](https://www.geeksforgeeks.org/html-dom-getelementbyid-method/)**:** returns the element having the given id value.
* [**getElementsByName()**](https://www.geeksforgeeks.org/html-dom-getelementsbyname-method/)**:** returns all the elements having the given name value.
* [**getElementsByTagName():**](https://www.geeksforgeeks.org/html-dom-getelementsbytagname-method/)returns all the elements having the given tag name.
* [**getElementsByClassName()**](https://www.geeksforgeeks.org/html-dom-getelementsbyclassname-method/)**:** returns all the elements having the given class name.
* ex:
* <!DOCTYPE html>
* <**html**>
* <**body**>
* <**h2**>GeeksforGeeks</**h2**>
* <!-- Finding the HTML Elements by their Id in DOM -->
* <**p** id="intro">A Computer Science portal for geeks.</**p**>





* <**p**>This example illustrates the <**b**>getElementById</**b**> method.</**p**>


* <**p** id="demo"></**p**>


* <**script**>
* const element = document.getElementById("intro");
* document.getElementById("demo").innerHTML =
* "GeeksforGeeks introduction is: " + element.innerHTML;
* </**script**>
* </**body**>
* </**html**>

6.)

On mouse down on mouse up:

<!DOCTYPE html>

<html>

<body>

<p id="myP" onmousedown="mouseDown()" onmouseup="mouseUp()">

Click the text! The mouseDown() function is triggered when the mouse button is pressed down over this paragraph, and sets the color of the text to red. The mouseUp() function is triggered when the mouse button is released, and sets the color of the text to green.

</p>

<script>

function mouseDown() {

document.getElementById("myP").style.color = "red";

}

function mouseUp() {

document.getElementById("myP").style.color = "green";

}

</script>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<img onmouseover="bigImg(this)" onmouseout="normalImg(this)" border="0" src="smiley.gif" alt="Smiley" width="32" height="32">

<p>The function bigImg() is triggered when the user moves the mouse pointer over the image.</p>

<p>The function normalImg() is triggered when the mouse pointer is moved out of the image.</p>

<script>

function bigImg(x) {

x.style.height = "64px";

x.style.width = "64px";

}

function normalImg(x) {

x.style.height = "32px";

x.style.width = "32px";

}

</script>

</body>

</html>