1.Define HTML.

HTML (Hyper Text Markup Language) is a code that defines the structure and content of a web page.

2. Define the <div> tag in HTML.

The <div> tag defines a division or a section in an HTML document. The <div> tag is used as a container for HTML elements - which is then styled with CSS or manipulated with JavaScript. The <div> tag is easily styled by using the class or id attribute.

3. Define the <span> tag in HTML.

The <span> tag is an inline container used to mark up a part of a text, or a part of a document. The <span> tag is easily styled by CSS or manipulated with JavaScript using the class or id attribute.

4. Explain the basic syntax of an HTML document.

**Basic Syntax**

* The fundamental syntactic units of HTML are called tags.
* The syntax of a tag is the tag's name surrounded by angle brackets (< and >).
* Tag names must be written in all lowercase letters

5. Define the <img> tag and its usage in HTML.

ADDING GRAPHICS TO WEB PAGES <IMG> Tag : This tag is used to insert align and size an image. An image can be put in an HTML document using

<IMG> tag's SRC attribute as follows:

<IMG SRC="filename"/>

e. g.: <IMG SRC = "d:\abc.jpg"/>

<IMG SRC="abc.jpg"/>

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

6. Define the <ul> and <ol> tags in HTML.

<ul> defines an unordered list (typically rendered as a bulleted list),<ol> defines an ordered list (typically rendered as a numbered list). Both use <li> tags to define list items.

7. Define the <form> tag in HTML.

The form tag in HTML is used to collect data from the user and send it to the server for further processing

8. Define the <table> tag and its use in HTML.

The HTML <table> tag is used to define the beginning and end of an HTML table. An HTML table is created using the <table> tag. Inside this tag, you use

* <tr> to define table rows,
* <th> for table headers, and
* <td> for table data cells

9. List the features in HTML.

1. New Tags for Better Structure:

2. Built-in Audio and Video

3. Graphics with Canvas:

4. Storage for Data:

5. Location Services:

6. Better Forms:

7. Offline Apps:

8. New Tools for Developers:

9. Responsive Design

10. Better Accessibility:

10. Define CSS.

CSS stands for Cascading style sheets. It describes to the user how to display HTML elements on the screen in a proper format. CSS is the language that is used to style HTML documents.

11. Explain the purpose of the <a> tag in HTML.

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

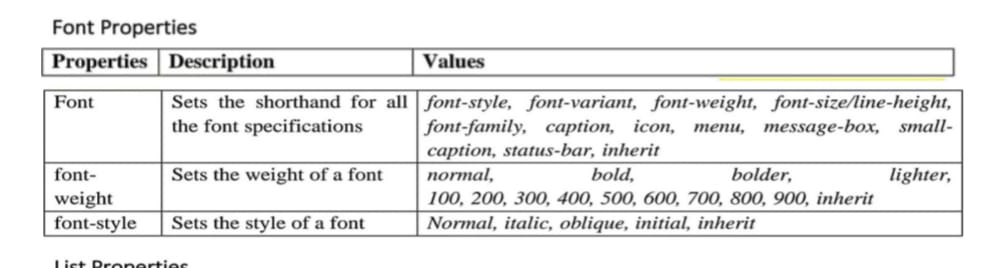
12. Define the difference between internal, external, and inline CSS.

|  |  |  |
| --- | --- | --- |
| Internal | External | Inline |
| An internal style sheet may be used if one single HTML page has a unique style. | The external style sheet is generally used when you want to make changes on multiple pages. | In inline style may be used to apply a unique style for a single element. |
| The internal style is defined inside the <style> element, inside the head section. | external styles are defined within the <link> element, inside the <head> section of an HTML page | To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property. |
| General syntax:  <style type =”text/css”>  Rule list  </style> | General Syntax:  <link rel=”stylesheet” type=”text/css”  Href=”http://www.wherever. org/termpaper.css”>  </link> | General syntax:  Style=”property\_1: value\_1;  property\_2: value\_2;  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  property\_n: value\_n;” |

13. Define the color property in CSS.

HTML colors are specified with predefined color names, or with RGB, HEX, HSL, RGBA, or HSLA values.

14. Define the font-family property in CSS.



15. Define the text-align property in css.

Text alignment properties in CSS: Following are the text alignment properties of CSS. These are the properties using which text content can be positioned as per your wish.

* Text-align:right;
* Text-align:left;
* Text-align:center;

16. Define the background-color property in CSS.

The background-color CSS property sets the background color of an element, allowing you to create a solid color backdrop for any HTML element.

17. Explain the purpose of CSS selectors.

Selectors are used in internal or external css. After selector name, a declaration block, i.e curly brackets {} are used. Within {}, css code for targeted element is written with key : value pair.

18. Define the margin property in CSS.

The CSS margin properties are used to create space around elements, outside of any defined borders.

19. Define the padding property in CSS.

CSS padding is a fundamental concept in web design that defines the space between an element's content and its border.

20. List the different types of CSS selectors.

Type of CSS selectors

1) CSS Element Selector

2) CSS Id Selector

3) CSS Class Selector

4) CSS Universal Selector

5) CSS Group Selector

21. Define JavaScript.

JavaScript is a programming language used to create dynamic content for websites. It is a lightweight, cross-platform, and single-threaded programming language. JavaScript is an interpreted language that executes code line by line providing more flexibility

22. Define the <frameset> tag and its use in HTML.

The <frameset> tag in HTML, now deprecated in HTML5, was used to divide a web page into multiple sections (frames), each displaying a separate HTML document. It allowed for creating multi-pane layouts, with the rows and cols attributes specifying the layout structure.

23. Define what an object is in JavaScript.

An object in JavaScript is a data structure used to store related data collections. It stores data as key-value pairs, where each key is a unique identifier for the associated value. Objects are dynamic, which means the properties can be added, modified, or deleted at runtime. Name, age, greet.

24. Define a primitive data type in JavaScript.

In JavaScript, a primitive (primitive value, primitive data type) is data that is not an object and has no methods or properties.

25. Define what a loop is in JavaScript.

A loop is a programming tool that is used to repeat a set of instructions. Iterate is a generic term that means “to repeat” in the context of loops. A loop will continue to iterate until a specified condition, commonly known as a stopping condition, is met.

26. Define a function in JavaScript.

a function is a "subprogram" that can be called by code external (or internal, in the case of recursion) to the function

27. Define the alert() function in JavaScript.

JavaScript Alert or alert() is an in-built function in JavaScript that can be used to display a dialog box on a user's screen to alert the user about a specific event.

28. Define the console.log() method in JavaScript.

For debugging purposes, you can call the console.log() method in the browser to display data. Or a function that writes a message to log on the debugging console.

29. Define the document.getElementById() method in JavaScript.

The getElementById() method of the Document interface returns an Element object representing the element whose id property matches the specified string.

30. Explain the purpose of the prompt() function in JavaScript

The prompt() function is one of the simplest ways to get user input in JavaScript. This built-in function opens a dialog box where the user can type something, and the value is returned as a string.

1. **marks**

1. Explain different form elements.

Form Elements

|  |  |
| --- | --- |
| Elements | Descriptions |
| <button> | It defines a clickable button to control other elements or execute a functionality. |
| <select> | It is used to create a drop-down list. |
| <textarea> | It is used to get input long text content. |
| <fieldset> | It is used to draw a box around other form elements and group the related data. |

**2. Define and explain the concept of basic text markup in HTML**

In HTML, basic text markup involves using tags to structure and format text content, defining elements like headings, paragraphs, lists, and emphasizing specific words or phrases.

**HTML Tags:**

HTML uses tags (like <p>, <h1>, <ul>, <li>, <strong>, <em>) to identify and structure different types of content on a webpage.

**Elements:**

An HTML element consists of a start tag, content, and an end tag, which together define the element's structure and functionality. For example, <p>This is a paragraph.</p>.

**Basic Text Elements:**

Paragraphs: <p> defines a paragraph of text.

Headings: <h1> through <h6> define headings of different levels.

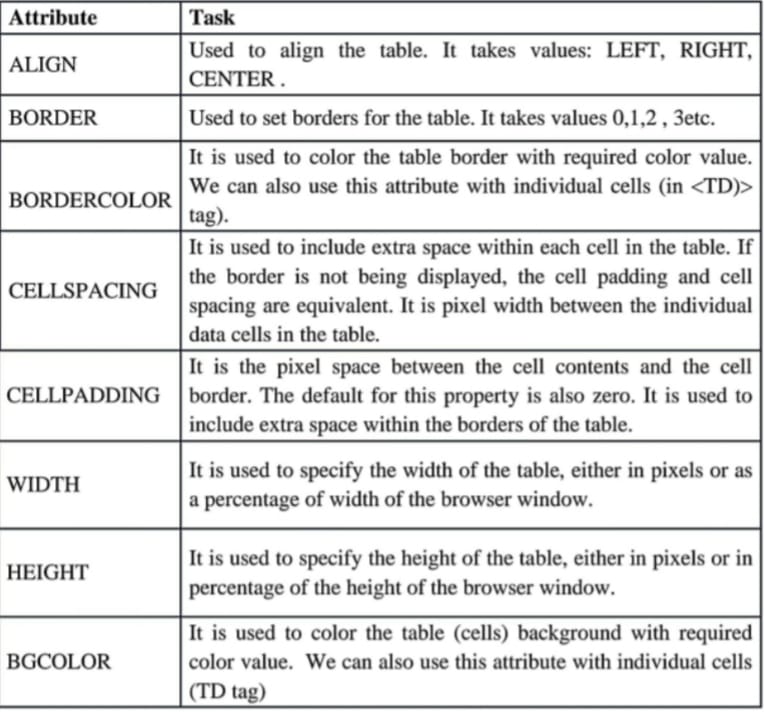
Lists: <ul> (unordered list) and <ol> (ordered list) define lists, with <li> defining list items.

Emphasis: <strong> makes text bold, and <em> makes text italic.

Links: <a> creates hyperlinks.

Line Break: <br> inserts a single line break.

**3.Explain table tag in HTML.**



4. Explain how the <div> and <span> tags are used in HTML.

**Div Tag** :- The <div> tag defines a division or a section in an HTML document. The <div> tag is used as a container for HTML elements - which is then styled with CSS or manipulated with JavaScript. The <div> tag is easily styled by using the class or id attribute. Example - <html>

<head>

<div style="border:6px solid pink;">

<p>Welcome to Javatpoint.com, Here you get tutorials on latest technologies.</p>

<p>This is second paragraph</p>

</div>

</body>

</html>

**Span Tag –**The <span> tag is an inline container used to mark up a part of a text, or a part of a document. The <span> tag is easily styled by CSS or manipulated with JavaScript using the class or id attribute. Example - Output <!DOCTYPE html>

<html>

<body>

<h1>The span element</h1>

<p>My mother has <span style="color:blue; ">blue</span> eyes and my father has <span style="color:red; "> dark green</span> eyes.</p>

</body>

</html>

5. Explain the three types of CSS (inline, internal, and external)

**1.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. **General form:**

<style type =”text/css”>

Rule list

</style>

**Example :**

<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>

**2.External CSS:-**

 The external style sheet is generally used when you want to make changes on multiple pages.  External styles are defined within the <link> element, inside the <head> section of an HTML page

 The external style sheet can be written in any text editor, and must be saved with a css extension. General form:

 The rel attribute is used to specify the relationship of the linked to document to the document  The href attribute is used to specify the url of the style sheet document

**General Syntax:**

<link rel=”stylesheet” type=”text/css”

Href=”http://www.wherever. org/termpaper.css”>

</link>

**Example**:

mystyle.css

body (

background-color: lightblue;

}

h1 {

color: navy;

margin-left: 20px;

<html>

<head>

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

</head>

<body>

<h1>This is a heading</h1>

<p>This is a paragraph.</p>

**3.Inline CSS:-**

 In 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**.**

**General Format :**

Style=”property\_1: value\_1;

property\_2: value\_2;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

property\_n: value\_n;”

**Example :**

<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>

6. List the different types of CSS selectors and explain how they work.

1**) ID Selectors:-**

 ID Selector is used to call an HTML Element by its unique id name.  Id is always unique in a single web page.  We can not give same ID name to any other HTML Element in same webpage

 In css, id selector is called using HASH ( #), followed by id name. Example :-

<html>

<head>

<style>

#para1 {

text-align: center;

color: red;

}

</style>

</head> <body>

<p id="para1">Hello World!</p>

<p>This paragraph is not affected by the style.</p>

</body>

</html>

**2) Class Selectors:-**

 Class Selector:-

in css is used to call all html elements with same class name.  Class represents a group of different or same html elements.  We can give same class name to two or more different HTML Elements.  In css, class is called using DOT ( .), followed by class name. Example :

<html>

<head>

<style>

.center {

text-align: center;

color: red;

}

</style>

</head>

<body>

<h1 class="center">Red and center-aligned heading</h1>

<p class="center">Red and center-aligned paragraph.</p>

</body>

</html>

**3) Element Selector :-**

 Element type Selector is the first Major selector in css.  Any html element can be accessed in css using their tag name.  Tags can be used more than once in html, so all elements will be called. Example :

<html>

<head>

<style>

p {

text-align: center;

color: red;

}

</style>

</head>

<body>

<p>Every paragraph will be affected by the style.</p>

<p id="para1">Me too!</p>

<p>And me!</p>

</body>

</html>

**4) Universal Selectors:-**

 The universal selector is used as a wildcard character.  It selects all the elements on the pages.  Universal Selector Asterisk \* is used to call all html elements in css.

Example :

<html>

<head>

<style>

\* {

text-align: center;

color: blue;

}

</style>

</head>

<body>

<h1>Hello world!</h1>

<p>Every element on the page will be affected by the style.</p>

<p id="para1">Me too!</p>

<p>And me!</p>

</body>

</html>

**5) Group Selectors:-**

 The grouping selector is used to select all the elements with the same style definitions.  Grouping selector is used to minimize the code.  Commas are used to separate each selector in grouping. Example :

<html>

<head>

<style>

h1, h2, p {

text-align: center;

color: red;

}

</style>

</head>

<body>

<h1>Hello World!</h1>

<h2>Smaller heading!</h2>

<p>This is a paragraph.</p>

</body>

</html>

7. List and explain the general syntactic characteristics of JavaScript, such as statements, expressions, and syntax rules.

**General Syntactic Characteristics**

**1. Light-Weight Scripting Language** :-JavaScript is designed for client-side execution, making it lightweight for web applications.

**2. Dynamic Typing:-** JavaScript variables' types are determined by the value assigned, without needing explicit type declarations.

**3. Object-Oriented Programming Support :-** JavaScript supports object-oriented programming with concepts like classes, encapsulation, and inheritance (especially from ES6 onwards).

**4. Functional Style :-** JavaScript allows functions to be treated as objects, enabling functional programming features like callbacks and closures.

**5. Platform Independent :-** JavaScript can run on any platform or browser without affecting functionality.

**6. Prototype-based Language :-** JavaScript uses prototypes for object inheritance instead of classes, allowing objects to inherit properties and methods.

**7. Interpreted Language** :-JavaScript is processed line by line by the browser’s interpreter, although modern engines use Just-In-Time (JIT) compilation for optimization.

***8. Single-threaded :-*** JavaScript executes one task at a time by default, but asynchronous features allow for parallel execution.

**9. Async Processing** :- JavaScript supports asynchronous operations using promises and async/await, allowing non- blocking tasks.

**10. Web Workers :-** Web Workers allow background processing in separate threads, enabling parallel execution for resource-heavy tasks

***Primitives:-***

* Primitives are the basic data types provided by the language.
* Integer: Whole numbers, e.g., 5, -23.
* Float: Numbers with decimal points, e.g., 3.14, -0.001.
* Boolean: Represents True or False values.
* String: Sequence of characters, e.g., "Hello, World!".

Example –

num = 10; // Integer

price = 99.99; // Float

***Operations:-***

Operations are the actions that can be performed on primitives, such as arithmetic, logical, andcomparison operations.

**1. Arithmetic Operations:**

a. Addition: +

b. Subtraction: - c. Multiplication: \* d. Division: /

e. Modulus (remainder): %

**Example –**

let a = 5, b = 2;

let addition = a + b; // 7

let subtraction = a - b; // 3

**2. Comparison Operations:**

a. Equal: == b. Not Equal: !=

c. Greater than: > d. Less than: <

**Example –**

let x = 10, y = 20;

console.log(x == y); // false (equal to)

console.log(x != y); // true (not equal to)

**3. Logical Operations:**

a. AND: and

b. OR: or

c. NOT: not

**Example –**

letisTrue = true, isFalse = false;

console.log(isTrue&&isFalse); // false (AND)

console.log(isTrue || isFalse); // true (OR)

***Expressions:-***

Expressions are combinations of variables, constants, and operators that produce a value. Example -

let x = 5, y = 10; let result = (x + y) \* 2; // 30 (expression: (5 + 10) \* 2)

8. List the primitive data types in JavaScript and explain each one.

**Primitives:-**

* Primitives are the basic data types provided by the language.
* Integer: Whole numbers, e.g., 5, -23.
* Float: Numbers with decimal points, e.g., 3.14, -0.001.
* Boolean: Represents True or False values.
* String: Sequence of characters, e.g., "Hello, World!".

Example –

num = 10; // Integer

price = 99.99; // Float

9. List the different ways to handle keyboard output and input in JavaScript and explain their usage.

In JavaScript, you can handle keyboard input and output using event listeners for keyboard events (keydown, keypress, keyup) and by manipulating HTML elements or using browser APIs like prompt() and alert().

**Keyboard Input:**

**keydown**: This event fires when a key is pressed down, regardless of whether it's a printable character or a modifier key (e.g., Shift, Ctrl).

**Usage**: Useful for actions that should happen repeatedly as long as a key is held down, like controlling a game character's movement.

Example:

document.addEventListener('keydown', function(event) {

console.log('Key pressed:', event.key);

if (event.key === 'ArrowLeft') {

// Move left

}

});

**keypress:** This event fires when a key is pressed and a character is generated (i.e., for printable characters).

**Usage:** Useful for capturing character input, like in a text field.

Example:

document.addEventListener('keypress', function(event) {

console.log('Character pressed:', event.key);

});

**keyup:** This event fires when a key is released.

**Usage:** Useful for actions that should happen once a key is released, like stopping a game character's movement.

Example:

document.addEventListener('keyup', function(event) {

console.log('Key released:', event.key);

});

**Keyboard Output:**

Manipulating HTML elements: You can update the content of HTML elements to display information based on keyboard input.

**Usage:** Displaying user input, showing game scores, or updating a log.

Example:

let inputField = document.getElementById('myInput');

inputField.addEventListener('keydown', function(event) {

document.getElementById('output').textContent = 'You typed: ' + event.key;

});

**Using window.alert()**

You can use an alert box to display data:

Example –

<html>

<body>

<script> window.alert(5 + 6); </script>

</body>

</html>

**Using console.log()**

For debugging purposes, you can call the console.log() method in the browser to display data. Example

<html>

<body>

<p>F12 on your keyboard will activate debugging.</p>

<p>Then select "Console" in the debugger menu.</p>

<script>

console.log(5 + 6); </script>

</body>

</html>

**prompt():**

This displays a dialog box with a message, an input field, and OK and Cancel buttons, allowing the user to enter text.

Usage: Getting user input in a simple way.

Example:

let name = prompt("Enter your name:");

console.log("You entered:", name)

10. List and explain the types of expressions in JavaScript and provide examples.

Expressions are combinations of variables, constants, and operators that produce a value.

**Types of Expression**

1.**Arithmetic Expressions:-**

These expressions involve arithmetic operations (+, -, , /, %, \*) and return a numeric value.

Example:

let x = 5 + 3; // 8

let y = 10 \* 2 - 3; // 17

let z = 2 \*\* 3; // 8 (exponentiation)

**2. String Expressions:-**

These involve string values and operators like + for concatenation.

Example:

let str = "Hello" + " " + "World"; // "Hello World"

**3. Logical (Boolean) Expressions:-**

These involve logical operators (&&, ||, !) and evaluate to a boolean value (true or false).

Example:

let a = 5 > 3 && 2 < 4; // true

let b = 5 > 10 || 3 < 1; // false

let c = !false; // true

**4. Comparison Expressions:-**

These involve comparison operators (==, ===, !=, !==, >, <, >=, <=) and return a boolean value.

Example:

let result = 10 > 5; // true

let isEqual = "5" == 5; // true (type conversion)

let isStrictEqual = "5" === 5; // false (no type conversion)

**5. Assignment Expressions:-**

These use assignment operators (=, +=, -=, \*=, /=, %=) to assign values to variables.

Example:

let x = 10;

x += 5; // x = x + 5; x becomes 15

x \*= 2; // x = x \* 2; x becomes 30

**6. Conditional (Ternary) Expressions:-**

These use the ternary operator (condition ? expr1 : expr2) to evaluate a condition and return one of two values.

Example:

let age = 20;

let status = age >= 18 ? "Adult" : "Minor"; // "Adult"

**7. Function Expressions:-**

Functions can be defined as expressions and assigned to variables.

Example:

const greet = function(name) {

return "Hello " + name;

};

console.log(greet("John")); // "Hello John"

**8. Object and Array Expressions:-**

These create objects and arrays directly.

Example:

let person = { name: "Alice", age: 30 };

let colors = ["red", "green", "blue"];

**9. Template Literal Expressions:-**

These use backticks (`) to embed expressions into strings using ${}.

Example:

let name = "John";

let message = Hello, ${name}!; // "Hello, John!"

**10. Regular Expression (RegExp) Expressions:-**

These are used to define patterns for string matching.

Example:

let pattern = /hello/;

let result = pattern.test("hello world"); // true