1.Differentiate HTML and XHTML.

HTML (Hypertext Markup Language) and XHTML (Extensible Hypertext Markup Language) are both markup languages used for creating web pages.

HTML has a more lenient syntax and allows certain deviations, such as unclosed tags or attribute values without quotes. On the other hand, XHTML follows a stricter syntax, which must conform to XML rules. All tags and attributes must be properly closed, and attribute values must be enclosed in quotes.

XHTML is an extension of HTML and was introduced to promote cleaner, more standardized markup. XHTML follows the rules of XML, making it more suitable for integration with other XML-based technologies. However, with the emergence of HTML5, the use of XHTML has diminished, and HTML5 has become the preferred choice for web development.

2.Explain basic HTML markup (tag) syntax with diagram.

The basic syntax of HTML markup involves using tags to define the structure and content of a web page. Here's an explanation of the syntax along with a diagram:

Here's a diagram of the basic HTML markup syntax:

```
<tagname attribute="value">
 Content
</tagname>
```

1. **Opening Tag:** An HTML tag begins with an opening angle bracket < followed by the tag name. For example, <tagname>. This indicates the start of an element.
2. **Attributes**: Attributes provide additional information about the element and are specified within the opening tag. They are written as attribute="value". Attributes are optional and can vary depending on the tag. For example, <tagname attribute="value">.
3. **Content**: The content of an HTML element is placed between the opening and closing tags. It can include text, other HTML elements, or both. For example, <tagname>Content</tagname>.
4. **Closing Tag:** An HTML tag also has a closing tag, which marks the end of the element. It is similar to the opening tag but includes a forward slash / before the tag name. For example, </tagname>.

3.What is the use of <audio> tag? Write its important attributes with their Purpose.

The **<audio>** tag in HTML is used to embed audio content into a web page. It allows you to play audio files directly within the browser without requiring external plugins. Here are some important attributes of the <audio> tag along with their purposes:

1. **src:** The src attribute specifies the URL or file path of the audio file to be played. It can be an absolute URL or a relative path to the audio file.
2. **controls**: The controls attribute enables the default audio controls to be displayed, such as play/pause, volume control, and progress bar. When set to controls, users can interact with the audio player directly.
3. **autoplay**: The autoplay attribute, when present, automatically starts playing the audio file as soon as the page loads. However, note that some browsers may restrict the autoplay feature for usability and performance reasons.
4. **loop**: The loop attribute, when added, allows the audio to continuously loop playback. It makes the audio restart from the beginning once it reaches the end.
5. **volume**: The volume attribute sets the default volume level of the audio, ranging from 0.0 to 1.0. For example, volume="0.5" sets the volume to 50%.
6. **muted**: The muted attribute, when added, sets the audio to be initially muted. Users can then manually unmute it using the audio controls.

4. What is the use of <video> tag? Write its important attributes with their Purpose.

The **<video>** tag in HTML is used to embed videos or media content on a webpage. It provides a standard way to include video files in different formats and allows the browser to handle the rendering and playback of the video. Here are some important attributes of the <video> tag and their purposes:

1. **src**: The src attribute specifies the URL or file path of the video file to be played. It can be an absolute URL or a relative path to the video file.
2. **controls**: The controls attribute enables the default video controls to be displayed, such as play/pause, volume control, progress bar, and fullscreen toggle. When set to controls, users can interact with the video player directly.
3. **autoplay**: The autoplay attribute, when present, automatically starts playing the video as soon as the page loads. However, note that some browsers may restrict the autoplay feature for usability and performance reasons.

4. **loop**: The loop attribute, when added, allows the video to continuously loop playback. It makes the video restart from the beginning once it reaches the end.
5. **width and height:** The width and height attributes define the dimensions of the video player in pixels. They can be used to set the initial size of the video player on the web page.
6. **muted**: The muted attribute, when added, sets the video to be initially muted. Users can then manually unmute it using the video controls.

5.How to write HTML5 comments.

In HTML5, you can write comments to add explanatory or descriptive notes within your HTML code. Here's how you can write HTML5 comments:

1. **Single-line Comment:** To write a single-line comment, use the following syntax:

```
<!-- This is a single-line comment -->
```

Anything between <!-- and --> will be treated as a comment and will not be rendered by the browser.

2**. Multi-line Comment:** If you have a comment that spans multiple lines, you can use the following syntax:

```
<!--
   This is a
    multi-line comment
-->
```

The comment starts with <!-- and ends with -->. You can write your comment content on multiple lines within these comment delimiters.

6.How to write HTML5 conditional comments.

Conditional comments are no longer supported in HTML5, but if you're working with older versions of HTML and need to use conditional comments, here's an example of how they were written:

```
<!--[if IE]>
   <p>This is Internet Explorer.</p>
<![endif]-->
```

```
<!--[if lt IE 9]>
   <p>This is Internet Explorer 8 or earlier.</p>
<![endif]-->
```

Within these conditional comment blocks, you can include HTML, CSS, or JavaScript code specific to the targeted version of Internet Explorer.

7.What is the use of the target attribute of <a> element? List the possible values.

The target attribute of the <a> (anchor) element in HTML is used to specify where the linked document should be opened when the user clicks on the link. It determines the target browsing context or window in which the linked URL will load. Here are the possible values for the target attribute:

1. **_blank:** When target="_blank" is specified, the linked document will open in a new browser window or tab, depending on the user's browser settings.
2. **_self:** The default behavior of links is to open the linked document in the same browsing context or window that the link was clicked in. The target="_self" attribute explicitly specifies this behavior.
3. **_parent:** If the web page containing the link is nested within frames or iframes, using target="_parent" will cause the linked document to load in the parent frame or iframe.
4. **_top:** When target="_top" is used, the linked document will open in the full body of the window, regardless of whether the current page is within a frame or iframe. It essentially breaks out of any nested frames.
5. **framename:** Instead of the predefined target values, you can specify a custom name for a target frame or iframe in your HTML. For example, target="myframe" would open the linked document in a frame or iframe with the name "myframe". This requires the frame or iframe to have a corresponding name attribute.

8. Write a note on HTML5 audio tag with its attributes.

The <audio> tag in HTML5 is used to embed audio content into a web page. It provides a standardized way to play audio files directly within the browser without requiring external plugins. Here's a note on the HTML5 <audio> tag along with its attributes:

The basic syntax of the <audio> tag is as follows:

```
<audio src="audio_file.mp3" controls></audio>
```

- **src**: The src attribute specifies the URL or file path of the audio file to be played. It can be an absolute URL or a relative path to the audio file.
- **controls**: The controls attribute enables the default audio controls to be displayed, such as play/pause, volume control, and progress bar. When added to the <audio> tag, it allows users to interact with the audio player directly.

**9.What does
 tag do? Whats the use of its clear attribute?**

The
 tag in HTML is a self-closing tag that represents a line break, meaning it creates a line break or a new line within a block-level element. It is used to add a single line break or a vertical space between two lines of text or elements.
For example, consider the following code:

```
<p>This is the first line.<br/>This is the second line.</p>
```

Resulting in the following output:
This is the first line.
This is the second line.

The **clear** attribute is used to control the behaviour of floated elements and specifies on which side of the floating elements other content should not be allowed. It can take three possible values:

- **clear="left":** Specifies that no floating elements are allowed on the left side.
- **clear="right"**: Specifies that no floating elements are allowed on the right side.
- **clear="both":** Specifies that no floating elements are allowed on either side.

For Eg:

```
<div style="float: left;">Float on the left</div>
<br clear="left" />
<p>This text will appear below the floated element.</p>
```

10.What is the purpose of tag? What's the use of its nowrap attribute.

The purpose of the tag is to provide a way to group related elements together and apply common styling or behaviour to them. It helps in organizing the layout and structure of a web page, allowing developers to create sections, columns, grids, or other structural elements.

In an HTML table, the nowrap attribute is used to specify that the content of a specific table cell should not wrap to the next line and should remain on a single line. It prevents text or content from breaking into multiple lines within a table cell.

11.What is the purpose of <dl> and <dt> elements?

The <dl> (description list) and <dt> (description term) elements in HTML are used together to create a list of terms and their corresponding descriptions. They provide a semantic way to structure and present definitions, glossaries, or other similar content on a web page.

Here's the purpose of each element:

1. **<dl> (description list):**
    - The <dl> element is used to define a description list. It acts as a container for a set of terms and their corresponding descriptions.
    - The <dl> element typically contains one or more <dt> (description term) and <dd> (description) elements as its children.
2. **<dt> (description term):**
    - The <dt> element is used to define a term or a name within a description list.
    - It represents the term or name that is being defined or described.

Here's an example of how <dl> and <dt> elements can be used together:

```
<dl>
    <dt>HTML</dt>
    <dd>HyperText Markup Language</dd>
</dl>
```

12.What is the use of <fieldset> element?

The **<fieldset>** element in HTML is used to group related form elements together and create a visual and logical separation within a form. It is typically used in conjunction with the <legend> element to provide a title or caption for the group of form controls within the <fieldset>.

The **<legend>** element is used as a child of the <fieldset> and provides a caption or title for the group of form controls. It is positioned above or to the side of the form controls and helps in describing the purpose or category of the controls within the fieldset.

For Eg:

```
<form>
  <fieldset>
     <legend>Contact Information</legend>
     <label for="name">Name:</label>
     <input type="text" id="name" name="name">
     <label for="email">Email:</label>
     <input type="email" id="email" name="email">
   </fieldset>
  <button type="submit">Submit</button>
</form>
```

13. What is the use of heading (<h1> to <h6>) tag?

The heading tags, ranging from <h1> to <h6>, are used in HTML to define different levels of headings or titles for sections of a web page. These tags represent hierarchical structure and help organize the content and provide visual hierarchy to the headings.

Here's a breakdown of the use of each heading tag:

- **<h1>**: Represents the highest-level heading on a web page. It is typically used for the main title or heading that describes the overall content of the page. There should only be one <h1> per page, and it is usually placed at the top of the document.
- **<h2> to <h6>**: Represent lower-level headings that provide subheadings or section titles within the page.

14.What is the use of document <head> tag?

The <head> tag in HTML is used to define the head section of an HTML document. It contains metadata, title, linked stylesheets, scripts, and other elements that provide information about the document and define its behavior, appearance, and relationships with external resources.

Here are some common elements you can find within the <head> tag:

1. **Title**: The <title> element is used to specify the title of the web page. It appears in the title bar of the browser window or as the tab's title when the page is bookmarked.
2. **Metadata**: Metadata elements provide additional information about the document. The most commonly used metadata element is the <meta> tag. It can be used to specify character encoding, viewport settings, author information, keywords, and descriptions for search engine optimization (SEO).
3. **Linked stylesheets:** The <link> element is used to link external CSS stylesheets to the HTML document. It allows you to separate the presentation (CSS) from the structure (HTML) of the page, making it easier to maintain and update the styling.
4. **Inline styles:** The <style> element can be used to define CSS styles directly within the HTML document. It is placed within the <head> section and contains CSS rules that are applied to the HTML elements on the page.
5. **Scripts:** The <script> element is used to include JavaScript code within the HTML document. It can be placed in the <head> section to define global scripts or within the <body> section to specify scripts specific to that section.

For Eg:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Page Title</title>
  <link rel="stylesheet" href="styles.css">
  <script src="script.js"></script>
</head>
<body>
  <!-- Content of the web page goes here -->
</body>
</html>
```

15.Write the use of <header> tag.

The <header> tag in HTML is used to define the introductory or container element at the top of a web page or a section. It typically contains the branding, logo, navigation menu, search bar, or other elements that appear at the beginning of a document or a specific section.

Here's an example of how the <header> tag can be used:

```
<header>
 <h1>Website Logo</h1>
  <nav><ul>
     <li><a href="#">Home</a></li>
     <li><a href="#">About</a></li>
     <li><a href="#">Services</a></li>
     <li><a href="#">Contact</a></li>
   </ul></nav>
</header>
```

16. What is the use of <label> tag? What is the purpose of it's for attribute?

The **<label>** tag in HTML is used to associate a label with a form element or indicate a caption for a specific piece of content. It helps in improving the usability and accessibility of web forms and provides a clear relationship between the label and the associated input or content.

The **for** attribute of the <label> tag is used to specify which form element the label is associated with. It takes the value of the id attribute of the form element. This association is established by using the same value for for and id attributes.

Here's an example of using the <label> tag to associate a label with a form element:

```
<label for="username">Username:</label>
<input type="text" id="username" name="username">
```

In this example, the <label> element has a for attribute with a value of "username", which matches the id attribute of the <input> element. This association allows users to click on the label to activate or focus the corresponding input field.

17. What is the purpose of element?

The element in HTML is an inline-level element used to group and apply styling or manipulate specific portions of text or content within a larger block-level element. It does not inherently represent any particular semantic meaning, but rather serves as a container for styling or targeting purposes.

Here's the purpose and common uses of the element:

1. **Styling and formatting:** The element is often used to apply CSS styles, such as font color, font size, background color, or text formatting, to a specific portion of text within a larger block of content. By wrapping the desired text with tags, you can target and style that text using CSS rules or inline styles.
2. **JavaScript manipulation:** The element can be used as a target for JavaScript manipulation or dynamic content generation. By assigning an ID or class to a element, you can select and manipulate its content programmatically using JavaScript.
3. **Semantic grouping:** While the element does not provide any inherent semantic meaning, it can be used to group inline elements or text within a block-level element. This can be helpful for organizing and targeting specific parts of the content for styling or scripting purposes.

Here's an example of how the element can be used:
This is a highlighted text.

18.What is the use of cellpadding and cellspacing attributes in <table> element?

The cellpadding and cellspacing attributes are used in the <table> element in HTML to control the spacing and padding between the cells within a table.

Here's the purpose of each attribute:

1. **cellpadding**: This attribute is used to specify the amount of space or padding between the content within a table cell and the edges of the cell. It defines the padding on all four sides (top, right, bottom, left) of each cell in the table. The value of cellpadding is specified in pixels.
   For example, using cellpadding="5" will add a 5-pixel padding around the content within each cell of the table.

2. **cellspacing**: This attribute is used to define the space or gap between adjacent cells in a table. It determines the spacing between cells both horizontally and vertically. The value of cellspacing is specified in pixels.
   For instance, using cellspacing="10" will create a 10-pixel gap between adjacent cells in the table.

Both cellpadding and cellspacing attributes affect the appearance and spacing of the table but in slightly different ways. cellpadding controls the space between the cell content and the cell borders, while cellspacing determines the space between adjacent cells.

Here's an example of how these attributes can be used within the <table> element:

```
<table cellpadding="10" cellspacing="5">
  <tr>
    <td>Cell 1</td>
    <td>Cell 2</td>
  </tr>
  <tr>
    <td>Cell 3</td>
    <td>Cell 4</td>
  </tr>
</table>
```

19.What is the purpose colspan and rowspan attributes of <td> tag?

The colspan and rowspan attributes are used in the <td> (table data cell) tag in HTML to define the span or merge cells horizontally (colspan) or vertically (rowspan) within a table. These attributes allow you to create table cells that span multiple columns or rows, providing flexibility in structuring and organizing tabular data.

Here's the purpose of each attribute:

1. **colspan**: This attribute is used to specify the number of columns that a table cell should span. It allows a single cell to occupy multiple columns, merging or stretching across them horizontally.
   For example, using colspan="2" on a <td> element will cause it to span two adjacent columns in the table, effectively merging those cells into a single larger cell.
2. **rowspan**: This attribute is used to specify the number of rows that a table cell should span. It allows a single cell to occupy multiple rows, merging or stretching

across them vertically.
For instance, using rowspan="3" on a <td> element will cause it to span three adjacent rows in the table, merging those cells into a single taller cell.

Here's an example that demonstrates the use of colspan and rowspan attributes:

```
<table>
  <tr>
    <td rowspan="2">Cell 1</td>
    <td colspan="2">Cell 2</td>
  </tr>
  <tr>
    <td>Cell 3</td>
    <td>Cell 4</td>
  </tr>
</table>
```

20.What is the purpose of <th> tag? List any four attributes.

The <th> tag in HTML is used to define a header cell in a table. It represents a table header, typically used to label or describe the content in a specific column or row.

Here are four commonly used attributes of the <th> tag:

1. **scope**: The scope attribute is used to specify whether a <th> element represents the header of a column (col), row (row), group of columns (colgroup), or group of rows (rowgroup).
2. **abbr**: The abbr attribute is used to provide an abbreviated version or abbreviation of the header cell's content. It can be helpful for users who may benefit from a shorter description or abbreviation of the header.
3. **colspan**: The colspan attribute, as discussed earlier, specifies the number of columns a header cell should span. It allows a single header cell to cover multiple columns.
4. **rowspan**: Similarly, the rowspan attribute specifies the number of rows a header cell should span. It allows a single header cell to cover multiple rows.

Here's an example of a table with <th> elements and some attributes:

```
<table>
  <tr>
    <th abbr=”ID” scope="col">Identification Number</th>
```

```
    <th scope="col">Name</th>
  </tr>
  <tr>
    <th scope="row">1</th>
    <td>John Doe</td>
  </tr>
  <tr>
    <th scope="row">2</th>
    <td>Jane Smith</td>
  </tr>
</table>
```

21.Mention any four features of <title> tag?

The <title> tag in HTML is used to specify the title of a web page. It appears in the browser's title bar or tab, providing a concise description of the page's content. Here are four features of the <title> tag:

1. **Browser title:** The most prominent feature of the <title> tag is that it sets the title of the web page that appears in the browser's title bar or tab. The title provides a brief summary of the page's content and helps users identify and distinguish between different open tabs or windows.
2. **Search engine optimization (SEO):** The content within the <title> tag plays a crucial role in search engine optimization. Search engines often display the title of a web page in search results, and having a descriptive and relevant title can improve the visibility and click-through rate of the page. It's important to include keywords and accurately reflect the content of the page within the <title> tag for better SEO.
3. **Bookmarking and history:** When a user bookmarks a web page or saves it to their browser's history, the title specified in the <title> tag is typically used as the default name for the bookmark or history entry. A concise and meaningful title can help users easily identify and locate previously visited pages.
4. **Accessibility:** The <title> tag also contributes to the accessibility of a web page. Screen readers and other assistive technologies can announce the title to visually impaired users, providing them with an understanding of the page's purpose or content before delving into its contents. A descriptive and informative title enhances the accessibility of the page.

22.What is the use of <ul> tag? What is the purpose of its type attribute?

The <ul> tag in HTML is used to create an unordered list. It represents a list of items that do not have a specific numerical or sequential order. The <ul> tag is typically used in conjunction with <li> (list item) tags to define each individual item within the list.

Here's the use of the <ul> tag and the purpose of its type attribute:

1. **Unordered list:** The primary purpose of the <ul> tag is to create an unordered list, where the order of the items is not important. The list items are typically marked with bullet points by default, indicating that they are part of an unordered list.
2. **Nested lists**: The <ul> tag can be nested within other list elements, such as <li> (list item) tags or even another <ul> tag. This allows for the creation of nested or hierarchical lists, where sub-lists can be used to group related items together.
3. **type attribute:** The type attribute of the <ul> tag is used to specify the style of bullet points or markers used for the list items. It can take several values to customize the appearance of the list:
    - **type="disc"** (default): This is the default style, representing the bullet point as a filled circle.
    - **type="circle":** This style displays the bullet point as an empty circle.
    - **type="square":** This style uses a filled square as the bullet point.
    - **type="none":** This value removes the bullet points altogether, resulting in a plain list without any markers.

By default, most browsers render unordered lists with bullet points, but the type attribute allows you to customize the appearance based on your design preferences.

Here's an example of an unordered list using the <ul> tag and the type attribute:

```
<ul type="circle">
  <li>List item 1</li>
  <li>List item 2</li>
  <li>List item 3</li>
</ul>
```

23. What are the ways to include JavaScript in a webpage? Give example code for each.

There are several ways to include JavaScript in a webpage. Here are four common methods:

I. Internal Javascript or Embedding code:-

To add the JavaScript code into the HTML pages, we can use the <script>.....</script> tag of the HTML that wrap around JavaScript code inside the HTML program. Users can also define JavaScript code in the <body> tag (or we can say body section) or <head> tag because it completely depends on the structure of the web page that the users use.

```
<!DOCTYPE html >
<html>
<head>
        <title> page title</title>
        <script>
        document.write("Welcome to Javatpoint");
        </script>
</head>
<body>
        <p>JavaScript in the head section </p>
</body>
</html>
```

II. Inline Javascript or Inline code:-

Generally, this method is used when we have to call a function in the HTML event attributes. There are many cases (or events) in which we have to add JavaScript code directly eg., OnMouseOver event, OnClick, etc. Here we can add JavaScript directly in the html *without using the <script>.... </script> tag*.

```
<!DOCTYPE html >                        ( Eg: inline javascript )
<html>
<head> <title> page title</title>
</head>  <body>
<p>   Inline JavaScript
<a href="#" onClick="alert('Welcome !');">Click Me</a>  </p>
</body> </html>
```

III. External javascript:-

We can also create a separate file to hold the code of JavaScript with the (.js) extension and later incorporate/include it into our HTML document using the src attribute of the <script> tag. It becomes very helpful if we want to use the same code in multiple HTML documents. It also saves us from the task of writing the same code over and over again and makes it easier to maintain web pages.

Here “type” attribute Specifies the media type of the script , which is optional and “src” Specifies the URL of an external script file

(Eg : external js)

```
<html>
<head>
        <title>Including a External JavaScript File</title>
</head>
<body>
        <form>
                <input type="button" value="Result" onclick="display()"/>
        </form>
        <script src="hello.js"></script>
</body>
</html>
```

And the external hello.js file will contain :

```
function display()
{
        alert("Hello World!");
}
```

The external javascript file will not contain <script> tag, as similar to external css file which will not contain <style> tag. The file extension itself specifies which type of file it is.

24. What is the use of <noscript></noscript> tag?

One way of providing alternate content for those viewers without JavaScript (or with JavaScript turned off) is to use the noscript tag. The <noscript></noscript> tags may be placed anywhere in the HTML document and can contain any content needed for those viewers browsing without JavaScript (such as viewers using mobile browsers like the ones on a Blackberry or iPhone).

For example:

```
<script type="text/javascript">
        document.write("The color is red.");   //  viewers with JavaScript
</script>
<noscript>
        The color is red.    // viewers without JavaScript
</noscript>
```

25.What is a variable? How to declare variable in JavaScript? Give example.

A variable represents or holds a value. The actual value of a variable can be changed at any time.

Declaring Variables

To declare text as a variable, you use the var or let keyword, which tells the browser that the text to follow will be the name of a new variable:

var variablename;

For example, to name your variable coolcar, the declaration looks like this:

var coolcar;

To assign a value to a variable, you use the JavaScript assignment operator, which is the equal to (=) symbol. If you want to declare a variable and assign a value to it on the same line, use this format:

var variablename=variablevalue;

For example, to name your variable paycheck and give it the numeric value 1200, use this statement:

var paycheck=1200;

26.What is an anonymous function in JavaScript? Provide proper example.

Anonymous Functions One effective use of the function expression is to use it *without a function name* to create an anonymous function. An anonymous function is one *that is created and called at the same time*, and is helpful when you wish to call a function in only one place in your code (rather than declaring the function elsewhere in the code and reusing it by calling it more than once). The following is the general format for an anonymous function:

```
var varname = function(parameters) {
        Code for function
};
```

Anonymous functions are quite useful when dealing with JavaScript events. For example, to react to a user clicking the mouse while on a Web page, you could write a simple function for a click event on the document and then call it, as in the following code:

```
function do_not_click() {
        window.alert("Do not click on my page!");
}
document.onclick = do_not_click;
```

This declares the function, then calls it afterward (without parentheses) to handle the click event. However, you could combine these two steps into one using an anonymous function, as follows:

```
document.onclick = function() {
        window.alert("Do not click on my page!");
};
```

27.What are the uses of === and !== operators of JavaScript.

The Strict Is-Equal-To Operator (===)

the === operator to return true, the values or statements on each side must be equal and must be of the same type. This means that if you use a statement such as 3===“3”, the operator will return false because the value on the left is a number and the value on the right is a string; the values are of different types. Whereas the is-equal-to (==) operator first attempts to convert the values on each side of the operator to the same type and then determines if they are equal, the strict is-equal-to operator automatically returns false if the values are not of the same type.

For Example : 4===4 True Two equal numbers

‘4’===4 False Values on each side are of different types

The Strict Is-Not-Equal-To Operator (!==)

For the !== operator to return true, the values or statements on each side must not be equal or must not be of the same type. This means that if you use a statement such as 3!=="3", the operator will return true because the value on the left is a number and the value on the right is a string; the values are of different types (and thus not strictly equal). Whereas the is-not-equal-to (!=) operator first attempts to convert the values on each side of the operator to the same type and then determines if they are not equal, the strict is-not-equal-to operator (!==) automatically returns true if the values are not of the same type.

For Example : "4"!==4 True Values on each side are of different types

4!==4 False 4 is equal to 4

28. List any four special operators of JavaScript with their purpose.

Operator	Symbol	Purpose
Conditional	?:	Often used as a short if/else type of statement. A condition is placed before the question mark (?) and a value is placed on each side of the colon (:).
Comma	,	Evaluates the statements on both sides of the operator, and returns the value of the second statement.
Delete	delete	Used to delete an object, a property, or an element in an array.
In	in	Returns true if a property is in a specified object.
Instanceof	instanceof	Returns true if an object is of a specified object type.
New	new	Creates an instance of an object.
This	this	Refers to the current object.
Typeof	typeof	Returns a string that tells you the type of the value being evaluated.
Void	void	Allows an expression to be evaluated without returning a value.

Table 5-5 Special Operators

29.Write the purpose of for in and for each in loops of JavaScript.

The for in loop allows you to loop over all the names of the properties of an object and execute statements for each property using a variable to hold the name of the property. The general format for a for in loop is shown here:

```
for (varname in objectname) {
        JavaScript Code Here
}
```

Example :

```
const person = {
        name: 'John',
        age: 30,
        occupation: 'Developer'
};

for ( var key in person) {
        console.log(key + ': ' + person[key]);
}
```

The for each in loop is very similar to the for in loop, but rather than looping through the name of each property it allows you to loop through the value of each of the properties.

Example :

```
const numbers = [1, 2, 3, 4, 5];
numbers.forEach(function(element) {
        console.log(element);
});
```

30. What do you mean by an Event? What is an event handler?

An event is something that happens when the viewer of the page performs some sort of action, such as clicking a mouse button, clicking a button on the page, changing the contents of a form element, or moving the mouse over a link on the page. Events can also occur simply by the page loading or other similar actions.

An event handler is a predefined JavaScript property of an object (in most cases an element in the document) that is used to handle an event on a Web page.

31.What is the use of the attachEvent() method? Give an example.

The **attachEvent()** method works in a similar way to addEventListener().
attachEvent() is to bind an event handler function to an element, allowing the function to be executed when the specified event occurs on that element.

Here's an example of using the attachEvent() method to attach a click event handler to a button

```
var button = document.getElementById('myButton');
button.attachEvent('onclick', function() {
        alert('Button clicked!');
});
```

32.What are object initializers? Provide a proper example.
Object initializers, also known as object literals, are a shorthand syntax in JavaScript for creating and defining objects.An object initializer is a little bit shorter than a constructor function. The following is the syntax of an object initializer:

```
object_name={property:value}
```

Here's an example of using object initializers to create and define an object:

```
const person = {
  name: 'John',
  age: 30,
  occupation: 'Developer',
};
```

In this example, an object named person is created using object initializers. The object has properties such as name, age, and occupation. The properties are defined within the curly braces {} using a key-value pair syntax.

DESCRIPTIVE QUESTIONS

1.List and explain some of the most commonly used (X)HTML elements.

Here are some of the most commonly used (X)HTML elements along with a brief explanation of their purpose:

1. **<html>:** The <html> element is the root element of an HTML document. It represents the entire HTML document and contains all other elements.
2. **<head>:** The <head> element is a container for metadata and other head elements, such as the page title, character encoding, CSS stylesheets, and JavaScript scripts. It provides information about the document but is not displayed on the webpage.
3. **<title>:** The <title> element is used to define the title of an HTML document. It appears in the browser's title bar or tab and is often displayed by search engines.
4. **<body>:** The <body> element represents the content of an HTML document. It contains all the visible elements that are displayed on the webpage, including text, images, links, headings, paragraphs, and more.
5. **<h1> to <h6>:** Heading elements (<h1> to <h6>) are used to define section headings in a document, with <h1> being the highest level (most important) heading and <h6> being the lowest level (least important) heading.
6. **<p>:** The <p> element is used to define a paragraph of text. It represents a block of text that is separated from adjacent paragraphs by vertical spacing.
7. **<a>:** The <a> element is used to create a hyperlink. It allows you to link to other web pages, files, email addresses, or specific parts of a document using the href attribute.
8. **<img>:** The <img> element is used to embed an image in an HTML document. It requires the src attribute to specify the URL or file path of the image.
9. ** and <li>:** The <ul> element is used to create an unordered (bulleted) list, and the <li> element represents each individual list item within the list.
10. ** and <li>:** The <ol> element is used to create an ordered (numbered) list. It behaves similarly to <ul>, but the list items are automatically numbered.
11. **<table>, <tr>, <th>, and <td>:** These elements are used to create tables in HTML. <table> defines the table itself, <tr> represents a table row, <th> defines a table header cell, and <td> represents a regular table data cell.
12. **<form>, <input>, and <button>:** These elements are used to create HTML forms for user input. <form> defines the form, <input> is used for various input fields like text, checkboxes, radio buttons, etc., and <button> creates a clickable button.

2. Explain HTML AND XHTML DTD specifications with example.

DTD (Document Type Definition) is a specification that defines the structure and syntax rules for an XML or SGML-based document. Both HTML and XHTML have their respective DTD specifications. Let's explore HTML and XHTML DTD specifications with examples:

1. HTML DTD:

- HTML (Hypertext Markup Language) is a widely used markup language for creating web pages. HTML has different versions, and each version has its DTD specification.
- The HTML DTD specifies the rules and allowed elements for a particular version of HTML.
- Example: The HTML5 specification doesn't rely on a DTD. Instead, it uses a simplified doctype declaration:

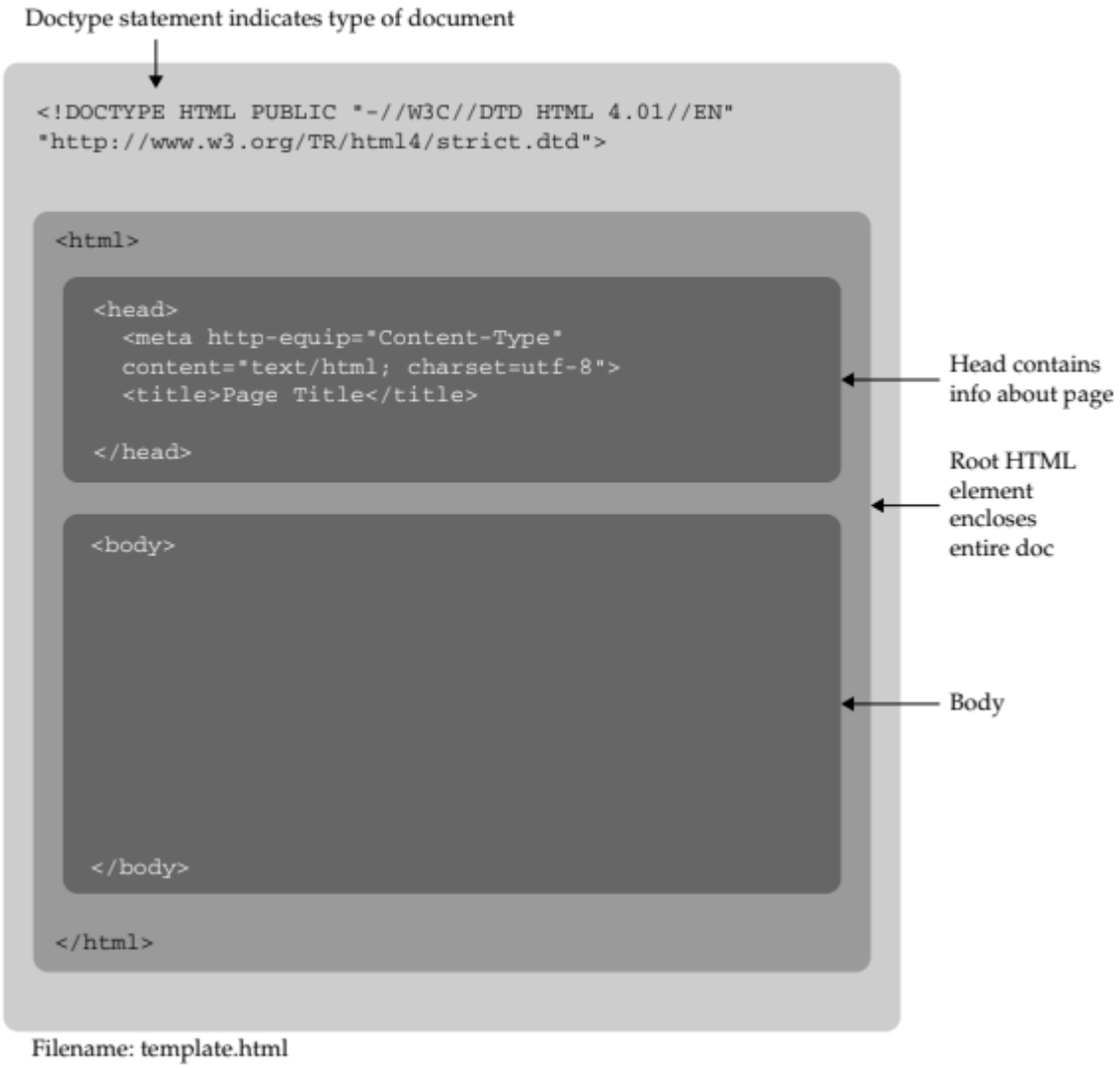
```
<!DOCTYPE html>
<html>
 <head>  <title>My HTML Page</title> </head>
 <body>
        <h1>Welcome to my webpage!</h1>
         <p>This is a paragraph of text.</p>
 </body>
</html>
```

2.XHTML DTD:

- XHTML (Extensible Hypertext Markup Language) is a stricter and more XML-compliant version of HTML. It combines the syntax of XML with the HTML structure.
- XHTML follows specific rules and requires a DTD specification to ensure well-formed and valid XML documents.
- Example: The XHTML 1.0 Strict DTD declaration:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD  XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head> <title>My XHTML Page</title> </head>
 <body>
   <h1>Welcome to my webpage!</h1>
   <p>This is a paragraph of text.</p>
 </body>
</html>
```

3.Explain the HTML document structure with the help of a diagram.



In this diagram:

- The **<!DOCTYPE>** declaration specifies the version of HTML being used and informs the browser about the document type.
- The **<html>** element serves as the root element of the HTML document.
- Inside the **<html>** element, we have the **<head>** element, which contains metadata and other information about the document, such as the page title, CSS stylesheets, and JavaScript scripts.
- The **<title>** element within the **<head>** element defines the title of the document, which appears in the browser's title bar or tab.
- After the **<head>** element, we have the **<body>** element, which contains the visible content of the webpage, including text, images, links, headings, paragraphs, and other elements.
- The content within the **<body>** element represents what will be rendered and

displayed on the webpage.
- The **<body>** element can contain various nested elements and structure, defining the overall layout and organization of the webpage content.

This basic structure provides a foundation for building HTML documents and organizing their content in a structured and meaningful way.

4. Explain all possible elements inside the HTML document head. (Q.14)

The **<head>** element in an HTML document contains metadata and other head elements that provide information about the document and define its behavior. Here are the possible elements that can be included inside the **<head>** element:

1. **<title>:** The **<title>** element is used to define the title of the HTML document. It appears in the browser's title bar or tab and is often displayed by search engines.
2. **<meta>:** The **<meta>** element is used to provide metadata about the HTML document. It includes information such as character encoding, viewport settings, author, description, keywords, and more.
3. **<link>:** The <link> element is used to link external resources to the HTML document, such as CSS stylesheets, icon files, and alternative versions of the document. It defines the relationship between the current document and the linked resource.
4. **<style>:** The <style> element is used to embed CSS (Cascading Style Sheets) directly within the HTML document. It allows you to define the style rules that will be applied to elements in the document.
5. **<script>:** The <script> element is used to embed or reference JavaScript code within the HTML document. It allows you to add interactivity, manipulate the DOM (Document Object Model), and perform other client-side scripting tasks.
6. **<noscript>:** The <noscript> element is used to provide alternate content that should be displayed when JavaScript is disabled or not supported by the browser. It typically contains text or instructions that inform users about the need to enable JavaScript.
7. **<meta charset="UTF-8">:** This specific **<meta>** element is commonly used to specify the character encoding for the document. It ensures that the document is interpreted and displayed correctly by browsers.

These elements, when used inside the <head> element, provide crucial information about the HTML document, its presentation, and its behaviour. They play a vital role in search engine optimization, accessibility, and proper rendering of the document in different browsers and devices.

5. Explain (X)HTML rules.

(X)HTML follows a set of rules and guidelines to ensure well-formed and valid documents. These rules define the structure, syntax, and behavior of the markup language. Here are some key rules for (X)HTML:

1. **Nesting and Hierarchy:** (X)HTML elements must be properly nested and follow a hierarchical structure. Each opening tag must have a corresponding closing tag, and elements cannot overlap or be improperly nested.
2. **Element Names:** (X)HTML element names must be written in lowercase. While browsers are forgiving and often interpret uppercase names correctly, adhering to lowercase is considered best practice for consistency and compatibility.
3. **Attribute Values:** Attribute values in (X)HTML must be enclosed in quotes, either double quotes or single quotes. This ensures that attribute values are clearly defined and correctly interpreted.
4. **Closing Tags:** Most (X)HTML elements require a closing tag, except for a few self-closing elements like
, <img>, and <input>. It is important to close tags properly to maintain the integrity of the document structure.
5. **Empty Elements:** Some (X)HTML elements do not require any content and are considered empty. These elements, such as
, <img>, and <input>, are self-closing and do not require a closing tag. They are written as <element /> or <element>. For example:
 or .
6. **Attribute Names:** Attribute names in (X)HTML are case-insensitive and should be written in lowercase for consistency. It is common practice to use lowercase attribute names, although uppercase names are also recognized by browsers.
7. **Quoting Attribute Values:** Attribute values in (X)HTML should be enclosed in quotes, either double quotes (") or single quotes ('). This is necessary to distinguish attribute values from other content and to ensure correct interpretation by browsers.
8. **Character Encoding:** It is essential to specify the character encoding for (X)HTML documents using the <meta charset="charset-name"> tag within the <head> element. The most commonly used character encoding is UTF-8, which supports a wide range of characters.

6.Explain HTML5 structural elements.

HTML5 introduced several new structural elements that provide semantic meaning to different sections of a webpage. These elements help to improve the document's structure, accessibility, and search engine optimization. Here are some of the key HTML5 structural elements:

1. **<header>:** The **<header>** element represents the introductory or navigational content for a section or the whole document. It typically contains the site logo, page title, navigation menu, and other header-related content.
2. **<nav>:** The **<nav>** element represents a section of a document that contains navigation links. It is used to define a navigation menu or a collection of links that allow users to navigate through different parts of the website or related pages.
3. **<main>:** The **<main>** element represents the main content of the document. It should contain the primary content that is unique to the webpage, such as articles, blog posts, product descriptions, or any other significant content relevant to the document's purpose.
4. **<article>:** The **<article>** element represents a self-contained composition within a document. It is suitable for content that can be distributed or independently syndicated, such as blog posts, news articles, forum posts, or any content that can stand alone.
5. **<section>:** The **<section>** element represents a standalone section within a document. It helps in organizing and grouping related content together. For example, a webpage might have multiple sections for introduction, features, testimonials, or any other logical divisions.
6. **<aside>:** The **<aside>** element represents content that is tangentially related to the main content, such as sidebars, pull quotes, or advertisements. It is typically used for content that can be considered separate from the main flow of the document but still related or supportive.
7. **<footer>:** The **<footer>** element represents the footer section of a document or a section within it. It usually contains information about the author, copyright notice, contact details, site map, or any other relevant information related to the document or the website.

7. Explain new HTML5 form fields.

HTML5 introduced several new form fields that provide enhanced functionality and user experience. Here are some of the new HTML5 form fields:

1. **<input type="email">:** The <input> element with the type="email" attribute is used to capture email addresses. It provides validation to ensure that the entered value is a valid email address. Browsers may display a specific keyboard layout for email input on mobile devices.
2. **<input type="url">:** The <input> element with the type="url" attribute is used to capture URLs or web addresses. It validates the entered value to ensure it is a valid URL format. Browsers may also provide specific keyboard layouts for URL input on mobile devices.
3. **<input type="tel">:** The <input> element with the type="tel" attribute is used to capture telephone numbers. It can help enforce phone number formats or provide specific keyboard layouts for phone number input on mobile devices.
4. **<input type="number">:** The <input> element with the type="number" attribute is used to capture numeric input. It provides built-in validation to ensure that only numeric values are entered. Browsers may also display a numeric keyboard on mobile devices.
5. **<input type="range">:** The <input> element with the type="range" attribute is used to create a slider control. It allows users to select a value from a specified range by dragging a slider thumb.
6. **<input type="date">, <input type="time">**: These input types are used to capture specific date, time, or date-time values. They provide a user-friendly interface for selecting dates, times, or both, depending on the input type.
7. **<input type="color">:** The <input> element with the type="color" attribute is used to create a color picker. It allows users to select a color from a palette or enter a color code.
8. **<input type="file">:** The <input> element with the type="file" attribute is used to create a file upload field. It allows users to select and upload files from their local device.

8.Write a short note on <!DOCTYPE> element.

The **<!DOCTYPE>** element, also known as the Document Type Declaration, is an important component of an HTML document. It specifies the version of HTML or XHTML being used and helps browsers interpret and render the document correctly.

Here are some key points about the **<!DOCTYPE>** element:

1. **Declaration Syntax:** The <!DOCTYPE> element is placed at the very beginning of an HTML document, **before the <html>** tag. It is written as **<!DOCTYPE html>** for HTML5, which is the recommended doctype for modern web development.
2. **Document Type:** The <!DOCTYPE> element specifies the document type definition (DTD) or schema that defines the markup language being used. In the case of **<!DOCTYPE html>**, it indicates that the document follows the **HTML5 specification.** It informs the browser about the version of HTML or XHTML being used in the document.
3. **Compatibility Modes:** The <!DOCTYPE> declaration helps browsers determine the rendering mode to use. With a valid and correct doctype declaration, browsers enter the standards mode, which ensures consistent rendering and behaviour across different browsers.
4. **Validation and Error Handling:** The <!DOCTYPE> declaration is also important for validation purposes. It allows validators to check the HTML document against the appropriate specification and identify any errors or inconsistencies.
5. **XHTML Doctype:** In addition to HTML5, different doctype declarations exist for XHTML, such as <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">. These declarations define the rules and syntax for XHTML documents.

It is important to include the <!DOCTYPE> declaration in your HTML document to ensure proper rendering, standard compliance, and compatibility across browsers. For modern web development, using <!DOCTYPE html> as the doctype declaration for HTML5 is recommended, as it provides the most up-to-date features and compatibility.

9. What is the use of <a> tag? Mention any 5 element specific attributes of it.

The **<a>** tag in HTML is used to create hyperlinks, allowing users to navigate to other web pages or different sections within the same page. Here are five commonly used attributes of the **<a>** tag:

1. **href**: The href attribute specifies the URL or destination where the link should navigate to. It can be an absolute URL, relative URL, or an anchor reference within the same document.
2. **target**: The target attribute defines how the linked page should be opened. It specifies the target window or frame where the linked content should be displayed. Common values include
   _blank to open the link in a new window or tab,
   _self to open in the same window,
   _parent to open in the parent frame, and
   _top to open in the full body of the window.
3. **title**: The title attribute provides additional information about the link. When users hover over the link, the value of the title attribute is displayed as a tooltip, providing a description or explanation of the linked content.
4. **download**: The download attribute, when specified, indicates that the linked file should be downloaded instead of opened in the browser. It allows users to save the file directly to their device.
5. **id**: The id attribute specifies a unique identifier for the <a> element. It can be used to target and style the link using CSS or to create anchors within the same document that can be linked to using anchor references.
6. **class**: The class attribute assigns one or more class names to the <a> element. It allows you to apply CSS styles or JavaScript behaviours to specific links based on their class.

10. Explain the use of <article> element of HTML5 with any five unique attributes.
The <**article**> element in HTML5 is used to define a self-contained, independent piece of content within a document. It represents a complete or standalone composition that can be distributed or syndicated independently from the rest of the page. Here are five unique attributes associated with the <article> element:

1. **id**: The id attribute specifies a unique identifier for the <article> element. It can be used to target and style the specific article using CSS or to create anchors within the document that can be linked to using anchor references.
2. **class**: The class attribute assigns one or more class names to the <article>

element. It allows you to apply CSS styles or JavaScript behaviours specifically to the article based on its class.
3. **lang**: The lang attribute specifies the language of the content within the <article> element. It helps assistive technologies and search engines understand and handle the content correctly based on the specified language.
4. **style**: The style attribute allows you to apply inline CSS styles to the <article> element. It enables you to add specific styling, such as colors, margins, padding, or backgrounds, directly to the article.
5. **title**: The title attribute provides a text description or a tooltip for the <article> element. When a user hovers over the article, the value of the title attribute is displayed, offering additional information about the content.

11. Write a short note on <aside> with any of its 5 unique attributes.

The **<aside>** element in HTML5 represents a section of the page that can be considered separate from the primary content but still relevant and related to it. Here's a short note on the <aside> element along with one of its unique attributes:

The **<aside>** element is typically used for content that is not directly related to the main flow of the document but provides additional information or supplementary content. It can be used for sidebars, pull quotes, advertisements, related links, author information, or other supporting content.

Explain the Attributes Like **: id , class , style , title , lang**

12. Explain any 4/6 unique element specific attributes of <body> tag.

The <body> tag in HTML represents the main content area of a web page. It contains all the visible content that is displayed to the user. Here are four commonly used and unique attributes associated with the <body> tag:

1. **onload**: The onload attribute specifies a JavaScript function to be executed when the web page has finished loading. It allows you to trigger specific actions or functionality after the page has been fully rendered.
2. **onunload**: The onunload attribute specifies a JavaScript function to be executed when the user navigates away from the web page. It can be used to perform cleanup tasks or save data before the page is closed or redirected.
3. **bgColor**: The bgColor attribute sets the background color of the <body> element. You can specify the color using a hexadecimal value or predefined

color names. This attribute is commonly used to define the background color for the entire web page.

4. **text**: The text attribute sets the default text color for the content within the <body> element. You can specify the color using a hexadecimal value or predefined color names. This attribute allows you to define the default color for the text within the page.
5. **link**: The link attribute sets the default color for unvisited links within the <body> element. You can specify the color using a hexadecimal value or predefined color names. This attribute allows you to define the default color for links before they are visited by the user.
6. **vLink**: The vLink attribute sets the default color for visited links within the <body> element. You can specify the color using a hexadecimal value or predefined color names. This attribute allows you to define the default color for links that have been previously visited by the user.

13. Explain any 4/6 unique element specific attributes of <button> tag.

The <button> tag in HTML is used to create a clickable button on a web page. It can be used to trigger JavaScript functions, submit form data, or navigate to another page. Here are four commonly used and unique attributes associated with the <button> tag:

1. **type**: The type attribute specifies the type of button. It can have different values like "submit" (default), "reset", or "button". The "submit" type is used to submit a form, "reset" type is used to reset form input values, and "button" type is used for generic button functionality without any specific action.
2. **disabled**: The disabled attribute, when present, disables the button. A disabled button cannot be clicked or triggered by the user. It is commonly used to indicate that an action is not available or to prevent user interaction until certain conditions are met.
3. **name**: The name attribute specifies the name of the button when it is submitted as part of a form. It is used to identify the button and its value on the server-side when processing form data. This attribute is especially useful when multiple buttons are present in a form, allowing you to differentiate them.
4. **value**: The value attribute specifies the value associated with the button when it is submitted as part of a form. It represents the data that will be sent to the server when the button is clicked and the form is submitted. This attribute allows you to define the specific value or data related to the button.

14. Explain any 4/6 unique element specific attributes of <font> tag.

The <**font**> tag is an outdated and deprecated HTML tag that was used to specify font styles, colors, and sizes for text. However, it is strongly advised to avoid using this tag as it is not supported in HTML5 and is considered bad practice. Instead, CSS should be used for styling purposes. Nevertheless, here are four unique attributes associated with the <font> tag:

1. **face**: The face attribute was used to specify the font family for the text enclosed within the <font> tag. You could provide a comma-separated list of font names, and the browser would attempt to display the text in the first available font from the list.
2. **size**: The size attribute was used to specify the size of the font. It accepted values from 1 to 7, where 1 represented the smallest size, and 7 represented the largest size. However, using relative or absolute font sizes with CSS is recommended instead.
3. **color**: The color attribute was used to define the text color. You could specify the color using a named color (e.g., "red") or a hexadecimal color value (e.g., "#FF0000"). Again, it is preferable to use CSS for defining text colors.
4. **bgcolor**: The bgcolor attribute was used to set the background color behind the text. You could specify the background color using a named color or a hexadecimal color value. However, CSS should be used for background color styling.

15. What is <form> tag? Write any of its 5 element specific attributes

The **<form>** tag in HTML is used to create an interactive form on a web page. It encapsulates form controls such as input fields, checkboxes, radio buttons, buttons, and more. The <form> tag defines a boundary for the form, allowing user input to be collected and submitted to a server for processing. Here are five commonly used element-specific attributes of the <form> tag:

1. **action**: The action attribute specifies the URL or script to which the form data should be submitted when the user clicks the submit button. It defines the server-side script or endpoint that will process the form data. For example: <form action="/submit-form" method="post">.
2. **method**: The method attribute specifies the HTTP method to be used when submitting the form data. It can be either "get" or "post". The "get" method appends the form data to the URL, while the "post" method sends the form data in the body of the HTTP request. For example: <form action="/submit-form"

method="post">.

3. **name**: The name attribute specifies a unique name for the form. It is used to identify the form when processing the form data on the server-side. The name attribute is particularly useful when multiple forms are present on a web page. For example: <form name="myForm" action="/submit-form" method="post">.
4. **target**: The target attribute specifies where the form response should be displayed when the form is submitted. It can be set to "_self" to load the response in the same window, "_blank" to open the response in a new window or tab, or a specific frame name to load the response in a particular frame. For example: <form action="/submit-form" method="post" target="_blank">.
5. **autocomplete**: The autocomplete attribute specifies whether the browser should enable or disable autocomplete for form input fields. It can be set to "on" to enable autocomplete or "off" to disable it. This attribute provides control over whether the browser should remember and suggest previously entered values for form fields. For example: <form action="/submit-form" method="post" autocomplete="off">.

16. Write a not <hr> tag with all of its element-specific attributes.

The **<hr>** tag in HTML is used to create a horizontal rule or a thematic break in the content. It is a self-closing tag, which means it does not require a closing tag. The <hr> tag does not have any element-specific attributes. However, it does support the following global attributes that can be applied to it:

1. **class**: The class attribute allows you to assign one or more CSS classes to the <hr> element. CSS classes can be used to style the horizontal rule or apply specific visual effects.
2. **id**: The id attribute provides a unique identifier for the <hr> element. It can be used to target the element using CSS or JavaScript for styling or manipulation purposes.
3. **style**: The style attribute allows you to apply inline CSS styles directly to the <hr> element. You can specify various style properties such as color, width, height, margin, and more.
4. **title**: The title attribute specifies additional information about the <hr> element. It can be used to provide a tooltip or description when the user hovers over or interacts with the horizontal rule.

17. Write the four element specific attributes of <html> element with their purpose.

The <**html**> element in HTML is the root element of an HTML document. It represents the entire HTML document structure. Here are four element-specific attributes of the <html> element along with their purpose:

1. **lang**: The lang attribute specifies the primary language of the HTML document. It helps define the language for screen readers and search engines, allowing them to present and process the content appropriately. The attribute value should be a valid language code, such as "en" for English or "es" for Spanish. Example: <html lang="en">
2. **dir**: The dir attribute specifies the text directionality of the content within the HTML document. It is used for languages that are written from right to left (RTL) such as Arabic or Hebrew. The attribute value can be set to "ltr" for left-to-right or "rtl" for right-to-left. Example: <html dir="ltr">
3. **manifest**: The manifest attribute is used to specify the URL of a cache manifest file. The cache manifest file contains instructions for the browser on how to cache and manage the resources associated with the HTML document, enabling offline access and improved performance. Example: <html manifest="example.manifest">
4. **xmlns**: The xmlns attribute is used in XHTML to declare the XML namespace of the HTML document. It defines the XML namespace for the XHTML elements and attributes used in the document. Example: <html xmlns="http://www.w3.org/1999/xhtml">

18. Explain the <iframe> element with it element specific attributes

The <**iframe**> (Inline Frame) element in HTML is used to embed another HTML document within the current document. It creates a rectangular region on the web page where the embedded content, typically another webpage, is displayed. The <iframe> element has several element-specific attributes that allow you to control its behaviour and appearance. Here are some of the commonly used element-specific attributes:

1. **src**: The src attribute specifies the URL of the content to be displayed within the <iframe>. It can be a relative or absolute URL pointing to an HTML document or any other valid web resource. Example: <iframe src="https://www.example.com"></iframe>
2. **width**: The width attribute sets the width of the <iframe> in pixels or as a percentage of the available space. You can specify a specific value, such as

300px, or a percentage value, such as 50%. Example: <iframe src="content.html" width="400px"></iframe>

3. **height**: The height attribute sets the height of the <iframe> in pixels. It defines the vertical dimension of the iframe region. Example: <iframe src="content.html" height="200px"></iframe>
4. **sandbox**: The sandbox attribute enables a restricted "sandbox" mode for the <iframe>. When the sandbox attribute is present, the content within the <iframe> is subject to a set of restrictions, such as not being able to execute JavaScript or submit forms. Example: <iframe src="content.html" sandbox></iframe>
5. **frameborder**: The frameborder attribute specifies whether or not to display a border around the <iframe>. By default, it is set to "1", which displays the border. Setting it to "0" removes the border. Example: <iframe src="content.html" frameborder="0"></iframe>
6. **allowfullscreen**: The allowfullscreen attribute enables or disables the ability to display the <iframe> in fullscreen mode. When present, it allows the user to enter fullscreen mode when interacting with the <iframe>. Example: <iframe src="video.html" allowfullscreen></iframe>

19. Explain <img> tag with its element specific attributes

The **<img>** tag in HTML is used to embed an image into a web page. It is a self-closing tag, which means it does not require a closing tag. The <img> tag has several element-specific attributes that allow you to specify the source and display of the image. Here are some commonly used element-specific attributes of the <img> tag:

1. **src**: The src attribute specifies the URL or file path of the image to be displayed. It can be a relative or absolute URL pointing to an image file. Example: <img src="image.jpg" alt="Image">
2. **alt:** The alt attribute provides alternative text for the image. It is used as a fallback when the image cannot be displayed or for accessibility purposes. The alternative text should describe the image's content or purpose. Example: <img src="image.jpg" alt="A beautiful sunset">
3. **width:** The width attribute sets the width of the image in pixels. It specifies the horizontal dimension of the image display area. Example: <img src="image.jpg" alt="Image" width="300">
4. **height**: The height attribute sets the height of the image in pixels. It specifies the vertical dimension of the image display area. Example: <img src="image.jpg" alt="Image" height="200">

5. **title**: The title attribute provides additional information about the image. It is displayed as a tooltip when the user hovers over the image. It can be used to provide more details or a descriptive caption for the image. Example: <img src="image.jpg" alt="Image" title="Sunset at the beach">

20. Explain <input> tag with its element specific attributes

The <**input**> tag in HTML is used to create various types of input fields or controls within a web form. It allows users to enter and submit data. The <input> tag has several element-specific attributes that control its behavior and appearance. Here are some commonly used element-specific attributes of the <input> tag:

1. **type**: The type attribute specifies the type of input control to be created. It determines the kind of data that can be entered or selected by the user. Examples include text input (type="text"), password input (type="password"), checkbox input (type="checkbox"), radio button input (type="radio"), etc. Example: <input type="text" name="username">
2. **name**: The name attribute defines the name of the input field. It is used to identify the input field when the form is submitted to the server. The name is important for data processing and server-side handling. Example: <input type="text" name="username">
3. **value**: The value attribute sets the initial value of the input field. It can be pre-filled with a default value or dynamically populated with a value from a server-side script or JavaScript. Example: <input type="text" name="username" value="John Doe">
4. **placeholder**: The placeholder attribute provides a short hint or example text that is displayed inside the input field before the user enters any value. It is used to give users an idea of what kind of input is expected. Example: <input type="text" name="email" placeholder="Enter your email">
5. **required**: The required attribute specifies that the input field must be filled out before the form can be submitted. It is used for mandatory fields to ensure that the user provides the necessary information. Example: <input type="text" name="name" required>
6. **disabled**: The disabled attribute disables the input field, preventing the user from interacting with it or entering data. It is commonly used to indicate read-only fields or to temporarily disable form inputs. Example: <input type="text" name="username" value="John Doe" disabled>

21. Explain <li> tag with its attributes.

The **<li>** (list item) tag in HTML is used to define an item in a list. It is a child element of <ul> (unordered list), <ol> (ordered list), or <menu> elements. The <li> tag does not have any specific attributes of its own, but it inherits attributes from its parent elements. However, there are some commonly used attributes that can be applied to the <li> tag indirectly through CSS. Here are a few examples:

1. **value**: The value attribute is used with the <li> tag when it is a child of an <ol> (ordered list) element. It specifies the value of the list item. By default, the value is determined by the position of the <li> tag within the list. However, you can override this by specifying a custom value.
   Example: <ol> <li value="1">First item <li value="2">Second item <li value="3">Third item
2. **class**: The class attribute allows you to assign a CSS class to the <li> tag. It is used for styling purposes or to target the list items with CSS or JavaScript.
   Example: <ul> <li class="highlight">Item 1 <li class="highlight">Item 2 Item 3
3. **id**: The id attribute is used to give a unique identifier to the <li> tag. It is often used to target specific list items for styling or scripting purposes.
   Example: <ul> <li id="item1">First item <li id="item2">Second item <li id="item3">Third item
4. **style**: The style attribute allows you to apply inline CSS styles to the <li> tag. It is used to define specific styling properties for individual list items.
   Example: <ul> <li style="color: red;">Item 1 <li style="font-weight: bold;">Item 2 Item 3

22. Explain the use of <link> tag with its attributes.

The <**link**> tag in HTML is used to link external resources, such as stylesheets and icons, to an HTML document. It is a self-closing tag and does not require a closing tag. The <link> tag has several attributes that define the relationship between the current document and the linked resource. Here are the commonly used attributes of the <link> tag:

1. **rel**: The rel attribute specifies the relationship between the current document and the linked resource. It tells the browser how to interpret the linked resource. For example, rel="stylesheet" is used to link an external CSS stylesheet, rel="icon" is used to specify the favicon of the document, and rel="preload" is used to indicate a resource that should be preloaded in advance. Example: <link rel="stylesheet" href="styles.css">
2. **href**: The href attribute specifies the URL or file path of the linked resource. It

defines the location of the external file that should be linked to the HTML document. For stylesheets, it specifies the path to the CSS file. For icons, it specifies the path to the image file. Example: <link rel="icon" href="favicon.ico">

3. **type**: The type attribute specifies the media type of the linked resource. It tells the browser the format or MIME type of the linked file. For stylesheets, the type attribute is typically set to "text/css". For icons, it can be set to "image/x-icon" or "image/png" depending on the file format. Example: <link rel="stylesheet" href="styles.css" type="text/css">
4. **media**: The media attribute specifies the media device or media type for which the linked resource is intended. It allows you to apply the linked resource (such as a stylesheet) only to specific media types or devices. Common values include "screen" for computer screens, "print" for print media, and "all" for all media types. Example: <link rel="stylesheet" href="styles.css" media="screen">

23. Explain the use of <marquee> tag with 4 of its element specific attributes.

The <**marquee**> tag in HTML is used to create scrolling or moving text or images on a web page. It creates a marquee effect where the content moves horizontally or vertically across the screen. The <marquee> tag has several element-specific attributes that control its behaviour. Here are four commonly used attributes of the <marquee> tag:

1. **direction**: The direction attribute specifies the direction in which the content moves. It can have one of the following values:
    - "left": The content moves from right to left.
    - "right": The content moves from left to right.
    - "up": The content moves from bottom to top.
    - "down": The content moves from top to bottom.

    Example: <marquee direction="left">Scrolling text</marquee>
2. **behaviour**: The behaviour attribute controls the scrolling behaviour of the content. It can have one of the following values:
    - "scroll": The content continuously scrolls across the screen.
    - "slide": The content slides into view and stops.
    - "alternate": The content continuously scrolls back and forth.

    Example: <marquee behaviour="scroll">Scrolling text</marquee>
3. **scrollamount**: The scrollamount attribute determines the speed of the scrolling motion. It specifies the number of pixels to be scrolled per iteration. Higher values make the content scroll faster, while lower values make it scroll slower.
   Example: <marquee scrollamount="5">Scrolling text</marquee>

4. **scrolldelay**: The scrolldelay attribute defines the time delay in milliseconds between each iteration of the scrolling motion. It controls the pause duration before the content starts scrolling again.
   Example: <marquee scrolldelay="2000">Scrolling text</marquee>

24. What is the use of <ol> ordered list? Explain with attributes.

The **<ol>** (ordered list) tag in HTML is used to create a numbered list of items. It represents an ordered sequence of items where each item is assigned a number. The <ol> tag has several attributes that control the appearance and behaviour of the ordered list. Here are some commonly used attributes of the <ol> tag:

1. **type**: The type attribute specifies the numbering style of the ordered list. It can have the following values:
   - "1": The list items are numbered with Arabic numerals (default).
   - "A": The list items are numbered with uppercase alphabetical letters.
   - "a": The list items are numbered with lowercase alphabetical letters.
   - "I": The list items are numbered with uppercase Roman numerals.
   - "i": The list items are numbered with lowercase Roman numerals.
     Example: <ol type="A">...
2. **start**: The start attribute specifies the starting number for the ordered list. It allows you to specify a different starting value for the numbering. By default, the numbering starts from 1.
   Example: <ol start="5">...
3. **reversed**: The reversed attribute reverses the numbering of the list items. When present, the list items are numbered in descending order.
   Example: <ol reversed>...
4. **compact**: The compact attribute specifies whether the spacing between the list items should be reduced. When present, it reduces the vertical spacing between the list items.
   Example: <ol compact>...

25. Explain what is the use of <script> tag? Explain its 6 element specific attribute.

The <**script**> tag in HTML is used to embed or reference external JavaScript code within an HTML document. It allows you to add dynamic functionality, perform calculations, manipulate the DOM, and handle events on a web page. The <script> tag has several element-specific attributes that control its behaviour. Here are six commonly used attributes of the <script> tag:

1. **src**: The src attribute specifies the source URL of an external JavaScript file that should be loaded and executed. It allows you to reference an external JavaScript file rather than embedding the code directly within the <script> tag.
   Example: <script src="script.js"></script>
2. **type**: The type attribute specifies the MIME type of the scripting language being used. For JavaScript, the value should be "text/javascript". However, in HTML5, the type attribute is optional, as JavaScript is the default scripting language.
   Example: <script type="text/javascript">...</script>
3. **charset**: The charset attribute specifies the character encoding of the external script file. It is typically used when the script file is served with a different character encoding than the HTML document. Common values include "UTF-8", "ISO-8859-1", and "UTF-16".
   Example: <script src="script.js" charset="UTF-8"></script>
4. **language**: The language attribute specifies the scripting language used in the <script> block. However, this attribute is deprecated in HTML5, and the type attribute should be used instead.
   Example: <script language="javascript">...</script>
5. **onload**: The onload attribute specifies a JavaScript code snippet that should be executed when the script has finished loading. It is commonly used to trigger actions or functions after the script has been successfully loaded.
   Example: <script src="script.js" onload="myFunction()"></script>
6. **onerror**: The onerror attribute specifies a JavaScript code snippet that should be executed if an error occurs while loading the script. It allows you to handle errors and take appropriate actions in case the script fails to load.
   Example: <script src="script.js" onerror="handleError()"></script>

26. Explain with attributes the <select> tag.

The <**select**> tag in HTML is used to create a dropdown list or a selection menu on a web page. It allows users to choose one or multiple options from a predefined list. The <select> tag has several element-specific attributes that control its behaviour. Here are some commonly used attributes of the <select> tag:

1. **name**: The name attribute specifies the name of the <select> element. It is used to identify the element when the form is submitted to the server. The selected option(s) are sent as part of the form data with the specified name.
   Example: <select name="fruit">...</select>
2. **size**: The size attribute specifies the number of visible options in the dropdown list without needing to open the menu. It determines the height of the dropdown box. If the size attribute is not specified, the dropdown will display a single selected option by default.
   Example: <select size="3">...</select>
3. **multiple**: The multiple attribute allows the selection of multiple options from the dropdown list. When this attribute is present, users can hold down the Ctrl (Windows) or Command (Mac) key to select multiple options. The selected options can be accessed as an array on the server.
   Example: <select multiple>...</select>
4. **disabled**: The disabled attribute disables the <select> element, making it unselectable and non-editable. Users cannot interact with the dropdown list, and its options are grayed out. This attribute is commonly used when you want to prevent user input or indicate a disabled state.
   Example: <select disabled>...</select>
5. **required**: The required attribute specifies that the <select> element must have a selected option before the form can be submitted. It is used for form validation to ensure that a value is selected. If no option is selected and the form is submitted, an error message will be displayed.
   Example: <select required>...</select>
6. **onchange**: The onchange attribute specifies a JavaScript code snippet that should be executed when the selected option(s) in the dropdown list are changed. It is commonly used to trigger actions or functions based on the selected option(s) in real-time.
   Example: <select onchange="updateSelection()">...</select>
7. **autofocus**: The autofocus attribute specifies that the dropdown list should automatically receive focus when the page loads. It is used to indicate that this element should be the initially selected or focused element when the page is loaded.
   Example: <select autofocus>...</select>

27. Explain <style> tag with its element-specific attributes.

The <**style**> tag in HTML is used to define CSS (Cascading Style Sheets) rules and styles within an HTML document. It allows you to apply styling to various elements on the page. The <style> tag does not have any specific attributes of its own, but it utilises a few important attributes to define its behaviour and specify the type of styles being used.

This element should be found only in the **<head>** element, though it appears HTML5 may loosen this restriction. Style rules directly found within a document's body generally should be set with the core style attribute for the particular element of interest.

Here are the commonly used attributes associated with the <style> tag:

1. **type**: The type attribute specifies the type of stylesheet language being used within the <style> block. The most commonly used value is "text/css", which indicates that the styles are written in CSS.
   Example: <style type="text/css">...</style>
2. **media**: The media attribute is used to specify the target media or device for which the styles defined within the <style> tag are intended. It allows you to define different styles for different devices, such as screen, print, or handheld.
   Example: <style media="screen">...</style>
3. **title**: The title attribute is used to provide a title or description for the <style> block. It helps to provide additional information about the purpose or context of the styles defined within the <style> tag.
   Example: <style title="Page Styles">...</style>
4. **lang**: Specifies the language of the styles being defined.
   Example: <style lang="en">...</style>

28. Explain <table> tag with any five element specific attributes.

The **<table>** tag is used in HTML to create a table structure on a webpage. It allows you to organize and display tabular data in rows and columns. Here are five element-specific attributes commonly used with the <table> tag:

border: The border attribute specifies the width of the border around the table and its cells. It defines the visual appearance of the table border.
Example: <table border="1">...</table>

bgcolor: The bgcolor attribute sets the background color for the entire table. It allows you to customize the background color of the table.
Example: <table bgcolor="#f2f2f2">...</table>

width: The width attribute sets the width of the table. It can be specified in pixels (px), percentage (%), or other valid CSS length units.
Example: <table width="500">...</table>

cellspacing: The cellspacing attribute defines the space between adjacent cells in the table. It helps in controlling the spacing and visual separation between cells.
Example: <table cellspacing="5">...</table>

cellpadding: The cellpadding attribute sets the space between the cell content and the cell border. It adds padding inside the cells to create spacing around the content.
Example: <table cellpadding="10">...</table>

summary: The summary attribute provides a brief description or summary of the table's content. It is useful for accessibility purposes and can be read by screen readers to provide additional context.
Example: <table summary="Sales data for the year">...</table>

29. Describe <time> element with its datetime attribute.

The <time> element in HTML is used to represent a specific date and/or time on a webpage. It helps provide semantic meaning to the date or time information and allows browsers, search engines, and assistive technologies to understand and handle it appropriately. The datetime attribute is a crucial attribute of the <time> element that specifies the date and time value associated with the content inside the element.

datetime This attribute is used to indicate the date and time of the enclosed content. The value of the attribute is a date in a special format as defined by ISO 8601. The basic date format is

YYYY-MM-DDThh:mm:ssTZD

where the following is true:

YYYY=four-digit year such as 1999
MM=two-digit month (01=January, 02=February, and so on.)
DD=two-digit day of the month (01 through 31)
hh=two-digit hour (00 to 23) (24-hour clock, not AM or PM)
mm=two-digit minute (00 through 59)
ss=two-digit second (00 through 59)
TZD=time zone designator

Example :

```
<p>Published on <time datetime="2023-06-15T09:30:00">June 15, 2023</time></p>
```

In the example above, the <**time**> element is used to display the publication date of an article. The **datetime** attribute is set to "2023-06-15T09:30:00", representing the specific date and time the article was published. The content within the <time> element, in this case, is the human-readable format of the date, which is "June 15, 2023".

30. Describe <video>element with its attributes.

The <**video**> element in HTML is used to embed videos or audio clips directly into a webpage. It provides a native way to play multimedia content without the need for third-party plugins. The <video> element supports various attributes that control the behaviour, appearance, and accessibility of the video. Here are some commonly used attributes:

src: The src attribute specifies the URL or file path of the video file to be played. It can be a relative or absolute URL.
Example: <video src="video.mp4"></video>

autoplay: The autoplay attribute automatically starts playing the video as soon as the page loads.
Example: <video src="video.mp4" autoplay></video>

controls: The controls attribute displays standard video controls (play, pause, volume, etc.) for the user to interact with the video playback.
Example: <video src="video.mp4" controls></video>

loop: The loop attribute makes the video automatically restart once it reaches the end, creating a continuous loop.
Example: <video src="video.mp4" loop></video>

poster: The poster attribute specifies an image that is displayed as a placeholder or preview before the video starts playing.
Example: <video src="video.mp4" poster="preview.jpg"></video>

width and **height**: The width and height attributes define the dimensions (in pixels) of the video player. They can be used to control the size of the video element on the webpage.
Example: <video src="video.mp4" width="640" height="360"></video>

preload: The preload attribute indicates how the video should be loaded. It can have values like "none", "metadata", or "auto". "none" prevents preloading, "metadata" loads only the video metadata, and "auto" loads the entire video file.
Example: <video src="video.mp4" preload="auto"></video>

31. Write a note on JavaScript with its features.

JavaScript is a high-level programming language that is primarily used for creating interactive and dynamic web pages. It is widely supported by modern web browsers and provides powerful capabilities for adding functionality and interactivity to websites. Here are some key features of JavaScript:

1. **Client-Side Scripting:** JavaScript is primarily executed on the client-side, meaning it runs in the user's web browser. This allows for immediate feedback and dynamic updates without needing to communicate with the server.
2. **Interactivity:** JavaScript enables interactivity by allowing users to interact with web elements, respond to events (such as button clicks , form submission), and modify the content and behaviour of a webpage in real-time.
3. **Dynamic Content:** JavaScript enables the manipulation and modification of HTML and CSS elements on the fly, allowing developers to create dynamic and personalized user experiences. It can be used to add, remove, or modify elements, change styles, and update content based on user actions or changing conditions.
4. **DOM Manipulation:** JavaScript provides access to the Document Object Model (DOM), which represents the structure and content of a webpage. This allows developers to traverse and manipulate HTML elements, modify their attributes, and dynamically create or remove elements.
5. **Event Handling:** JavaScript allows developers to define event handlers to respond to user actions such as button clicks, mouse movements, or keyboard input. This enables interactive features like form validation, dynamic menus, and interactive animations.
6. **Data Manipulation:** JavaScript provides powerful data manipulation capabilities. It supports various data types, including strings, numbers, arrays, and objects, and offers built-in functions and methods for manipulating and processing data.
7. **Browser Interaction:** JavaScript can interact with the browser's built-in objects and APIs, enabling access to browser features like cookies, local storage, geolocation, and more. This allows developers to create applications that store data, provide location-based services, or interact with external APIs.
8. **Cross-Browser Compatibility:** JavaScript is supported by all major web browsers, making it a versatile language for creating web applications that work across different platforms and devices.
9. **Extensibility:** JavaScript is highly extensible through the use of libraries, frameworks, and APIs. There are numerous JavaScript libraries and frameworks available, such as React, Angular, and jQuery, which provide additional functionality,

32. List and Explain different data types available in JavaScript.

JavaScript provides several built-in data types to store and manipulate different kinds of values. Here are the main data types in JavaScript:

Number: The number data type represents numeric values, including integers and floating-point numbers. It can be used for mathematical calculations and operations.
Example: let age = 25;

String: The string data type represents sequences of characters. It is used to store and manipulate text. Strings are enclosed in single quotes ('') or double quotes ("").
Example: let name = "John";

Boolean: The boolean data type represents logical values, either true or false. It is used for conditional statements and logical operations.
Example: let isStudent = true;

Null: The null data type represents the intentional absence of any object value. It is a special value that signifies the absence of a value.
Example: let person = null;

Undefined: The undefined data type represents the absence of a defined value. It is the default value assigned to variables that have been declared but not assigned a value.
Example: let city;

Object: The object data type is a complex data type that allows you to store collections of key-value pairs. Objects can hold properties and methods, making them a versatile data type in JavaScript.
Example: let person = { name: "John", age: 25 };

Array: The array data type represents an ordered list of values. It allows you to store multiple values in a single variable. Arrays are denoted by square brackets ([]).
Example: let numbers = [1, 2, 3, 4, 5];

Symbol: The symbol data type represents a unique identifier. Symbols are often used as property keys in objects to ensure their uniqueness.
Example: const id = Symbol("uniqueID");

BigInt: The bigint data type represents integers with arbitrary precision. It allows you to work with numbers larger than the maximum value supported by the number data type. Example: let largeNumber = 12345678901234567890;

33. Explain how to define a JavaScript function with proper example.

In JavaScript, a function is a reusable block of code that performs a specific task or calculation. Functions allow you to organize your code, improve code reusability, and create modular and maintainable applications.
Here's the syntax for defining a JavaScript function:

```
function functionName(parameter1, parameter2, ...) {
        // Code block or statements to be executed
        // Optional: Return statement to return a value
}
```

- **function**: This is the keyword used to declare a function.
- **functionName**: This is the name you choose for your function. It should be a valid identifier and follow JavaScript naming conventions.
- **parameter1**, **parameter2**,...: These are the parameters (or arguments) that the function accepts. You can have zero or more parameters, separated by commas. These parameters act as placeholders for values that will be passed into the function when it's called.
- **{}:** These curly braces enclose the function body, which contains the code or statements that will be executed when the function is called.
- **// Code block or statements**: This is where you write the actual code that defines the behaviour of the function. It can include any valid JavaScript statements, expressions, or function calls.
- **// Optional: Return statement:** If the function needs to return a value, you can use the **return** statement followed by the value to be returned. This terminates the function execution and sends the specified value back to the caller.

Here's an example that demonstrates the syntax:

```
function addNumbers(num1, num2) {
        let sum = num1 + num2;
        return sum;
}

let result = addNumbers(5, 3);
console.log(result);    // Output: 8
```

34. Explain the following with respect to JavaScript: i) function declaration ii) function constructor iii) function expression.

The Function Declaration

This is the method you have been using up to this point and one that will be used often in this book. As you will recall, you simply declare the function as follows:

```
function functionname() {
Code for function here
}
```

You can also add parameters and/or return statements as mentioned earlier in this chapter.

The Function Constructor

The function constructor creates a function object in the same way you would create a new instance of an object :

```
var functionname = new Function (arguments, code for function) ;
```

This will work like other functions, but the main drawback to this method is that it has poorer performance than the other methods (it is evaluated every time it is used rather than only being parsed once). More often than not, you will use one of the other two methods for defining functions.

The Function Expression

The function expression (also called the function operator) uses the same syntax as a function declaration. The main difference between this and a function declaration is that a function declaration creates a variable with the same name as the function name that can be used outside the function, while the function expression can only access the variable by its name within the function itself. For instance, the following code uses a function declaration and can output an alert using the function name as a variable (the value of the variable will be all of the function's code):

```
function send_alert() {
var my_num = 1;
}
window.alert(send_alert);
```

35. Explain how to use event handlers in JavaScript with example

Using an Event Handler in the Script Code

You can also use an event handler within the script code (whether using the script tags in the HTML document or using an external JavaScript file). One way to do this is to give the element an id attribute and then use the JavaScript method **document.getElementById()** to access the element. Once that is done, you can tie an event to the element.

Which can be accomplished by following code below :

```
<html><head><title> Event handling uusing JS</title></head>
<body>
<form>
<input type="button" value="Click Me!" id="say_hi" />
</form>
<script >
var hi_button = document.getElementById("say_hi");
hi_button.onclick = hi_and_bye();
function hi_and_bye() {
  window.alert(“ Hi.. “);
  window.alert(“ Bye..! ”);
}
</script>
</body>
```

The document.getElementById() method allows you to access any element in the HTML document that has an id attribute using the value of its id attribute. In order to access the button input element you have been using with an id of **say_hi,** and assign it to variable hi_button. And gives it the **onclick** event handler by adding it after the variable name and a dot (.). The function hi_and_bye (which displays the two alerts) is assigned to handle the click event on the input button. Thus, when the button is clicked, the viewer will see the two alerts!

36. Explain the addEventListener() method with an example.

The addEventListener() method in JavaScript is used to attach an event listener to an HTML element. It allows you to specify a function (known as an event handler) that will be executed when a specified event occurs on the element. Here's the syntax for using addEventListener():

```
element.addEventListener('event_type', function_name, true_or_false);
```

Let's break down the syntax:

- **element**: The HTML element to which you want to attach the event listener.
- **event**: The event type or name (e.g., "click", "keydown", "submit") that you want to listen for.
- **function**: The function that will be executed when the specified event occurs.
- **useCapture** (optional): A boolean value that indicates whether to use event capturing (**true**) or event bubbling (**false**, default). This parameter is rarely used and can be omitted in most cases.

Here's an example that demonstrates how to use **addEventListener():**

```
<button id="myButton">Click Me</button>
<script>
    function handleClick() {
        console.log("Button clicked!");
    }
    const button = document.getElementById("myButton");
    button.addEventListener("click", handleClick);
</script>
```

In the above example, we have an HTML button element with the id "myButton". We define a function called handleClick() that logs a message to the console when the button is clicked.

Using JavaScript, we select the button element using getElementById() and store it in the button variable. Then, we use addEventListener() to attach an event listener to the button. We specify the event type "click" and pass the handleClick function as the event handler.

When the button is clicked, the handleClick function will be executed, and the message "Button clicked!" will be logged to the console.

The addEventListener() method is flexible and can be used with various events (e.g., "click", "keydown", "mouseover") on different HTML elements. It allows you to respond to user interactions and create interactive web applications.

37. What is a constructor function? Explain with example.

In JavaScript, a constructor function is a special type of function that is used to create and initialize objects. It serves as a blueprint or template for creating multiple objects of the same type, often referred to as instances. Constructor functions are typically used in conjunction with the **new** keyword to create new objects.

Here's an example that illustrates how to define and use a constructor function:

```
function Person(name, age) {
  this.name = name;
  this.age = age;
}

// Creating instances using the constructor function
var person1 = new Person("John", 25);
var person2 = new Person("Alice", 30);

console.log(person1);     // Output: Person { name: 'John', age: 25 }
console.log(person2);     // Output: Person { name: 'Alice', age: 30 }
```

In the example above, we define a constructor function called **Person** with two parameters: **name** and **age**. Inside the constructor function, we use the **this** keyword to refer to the newly created object. We assign the values of the **name** and **age** parameters to the respective properties of the object.

To create new instances of the **Person** object, we use the **new** keyword followed by the constructor function name and pass the required arguments. This creates a new object with its own set of properties based on the constructor function's template.

In the example, we create two instances of the **Person** object: **person1** and **person2**. Each instance has its own set of properties defined by the constructor function. We can access and modify these properties using dot notation,
for example, **person1.name** or **person2.age**.

38. Briefly describe alert() , confirm() and prompt() methods of JavaScript.

The **alert(), confirm(),** and **prompt()** methods are commonly used in JavaScript to interact with the user through dialog boxes. Here's a brief description of each:

1. alert() : The **alert()** method displays an alert dialog box with a message and an **OK** button. It is used to provide information or notifications to the user. The function takes a single parameter, which is the message you want to display in the dialog box. After the user clicks the **OK** button, the dialog box is closed.

Syntax: alert(message);
Example : alert("Hello, World!"); // Displays an alert box with the message "Hello, World!"

2. confirm() : The **confirm()** method displays a confirmation dialog box with a message, **OK** button, and **Cancel** button. It is used to ask the user for confirmation or cancellation of an action. The function takes a single parameter, which is the message you want to display in the dialog box. When the user clicks the OK button, the function returns **true**. If the user clicks the Cancel button, it returns **false**.

Syntax: confirm(message);
Example:
```
var result = confirm("Are you sure you want to delete this item?");
if (result) {
        // Code to delete the item
} else {
        // Code to handle cancellation
}
```
In this example, the confirm() method displays a confirmation dialog box with the message "Are you sure you want to delete this item?". The result of the confirmation is stored in the result variable. If the user clicks the OK button, result will be true,

3. prompt() : The **prompt()** method displays a dialog box that prompts the user to enter some **input**. It takes **two parameters:** the first one is the message you want to display, and the second one is an optional default value for the input field. The function returns the user's input as a string if the **OK** button is clicked, or **null** if the **Cancel** button is clicked.

Syntax: prompt(message, defaultValue);
Example :
```
var name = prompt("Please enter your name:", "John Doe");
if (name !== null) {
  console.log("Hello, " + name + "!");
```

```
}
```

In this example, the prompt() method displays a dialog box with the message "Please enter your name:" and a default value of "John Doe" in the input field. The user can enter their name, and the entered value is stored in the name variable. If the user clicks the OK button, the entered name will be logged to the console as a greeting message. If the user clicks the Cancel button, the prompt() function returns null, and the greeting message is not displayed.