FULL STACK - MERN

1) what is HTML and what is its purpose?

Ans: HTML stands for Hypertext Markup Language. It is the standard markup language used to create web pages and other information that can be displayed in a web browser.

HTML is used to structure content on the web by defining elements such as headings, paragraphs, & links. It is also used to add images, videos, and other media to web pages.

The purpose of HTML is to provide a standardized way to create web content that can be interpreted by web browsers and displayed to users.

2) what is the difference between HTML & XHTML?

Ans: HTML & XHTML are both markup languages used for creating web pages. HTML is a markup language that uses a more relaxed syntax, while XHTML is a stricter & more structured version of HTML that follows the rules of XML

XHTML is essentially an extension of HTML that uses the same elements and attributes as HTML, but with stricter rules for how they are used. XHTML requires that all tags be properly closed, all elements be properly nested, and that all attributes value be enclosed in quotes. XHTML also requires that all documents be well-formed XML, which means that all opening tags must have a corresponding closing tags, and that all tags must be nested properly.

The main difference between HTML & XHTML is that XHTML is more strict & structured than HTML. XHTML is seen as a more modern and cleaner version of HTML, and is often used for creating web pages that are more accessible & compatible with different devices and browsers.

3) what are the new features introduced in HTML5?

Ans: HTML 5 introduced many new features that made it easier to create web pages and applications. Here are some of the key features.

1. new semantic elements like header, footer, article section, and nav to help structure web pages more meaningfully.
2. audio & video elements that allow playback of multimedia content without the need of third-party plugins.
3. Canvas element that allows for dynamic, scriptable rendering of 2D & 3D graphics.
4. Local storage & session storage that allow web applications to store data locally on a user's device.
5. Geolocation API that allows web applications to access a user's location.
6. Web workers & web sockets that allow for real time communication between a web application and a server.
7. Drag and drop API that allows user to drag and drop elements on a web page.
8. Improved form controls and validation.
9. Support for scalable vector graphics (SVG) and mathML.
10. offline web applications that can be accessed without an internet connection.

These new features made it easier to create rich, interactive web applications that could run on any device with a modern web browser.

4) How do you include comments in HTML?

Ans: You can include comments in HTML using the following syntax:

<!-- This is a comment -->

Anything you put between '<!--' and '-->' will be treated as a comment and will not be displayed in the browser. Comments are useful for adding notes to your code or temporarily disabling parts of your code.

5) Explain the difference between and tags?

Ans: The ' ' & '' tags are both used for grouping & formatting content on a web page, but they have different purpose and uses.

The ' ' tag is a block-level element that is used to group larger sections of content together. It is often used to create containers that can be styled with CSS or manipulated with Javascript. The ' ' tag can contain other block-level elements, such as headings, paragraphs, lists, and other ' ' elements.

The '' tag, on the other hand, is an inline-level element that is used to group smaller pieces of content together. It is often used to apply styles to specific words or phrases within a larger block of text. The '' tag can contain other inline-level elements, such as text, links, and images.

In summary, the ' ' tag is used for larger sections of content, while the '' tag is used for smaller pieces of content that need to be styled or manipulated individually.

6) what are semantic elements in HTML5 and why are they important.

Ans: Semantic elements in HTML5 are a set of tags that provide a more meaningful way to describe the content of a web page. These tags include elements like '<header>', '<footer>', '<nav>', '<article', '<section>', and others.

The main benefits of using semantic elements is that they improve the accessibility and SEO of web page. By using descriptive tags, you make it easier for search engines to understand the content & structure of your page, which can lead to higher search rankings. Additionally, semantic elements make it easier for screen readers and other assistive technologies to navigate and understand the content of your page.

Another benefits of using semantic elements is that they make your code easier to read & maintain. By using descriptive tags, you can more easily understand the structure & purpose of your code, which can make it easier to update or modify in the future.

Overall, using semantic elements is an important best practice for creating accessible, SEO-friendly, and maintainable web pages.

Q. what is the purpose of <header>, <nav>, <section>, and <footer> tags in HTML5?

Ans:- ⇒ The '<header>' tag in HTML 5 is used to define the header of the web page or a section of a webpage. The content inside the '<header>' tag typically includes things like the site logo, navigation links, and other important information.

⇒ The '<nav>' tag is used to define the section of a web page that contains the navigation links. The content inside the '<nav>' tag typically includes links to other pages on the site or other sections of the current page.

⇒ The '<section>' tag is used to define a section of the web page that groups related content together. The content inside the '<section>' tag can include headings, paragraphs, images, and other HTML elements.

⇒ The '<footer>' tag is used to define the footer of a web page or a section of a web page. The content inside the '<footer>' tag typically includes things like copyright information, contact information, & other important details.

In terms of output, these tags do not have any specific visual appearance on their own. However, they help to structure the content of a web page & make it more accessible and easier to understand for both users & search engines.

8) How do you create a hyperlink in HTML?

Ans: To create a hyperlink in HTML, you can use the '<a>' tag with the 'href' attribute. The 'href' attribute specifies the URL that the hyperlink points to.

Here is an example.

```
<a href="https://www.google.com"> click here
    to visit google.com </a>
```

In this example, the text, "click here to visit google.com" will be displayed as a hyperlink. When the user clicks on the link, they will be taken to the URL specified in the 'href' attribute.

9) what is the difference between <ol> & elements?

Ans: '<ul>' (unordered list):

The '<ul>' tag is used to create an unordered list, which represents a list of items without any specific sequence or order. The items in an unordered list are typically marked with bullet points. Each list item is represented by the '<li>' (list item) tag nested within the '<ul>' tag.

Ex:

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

output:

- Item 1
- Item 2
- Item 3

'<ol>' (ordered list):

The '<ol>' element is used to create an ordered list, which is a list of items that are numbered or ordered in some way. Each item in an ordered list is typically displayed with a number or letter.

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

output:

1. Item 1
2. Item 2
3. Item 3

So, the main difference between '<ol>' & '<ul>' is that '<ol>' creates an ordered list with numbered items, while '<ul>' creates an unordered list with bullet points or other markers.

10> How do you embed an image in HTML?

Ans: To embed an image in HTML, you can use the '<img>' tag. The <img> tag is an empty tag, which means that it doesn't have a closing tag. Instead, you use attributes to specify the source of the image, its size, and other properties.

Here's an example of how to use the '<img>' tag to embed an image.

```
<img src="image.jpg" alt="a beautiful sunset">
```

In this example, the 'src' attribute specifies the URL of the image file, and the 'alt' attribute specifies a text description of the image. The 'alt' attribute is important for accessibility, as it provides a text description of the image for users who can't see it.

You can also specify additional attributes to control the size, alignment, and other properties of the image.

Ex:

```
<img src="image.jpg" alt="A beautiful sunset" width="500" height="300" align="center">
```

In this example, the 'width' & 'height' attributes specify the size of the image in pixels, & the 'align' attribute specifies that the image should be centered horizontally on the page.

11) Explain the difference between the and tags.

Ans: The '' tag is used to indicate that the enclosed text should be considered important or strongly emphasized. By default, it is displayed in bold font.

Ex:
```
<!DOCTYPE html>
<p> This is a <strong> very important. </strong> message</p>
// output.
This is a "very important" message.
```

The tag is used to indicate that the enclosed text should be emphasized, but not necessarily strongly. By default, it is displayed in italic font.

Ex: This is an emphasized message .

output

This is an *emphasized* message.

12) How do you create a table in HTML?

Ans: To create a table in HTML, you can use the '<table>' tag, along with several other tags to define the structure & content of the table.

```
Ex: <table>
      <tr>
        <th> column 1 </th>
        <th> column 2 </th>
      </tr>
      <tr>
        <td> Row1, cell1 </td>
        <td> Row1, Cell2 </td>
      </tr>
      <tr>
        <td> Row2, cell1 </td>
        <td> Row2, cell2 </td>
      </tr>
      </table>
```

13) what is the purpose of <form> tag in html and how do you create a form?

Ans: The '<form>' tag in HTML is used to create an interactive form that allows users to input data & submit it to a server for processing. The '<form>' tag defines the beginning & end of the form, and it contains various form elements such as input fields, checkboxes, radio buttons, and buttons.

Ex:
```
<form>
  <label for="name">Name:</label>
  <input type="text" id="name"
    name="name">
  <br>
  <input type="submit" value="Submit">
</form>
```

14) what are some new input types introduced in html 5.

Ans: i) 'color': A color picker that allows users to select a color.

2) 'Date': A date picker that allows users to select a date.

3) 'datetime-local': A date and time picker that allows users to select a date and time.

4) 'email': An input field that is used for email addresses.

5) 'month': A date picker that allows users to select a month and year.

6) 'number': An input field that is used for numeric values.

7) 'range': A slider that allows users to select a value within a range.

9) 'search': An input field that is used for search queries.

a) 'tel': An input field that is used for telephone numbers.

10) 'time': A time picker that allows users to select a time.

11) 'url': An input field that is used for URLs

12) 'week': A date picker that allows users to select a week and year.

15) How to include audio and video content in HTML?

Ans: To include audio and video content in HTML, you can use the '<audio>' & '<video>' elements.

Ex: <video src="video.mp4" controls>
Your browser does nt support the video tag.
</video>

16) what is the purpose of the <iframe> tag and how is it used?

Ans: The '<iframe>' tag is used to embed another HTML document within the current HTML document. The purpose of '<iframe>' tag is to display content from another source, such as a different website or a different page on the same website, within the current document.

Ex: <iframe src="www.google.com"></iframe>

17) How do you add CSS styles to html elements?

Ans: To add CSS styles to HTML elements, you can use the 'style' attribute. The 'style' attribute allows you to add inline styles to an HTML element.

Ex:
This text is red and 24 pixels in size.

18) what is the role of the alt attribute in <img> tag?

Ans: The 'alt' attribute in the <img> tag provides alternative text for an image if the image cannot be displayed. The 'alt' attribute is used to describe the contents of the image in text form, which is useful for users who are visually impaired or who have images turned off in their web browser.

Ex: <img src="image.jpg" alt="A red apple on a white background">.

This code will display an image called "image.jpg" and provide alternative text that describes the contents of the image. If the image cannot be displayed for any reason, the alternative text will be displayed instead.

19) How do you create a numbered list with custom numbering styles in HTML?

Ans: You can use the 'list-style-type' property in CSS.

Ex:
```
<style>
  ol.custom {
    list-style-type: upper-roman;
  }
</style>
<ol class="custom">
  <li> Item 1 </li>
  <li> Item 2 </li>
  <li> Item 3 </li>
</ol>
```

20) what is the difference between <script async> and <script defer>

Ans: "async" attribute tells the browser to download the script asynchronously, which means that the script will be downloaded in the background while the rest of the page continues to load.

"defer" attribute tells the browser to download the script in the background while the rest of the page loads, but to defer execution of the script until the page has finished parsing.

The main difference between 'async' and 'defer' is when the script is executed. 'async' scripts are executed as soon as they are downloaded, even if the page is not fully loaded yet, while 'defer' scripts are executed after the page has finished loading.

21) what is responsive web design, and why is it important? ~~with example~~.

Ans: It is a technique used to create websites that can adapt to different screen sizes & devices. Its important because it ensures that your website looks good and functions properly on all devices, including desktops, laptops, tablets and smartphones.

22) How do you make a website responsive using CSS?

Ans: To make a website responsive using CSS you can use media queries to apply different styles based on the size of the device's screen. For example you can adjust the width, font size, and layout of your website based on the screen size.

Ex: @media only screen and (max-width: 600px){

```
.body {
    font-size: 16px;
}
.container {
width: 100%;
padding: 20px;
}
}
```

23) what is media query in CSS, how is it used for responsive design.

Ans: A media query is a CSS technique used to apply different style based on the device's screen size, resolution, and other characteristics It allows you to create responsive designs that adapt to different devices and screen sizes.

24) Explain the difference between a fluid layout & a fixed layout in terms of responsiveness?

Ans: Fluid layout:

It is a type of layout used in responsive web design that adjusts to the size of the device's screen. This means that width of the layout is defined in relative units, such as percentage, rather than fixed units, such as pixels.

Fixed layout:

It is a type of layout that has a fixed width and does not adjust to the screen size of the device. This means that the layout will look the same on all devices, regardless of their screen size.

Fluid layout → more flexible & adaptable to different screen size.

Fixed layout → more rigid & inconsistent

25) How do you make image responsive in CSS?

Ans: You can used the CSS property "max-width" to make an image responsive.

Ex:
```
img{
    max-width: 100%;
    height: auto;
}
```

26) what are breakpoints in responsive design, and how are they determined?

Ans: Breakpoints in responsive design refer to the specific screen widths at which the layout of a website or application changes. They are often determined based on common device sizes & resolutions.

27) How can you hide elements in specific screen sizes using CSS?

Ans: You can use media queries to hide elements in specific screen sizes.

Ex:
```
@media screen and (max-width: 768px){
    .my-element{
        display: none;
    }
}
```

Alternatively, we can use 'visibility' property.

Ex:
```
@media screen and (max-width: 768px){
    .my-element{
        visibility: hidden;
    }
}
```

28) what is the purpose of max-width property in responsive CSS?

Ans: The 'max-width' property in CSS is used to set a maximum width for an element. In responsive design, it is often used to ensure that elements do not exceed a certain width on smaller screens.

29) How to create a responsive navigation menu using CSS?

Ans: Creating a responsive navigation menu in CSS involves using media queries to adjust the layout of the menu at different screen sizes.

30) Explain the concept of mobile-first-design and how it relates to responsive CSS?

Ans: mobile-first-design is design approach that prioritizes designing for mobile device first, before designing for larger screens. This approach is based on the fact that more & more people are accessing websites & applications on mobile devices, and it is important to ensure that the experience is optimized for these users. It requires designing with a focus on flexibility & adaptability.

31) What is CSS flexbox, and what problem does it solve.

Ans: It is a layout model that allows you to easily create flexible, responsive layout. It is one-dimensional layout model that allows you to align & distribute items along a single axis, either horizontally or vertically.

Flexbox solves the problem of creating complex layouts that can adapt to different screen sizes and devices. It makes it easy to create a layout that are responsive and can change based on the screen size or orientation of the device.

32) Explain the difference between - flex container & flex items with example.

Ans: A flex container is an element that has been set to 'display: flex' or 'display: inline-flex'. It is parent element that contains one or more flex items.

A flex item is a direct child of a flex container. It is the child element that can be aligned, sized, and ordered within the flex container.

33) How do you create a flex container in CSS with example?

Ans: To create a flex container in CSS, you need to set the 'display' property of the container element to 'flex' or 'inline-flex'.

34) what are the main properties used to control the layout in flexbox?

Ans: 1. display: This property is used to create a flex container. It can be set to 'flex' or 'inline-flex'.

2. 'flex-direction': It is used to set the direction of the main axis. It can be set to 'row', 'row-reverse', 'column' or 'column-reverse'.

3. 'justify-content': This property is used to align the flex items along the main axis. It can be set to 'flex-start', 'flex-end', 'center', 'space-between', 'space-around', or 'space-evenly'.

4. 'align-items': It is used to align the items along the cross axis. It can be set to 'stretch', 'flex-start', 'flex-end', 'center', or 'baseline'.

5. 'align-content': It is used to align the flex lines along the cross axis. It can be set to 'stretch', 'flex-start', 'flex-end', 'center', 'space-between' 'space-around' and 'space evenly'.

6. 'flex wrap': This property is used to control whether the flex items should wrap if they can't fit on one line. It can be set to 'nowrap', 'wrap', or 'wrap-reverse'.

35) How do you specify the direction of flex items within the flex container.

Ans: You can specify the direction of flex items within flex containers using the 'flex-direction' property.

1. 'row': This sets the main axis to be horizontal, with flex items arranged in a row.
2. 'row-reverse': This sets the main axis to be horizontal, with flex items arranged in a row, but in reverse order.
3. 'column': This sets the main axis to be vertical with flex items arranged in a column.
4. 'column reverse": This sets the main axis to be vertical, with flex items arranged in a column, but in reverse order.

36) what is the purpose of flex grow, flex shrink and flex-basic properties?

Ans: 'flex grow':

This property specifies how much a flex item should grow relative to the other flex items in the container. The value is unitless number that represents the proportion of available space the flex item has a 'flex-grow' value of 1 and another has a value of 2, the second item will take up twice as much space as the first item. If all items have a 'flex-grow' value of 1, they will all take up the same amount of space.

flex-shrink:

This property specifies how much a flex item should shrink relative to the other flex items in the container. The value is a unitless number that represents the proportion of available space the flex item has a 'flex-shrink' value of 1 and another has a value of 2, the second item will shrink twice as much as the first item. If all items have a 'flex-shrink' value of 1, they will all shrink the same amount.

flex-basis:

This property specifies the initial size of the flex item before any remaining space is distributed. It can be fixed size. (e.g. 100px), a percentage (e.g. 50%), or the keyword 'auto', which means the item will be sized based on its content.

37) How do you align flex items horizontally and vertically within a flex container.

Ans: You can align flex items both horizontally and vertically within a flex container using the 'justify-content' & 'align-items' properties, respectively.

38) Explain the difference between justify content & align items properties in flex box.

Ans: 'Justify content'

It aligns items along the main axis of the container, which is determined by the 'flex-direction' property. This property is used to distribute extra space left over when the flex items don't use all available space on the main axis.

"Align-items":

Align items along the cross axis of the container, which is perpendicular to the main axis. This property is used to align items on the cross axis when the items don't use all available space.

So, both properties work together to center the items both horizontally and vertically within the container.

39) How can you control the order of flex items using CSS flexbox?

Ans: You can control the order of flex items using the 'order' property in CSS flexbox. This property specifies the order in which the flex items appear in the container.

By changing the order property, you can easily change the order in which the items appear in the container without changing the HTML structure.

40) What are flexbox breakpoints, & how can they be used for responsible design?

Ans: Flexbox breakpoints are specific points in the screen size range where you want to change the layout of your page. You can use these breakpoints to create a responsive design that adjusts to different screen sizes.

By using flexbox breakpoints, you can create a responsive design that adjusts to different screen sizes, ensuring that your webpage looks great on all devices.

41) What are HTML attributes?

Ans: HTML attributes provides additional information about a HTML element, such as an image, a hyperlink, or a form input. HTML attributes can be used to add more functionality & meaning to HTML elements, & there are many different attributes that can use depending on the type of element you are working with.

42) Explain the difference between Global attribute & element specific attribute in html?

Ans: Global attribute are attributes that can be used on any HTML element, while element-specific attributes are attributes that are specific to certain HTML elements.

In general, global attributes can be used on any HTML element, while element-specific attributes are specific to certain HTML elements and are used to provide additional functionality or information.

13) How do you add attributes to an html element?

Ans: You can add attributes to an HTML element by including them in the opening of tag of the element.

Ex: <img src="image.jpg" alt="A beautiful sunset" width="500" height="300">

14) what is the purpose of the id attribute in html, & how is it unique?

Ans: The 'id' attribute in HTML is used to give an element a unique identifier. This identifier can be used to target the element with CSS or Javascript, or to create links to specific parts of the page.

It's important to note that the 'id' attribute must be unique within the HTML document. If you use the same 'id' value for multiple elements, it can cause problems with CSS & Javascript.

45) what is the difference between class attribute & id attribute?

Ans: The 'class' attribute is used to group elements together based on a common characteristic.

The 'id' attribute is used to give an element a unique identifier that can be used to target it with CSS or Javascript.

46) Explain the role of href attribute in html, particularly in the context of links & anchors?

Ans: The 'href' attribute in html is used to specify the URL of the page that the link goes to. It's used in combination with the 'a' element to create links between pages.

Ex:

```
<a href="www.google.com">Go to Google</a>
```

It's important to note that the 'href' attribute must always be included in anchor tags. If the 'href' attribute is missing, the link won't work.

47) How do you add alternative text to an image using the alt attribute?

Ans: The 'alt' attribute in HTML is used to provide alternative text for an image. This text is displayed if the image cannot be loaded, or if the user is using a screen reader to access the web page.

It's important to include alternative text for images to make your web content more accessible to users with disabilities. The alternative text should be descriptive & provide the same information as the image. This helps users with visual impairments to understand the content of the page.

48) What is the purpose of the target attribute in html links, and what are its possible values.

Ans: The 'target' attribute in html is used to specify where to open the linked document. It's used in combination with the 'a' element to create links between pages.

The possible values of the 'target' attribute are:

1. "_self": opens the linked document in the same window/tab as the current page. This the default value if the 'target' attribute is not specified.

2. '_blank': opens the linked document in a new window/tab.
3. '_parent': opens the linked document in the parent frame of the current frame.
4. '_top': opens the linked document in the full body of the window, replacing all frames.

It's important to note that the 'target' attribute should be used sparingly and only when necessary - opening links in new windows/tabs can be disruptive to the user's browsing experience, so it's important to use this feature judiciously.

Q7. How do you use the src attribute to embed an external resource, such as an image or video in html?

Ans. The 'src' attribute in HTML is used to specify the URL of an external resource, such as an image or video, that should be embedded in a web page.

It's important to note that the external resource must be publicly accessible in order for it to be embedded in a web page. If the resource is located on a private server or behind a login screen, it may not be possible to embed it in a web page.

50) what is the purpose of disabled attribute, and how is it used in html form elements?

Ans: The 'disabled' attribute in HTML is used to disable a form element, such as button, input field, or select dropdown, so that it cannot be interacted with by the user.

Ex:
```
<label for="name">Name:</label>
<input type="text" id="name"
    name="name" disabled>
```

It's important to note that disabled form elements are not submitted with the form data, so if you want to include a disabled input field or button in the form data, you'll need to remove the 'disabled' attribute before submitting the form.

51) Is there any relation between java and javascript?

Ans: Despite the similar names, java and javascript are two different programming languages with different syntax, semantics and use cases.

Java is a general-purpose programming language that is used for developing desktop, web and mobile applications, as well as server-side applications & enterprise software. Java code is compiled into bytecode that can run on any platform that has a Java Virtual machine (JVM) installed.

Javascript, on the other hand, is a scripting language that is used primarily for creating dynamic and interactive web pages. Javascript code is executed by web browsers & can be used to manipulate the document object model (DOM), handle user events, & make asynchronous requests to web servers.

While Java & Javascript share some similarities in syntax & programming concepts, they are fundamentally different languages that are used for different purposes.

52) Is Javascript a compiled or interpreted language?

Ans: Javascript is an interpreted language, which means that the code is executed directly by the browser or runtime environment without being compiled into machine code beforehand.

53) Is Javascript a case sensitive language?

Ans: Yes, Javascript is a case-sensitive language, which means that variables, functions, and other identifiers must be spelled and capitalized the same way throughout the code.

Ex: Javascript

```
var greeting = "Hello, World!";
var Greeting = "Hi, there!";
alert(greeting); // displays "Hello, World!"
alert(Greeting); // displays "Hi, there!"
```

In this example, we have two variables named 'greeting' & 'Greeting', which have different capitalizations. When we display the values of these variables using the 'alert()' function, we get different results because the variables are spelled & capitalized differently.

To avoid errors in your code, it's important to be consistent in your use of capitalization & spelling when naming variables, functions, & other identifiers.

54) what is node.JS?

Ans: Node.JS is an open-source, cross-platform, server side Javascript runtime environment that allows developers to build scalable, high performance applications.

Node.js uses an event-driven, non-blocking I/O model that makes it lightweight & efficient, making it ideal for building real-time, data intensive applications.

Node.js can be used to build a wide variety of applications, including web server, command-line tools, desktop applications, and more.

55) what is the difference between let & var?

Ans: In Javascript let & var is used to declare variables, but they have some important differences.

'var' declares a variable globally, or locally to an entire function, regardless of block scope.

'let' declares a variable with block scope, which means that its only accessible within the block in which it was declared

Ex:
```
function example() {
    var x = 1;
    let y = 2;
    if (true) {
    var x = 3;
    let y = 4;
    }
  console.log(a);   // output 3
  console.log(y);   // output 2
  }
  example();
```

56) what are the difference between undeclared and undefined variable?

Ans: In Javascript "undeclared variable is a variable that has not been declared in the current scope. If you try to use an undeclared variable, you'll get a 'Reference Error'.

Ex:
```
function example () {
    console.log(x);        // throws a reference error
}
example ();
```

An 'undefined' variable, on the other hand is a variable that has been declared but has not been assigned a value. If you try to use an undefined variable, you'll get the value 'undefined'.

Ex:
```
function example () {
    var x;
    console.log(x);        // output undefined.
}
example ();
```

57) what is Hoisting?

Ans: Hoisting is a Javascript mechanism that moves variable and function declarations to the top of their respective scopes during compilation. This means that you can use a variable or function before it has been declared, without getting an error. However, its important to note that only the declarations are hoisted, not the assignments.

Ex: function example() {
console.log(x); // outputs undefined
var x = 1;
console.log(x); // outputs 1
}
example();

58) what is scope in javascript?

Ans: Scope refers to the accessibility of variables, functions, & objects in a particular part of your code.
There are 2 types of scope.
* Global scope.
* Local scope.

=> Global scope refer to variables, functions, & objects that are accessible throughout your entire code. //

Ex: const name = 'Aruind';
function greet() {
console.log('Hello, ${name}!');
}
greet(); // output. "Hello, Aruind!"

=> local scope refers to variable, functions, & objects that are only accessible within a specific function or block of code.

Ex: function greet() {
const name = 'Aruind';
console.log('Hello, ${name}!');
}
greet(); // output: Hello, Aruind!"
console.log(name); // output: uncaught ReferenceError: name is not defined

Understanding scope is important in Javascript because it helps you to avoid naming conflicts & write more efficient & organized code.

59) what are reserved words? Can I use reserved words as identifiers.

Ans: Reserved words are words that are part of the Javascript language & have a special meaning. These words cannot be used as identifiers (variable names, function names, etc.) in your code. Attempting to use a reserved word as an identifier will result in a syntax error.

Here's an example of trying to use a reserved word as a variable name:

Ex: var let = 10; // throws syntax error.

60) why do you need strict mode? How do you declare strict mode in Javascript with example?

Ans: Strict mode is a way to opt-in to a stricter & more secure version of Javascript. It helps prevent coding mistakes & makes it easier to write "secure" code.

Some benefits.

* makes it easier to write secure Javascript code by throwing errors for unsafe actions.
* It helps to prevent accidental global variable declarations by throwing an error when you try to create a global variable.
* It makes it easier to optimize your code by disabling certain runtime behaviours that are not used in modern Javascript.

61) what are global variables ?

Ans: Global variables in Javascript are variables that are defined outside of any function or block. These variables can be accessed from anywhere in your code, including inside functions.

```
Ex: var globalVar = 'Hello, World!';
    function sayHello() {
    console.log(globalVar);
    }
    sayHello();          // output "Hello, world!"
```

62) what is NaN property?

Ans: 'NaN' is a special value in Javascript that represents "Not a Number". It's a value that is returned when a mathematical operation fails or when a value is not a valid number.

```
Ex: var result = 10/'hello';
    console.log(result);    //output NaN.
```

You can use the 'isNaN()' function to check whether a value is 'NaN'.

```
Ex: var result = 10/'hello';
    if (isNaN(result)) {
    console.log('The result is not a valid
    number');
    } else {
    console.log('The result is' + result);
    }.
```

63) what are the problems with global variables?

Ans: Global variables can cause a number of problems, including naming collisions, unexpected changes, and difficulty with debugging.

Ex:
```
let count = 0;
function increment(){
 count++;
}
function decrement(){
 count--;
}
increment();
console.log(count);          //output: 1
decrement();
console.log(count);          //output: 0.
```

64) what is the purpose of delete operator?

Ans: The 'delete' operator in Javascript is used to remove a property from an object. It can also be used to remove an element from an array.

Ex:
```
var person = {
        name = 'Arvind',
        age = '25',
        city = 'Bangalore',
        };
      delete person.city;
     console.log(person);
                //outputs{name:'Arvind',age:25}
```

This is an example to delete or remove a property from an object.

This is an example of using 'delete' to remove an element from an array.

```
var fruits = ['apple', 'banana', 'orange'];
delete fruits[1];
console.log(fruits);
//outputs. ['apple', undefined, 'orange']
```

It's important to note that the 'delete' operator only removes the property or element from the object or array. It does not change the length of an array. Also it's not recommended to use 'delete' to remove array and cause unexpected behavior. Instead, you should use array methods like 'splice()' to remove elements from an array.

65) what is the difference between null and undefined?

Ans: In Javascript, 'null' & 'undefined' are both used to represent the absence of a value, but they are used in different ways.

'undefined' is a value that is automatically assigned to a variable that has not been initialized or to a parameter that has not been passed a value. It can also be explicitly assigned to a variable or property to indicate the absence of a value.

Ex: var x;
console.log(x); // outputs undefined.
function printMessage(message) {
console.log(message);
}
printMessage(); // outputs undefined.

'null', on the other hand, is a value that represents the intentional absence of any object value. It is often used to indicate that a variable or property should have no value.

Ex: var y = null;
console.log(y); // outputs null.

var person = {
name: 'Arvind',
age: null
};
console.log(person.age); // outputs null.

It's important to note that 'undefined' & 'null' are both falsy values in JavaScript, which means they evaluate to 'false' in a boolean context. However, they are not equal to each other or to any other value except for themselves ('undefined' is not equal to 'null', '0', 'false', or '', or any other value).

66) what is a bitwise operators available in Javascript

Ans: It provides several bitwise operators that allow you to manipulate the binary representations of numbers.

1) '&' (AND):- returns a one in each bit position for which the corresponding bits of both operands are ones.

2) '|' (OR):- returns a one in each bit position for which the corresponding bits of either or both operands are ones.

3) '^' (XOR):- returns a one in each bit position for which the corresponding bits of either but not both operands are ones.

4) '~' (NOT):- inverts the bits of its operand.

5) '<<' (LEFT SHIFT): shifts the first operand to the left by the no of positions specified by the second operand.
The left most bits are filled with zero.

6) '>>' (RIGHT SHIFT): Shifts the bits of the first operand to the right by the number of positions specified by the second operand.
The right most bits are filled with zeros.

7) '>>>' (Zero-Fill right shift):
shifts the bits of the first operand to the right by the number of positions specified by the second operand. The rightmost bits are filled with zero, and the leftmost bit is set to zero.

67) Can I redeclare let & const variables?

Ans: No, you cannot redeclare let & const variables in Javascript. If you try to redeclare a let or const variables in the same scope, you will get an error.

Ex: let a = 5;
let a = 10; // Error: Identifier 'a' has already been declared.

const b = 20;
const b = 30; // Error: Identifier 'b' has already been declared.

In Javascript, let & const variables are block-scoped, which means they are only accessible within the block they are declared in. Once a variable is declared with let or const, it cannot be redeclared in the same block. This is different from var variables, which can be redeclared within the same scope.

68) Does const variables makes the value immutable?

Ans: While const variables in Javascript are not completely immutable, they do prevent the variable from being reassigned to a different value.

Ex: const a = 5;
a = 10; // Error: Assignment to constant variable.

69) what is ES6? List down some of the features of ES6.

Ans: ES6 (also known as ECMA Script 2015) is a version of the ECMA Script standard, which is the specification that Javascript follows. ES6 introduced many new features and improvements to the language, making it more powerful & expressive.

features introduced in ES6:

1). 'let' & 'const' declarations for block-scoped variables:

```
let a=5;        // block scoped variable
const b=10;     // block scoped constant variable.
```

2) arrow function for more concise function syntax:

```
const add = (x,y) => x+y;
```

3) Template literals for more readable string interpolation:

```
const name ='Arunod';
console.log(`Hello ${name}!`);
```

4) Default function parameters for more concise function definitions:

```
function greet.(name ='World'){
console.log(`Hello ${name}!`);
}
```

5) Rest parameters for more flexible function parameters;

```
function sum(...numbers){
return numbers.reduce((total,num)=>
total+num,0);
}
```

6) Destructuring for more concise and expressive variable assignments:

```
const person = {name: 'Arwind', age: 25};
const {name, age} = person;
console.log(name, age);
```

7) Classes for more object-oriented programming in Javascript:

```
class Animal {
constructor(name) {
this.name = name;
}
speak() {
conole.log(`${this.name} makes a noise.`);
}
}
```

8) Modules for more organized and reusable code:

```
// math.js
export function add(x,y){
return x+y;
}

// app.js
import {add} from './math.js';
consolk.log(add(5,10));
```

70) what are the possible ways to create objects in Javascript?

Ans: 1. object literal notation;

```
const person = {
    name : 'Aruind',
    age : 30,
    greet : function (){
console.log ('Hello, my name is ${this.name}
and i'm ${this.age} years old.');
    }
};
```

2. Constructor function:

```
function Person (name, age){
    this.name = name;
    this.age = age;
    this.greet = function (){
console.log ('Hello, my name is ${this.name}
and I'm ${this.age} years old.');
    }
}
const person = new Person ('John', 30);
```

3. object.create method:

```
const personPrototype = {
 greet : function (){
console.log ('Hello, my name is ${this.name}
and I'm ${this.age} years old.');
 }
};
const person =
object.create (personPrototype);
person.name = 'Aruind';
person.age = 25;
```

4. ES6 classes:

```
class Person {
  constructor(name, age){
    this.name = name;
    this.age = age;
  }
  greet(){
    console.log('Hello, my name is ${this.name}
    and I'm ${this.age} years old.');
  }
}

const person = new person('Arvind', 25);
```

7) what is the difference between slice & splice

Ans: 'slice()' & 'splice' are both methods in Javascript used to manipulate arrays. The main difference between them is that 'slice()' returns a new array containing a portion of the original array, while 'splice()' modifies the original array by adding or removing elements.

Ex for slice();

```
const arr = ["apple", "banana", "cherry",
  "date", "elderberry"];
const slicedArr = arr.slice(1,4);

console.log(arr);          // ["apple", "banana", "cherry",
                              "date", "elderberry"];

console.log(slicedArr);    // ["banana", "cherry",
                              "date"];
```

Ex for splice();

```
const arr = ["apple", "banana", "cherry", "date",
    "elderberry"];
const splicedArr = arr.splice(1, 3, "grapefruit",
    "kiwi");
console.log(arr);          // ["apple", "grapefruit", "kiwi",
                           "elderberry"]
console.log(splicedArr);   // ["banana", "cherry",
                           "date"].
```

72) what is the difference between.

a. == and === operator.

Ans: "==" performs type coercion, while "===" does not.

Ex: console.log(5=='5'); //output: true.

"==" operator compares the number to string.

Ex: console.log(5==='5'); //output: false

b. = and == operator.

"=" operator is used for assignment.

"==" operator is used for comparison.

The "=" operator is used to assign a value to a variable.

Ex: var x=5;

The "==" operator is used to compare two values for equality.

Ex:

```
console.log(5=='5');      //output: true
```

c. %= and = operator.

"%=" operator is used for assignment with modulus i.e remainder of a division operation to a variable.

Ex: var x = 10.
x %= 3
console.log(x); //output: 1

The "=" operator is used to assign a value to a variable.

Ex: var x = 5;

7) what is higher order function?

Ans: It is a function that takes another function as an argument, or returns a function as its result.

one of the most common example of a higher-order function in Javascript is the 'Array.prototype.map()' method.

Ex: const numbers = [1, 2, 3, 4, 5];
const doubled = numbers.map(function(num)
{
return num * 2;
});
console.log(doubled); //output: [2, 4, 6, 8, 10].

74) what is currying function?

Ans: currying is the process of transforming a function that takes multiple arguments into a function that takes those arguments one at a time.

In Javascript currying can be achieved using closures.

Ex:
```
function add(x){
return function(y){
return x+y;
}
}
const add5 = add(5);
console.log(add5(3));    // output: 8
console.log(add5(10));   // output: 15.
```

75) what are arrow functions?

Ans: arrow functions are a concise way to write function in Javascript. They were introduced in ES6 & provide a shorthand syntax for writing function expressions.

Ex:
```
const add = (x, y) => {
    return x+y;
}
console.log(add(5,3));   // output: 8.
```

76) what is spread operator?

Ans: The spread operator is a syntax for "spreading" the elements of an iterable (like an array or a string) into a new array or function call.

Ex:

```
const arr1 = [1, 2, 3];
const arr2 = [4, 5, 6];
const arr3 = [...arr1, ...arr2];

console.log(arr3);    // output: [1, 2, 3, 4, 5, 6]
```

77) what is rest parameter?

Ans: The rest parameter is a syntax for representing an ~~undefined~~ indefinite number of arguments as an array.

Ex:

```
function sum(...args){
    return args.reduce((acc, val) => acc + val);
}
console.log(sum(1,2,3));      // output: 6
console.log(sum(1,2,3,4,5));  // output: 15.
```

78) what happens if you do not use rest parameter as a last argument?

Ans: If you don't use the rest parameter as the last argument in a function, you'll get a syntax error.

Ex:

```
function foo(a, b, c, ...d, e){
    console.log(a);   // output: 1
    console.log(b);   // output: 2
    console.log(c);   // output: 3
    console.log(d);   // output: [4, 5, 6]
    console.log(e);   // output: 7
}

foo(1, 2, 3, 4, 5, 6, 7);   // output: Uncaught SyntaxError: Rest parameter must be last formal parameter.
```

79) what are regular expression patterns?

Ans: Regular Expression patterns are used in Javascript to match and manipulate strings. They are a powerful tool that allows you to search for specific patterns of characters within a string.

Ex: const patter = /dog/;
const str = " I have a dog named Max";
if (patter.test(str)){
console.log("The string contains the word 'dog'");
} else {
console.log("The string does not contain the word 'dog'");
}

80) what is regular expression?

Ans: A regular expression, or regex, is a pattern used to match character combinations in strings.

In Javascript, you can create a regular expression by enclosing a pattern in forward slashes (/pattern/).

81) How do you search a string for a pattern in Javascript?

Ans: You can search for a pattern within a string using a regular expression. A regular expression is a sequence of characters that forms a search pattern.

Ex:
```
const str = "The quick brown fox jumps over the lazy dog";
const pattern = /fox/i;
const result = str.search(pattern);
console.log(result);          // 16.
```

82) What is the purpose of switch case?

Ans: 'switch' statement is used to execute different actions based on different conditions. It's a more concise way of writing multiple 'if...else' statements.

Ex:
```
const day = new Date().getDay();
let message;
switch (day) {
    case 0:
      message = "Today is Sunday";
    break;
    case 1:
      message = "Today is Monday";
      break;
    case 2:
      message = "Today is Tuesday";
      break;
    default:
      message = "invalid day";
  }
console.log(message);
```

83) what are the conventions to be followed for the usage of switch case?

Ans: * Each 'case' should be indented one level from the 'switch' statement.
* Each 'case' should end with a 'break' statement to prevent "falling through" to the next case.
* The 'default' case should be included to handle any unexpected values.
* Use strict equality ('===') to compare values in the 'case' statements, to avoid unexpected type coercion.
* Use a 'switch' statement when you have a limited number of possible options to choose from.

84) what are primitive datatypes?

Ans: i) "undefined": represents a value that is not yet defined, or a variable that has not been assigned a value.

Ex: let x;
console.log(x); // undefined.

ii) 'null': represents a deliberate non-value and is often used to indicate that a variable has no value.

Ex: let y = null;
console.log(y); // null.

iii) 'boolean': represents a logical value of 'true' or 'false'.

let a = true;
let b = false;

iv) 'number': represents a numeric value.

Ex: let a = 23
let b = 3.14;

v) 'string': represents a sequence of characters.

Ex: let e = "hello";
let f = 'world';

vi) 'symbol': represents a unique identifier.

Ex: const g = Symbol('description');

85) what are the different ways to access object properties?

Ans: 1. Dot notation: Use a dot ('.') to access a property of an object.

```
Ex: const person = {
        first name : "Aswind",
        last name : "Sundaraj",
        age : 25
        };
    console.log(person.age);        //25.
```

2. Bracket notation: Use a square bracket ('[]') to access a property of an object.

```
Ex: const person ={
        first name : "Aswind",
        Last name : "Sundaraj",
        age : 25
        };
console.log(person["firstname"]);        //Aswind
const propertyName = "firstName";
```

86) what are the function parameter rules?

Ans: 1. Function parameters are optional:
You can define a function with or without parameters.

2. Function parameters can have default value:
You can set default values for function parameter using the "=" operator.

3. Function Parameters can be rest parameters:
You can define a rest parameter by prefixing the parameter name with '...'.

4. Function parameters can be destructured

87) Different ways which create infinite loops?

Ans: 1. Using a while loop with a condition that's always true.

2. Using a for loop with a condition that's always true

3. Using a do-while loop with a condition that's always true.

4. Using recursion without a base case:

5. using a timer with a callback function that calls itself:

88) what are template literals?

Ans: It is a new feature in Javascript that allow you to embed expressions inside string literals. They are enclosed by backtick (`) characters instead of single quotes or double quotes.

Ex: const name = 'Arwind';
console.log(`Hello, ${name}`);

89) what are default values in destructuring assignment in Javascript?

Ans: Destructuring assignment is a feature in Javascript that allows you to extract values from arrays or objects and assign them to variables. Default values can be specified for variables in case the value being destructured is undefined or null.

Ex:
const [a=1, b=2, c=3] = [undefined, null, 0];
console.log(a, b, c); //output: null 0.

Overall, default values in destructuring assignment allow you to provide feedback values for variables in case the value being destructured is undefined or null. This can be useful for handling optional arguments or values in your code.

90) How do you swap variables in destructuring assignment?

Ans: In Javascript, you can use destructuring assignment to swap the values of two variables without using a temporary variable.

```
Ex: let a = 1;
    let b = 2;
    [a, b] = [b, a];
console.log(a);      //output: 2
console.log(b);      //output: 1
```

91) Is that possible to use expressions in Switch cases?

Ans: Yes, it's possible to use expressions in 'switch' cases in Javascript.

```
Ex: let fruit = 'apple';
switch(true){
case fruit === 'banana';
console.log('This is a banana');
break;
case fruit === 'apple';
console.log('This is a apple');
case fruit === 'orange':
console.log('This is an orange.');
break;
default;
console.log('I donot recognize this
        fruit');
}
```

92) what is the difference between for... of and for...in statements?

Ans: 'for... of' is used to iterate over the values of an iterable object, such as an array or a string.

```
Ex: let arr = [1, 2, 3, 4, 5];
    for (let val of arr) {
      console.log (val);
    }
```

// output
1
2
3
4
5

'for... in' is used to iterate over the properties of an object.

```
Ex: let obj = {a:1, b:2, c:3};
    for (let prop in obj) {
      console.log (prop + ': ' + obj [prop]);
    }
```

// output
a:1
b:2
c:3

93) What are the difference between arguments object and para rest parameter?

Ans: The 'arguments' object is an array-like object that contains all the arguments passed to a function.

```
Ex: function sum () {
      let total = 0;
      for (let i=0; i<arguments.length; i++) {
        total += arguments [i];
      }
      return total;
    }
console.log (sum (1, 2, 3, 4, 5));   //15
```

The rest parameter ('...') is a new feature introduced in ES6 that allows you to pass a variable number of arguments to a function as an array.

Ex:
```
function sum(...args){
  let total = 0;
  for(let arg of args){
    total += arg;
  }
  return total;
}
console.log(sum(1,2,3,4,5));   //15
```

94) what are the difference between spread operator & rest parameter?

Ans: The spread operator is used to expand an array or an object into individual elements.

Ex:
```
const arr1 = [1,2,3];
const arr2 = [4,5,6];
const arr3 = [...arr1, ...arr2];
console.log(arr3);   //[1,2,3,4,5,6]
```

The rest parameter is used to represent an indefinite number of arguments as an array

Ex:
```
function sum(...args){
  let total = 0;
  for(let arg of args){
    total += arg;
  }
  return total;
}
console.log(sum(1,2,3,4,5));   //15
```

Q5) Explain all the array methods, what are the output & whether the method modified the original array.

Ans: 1. 'concat()': The 'concat()' method is used to merge two or more arrays into a new array.

It doesn't modify the original array.

Ex:
```
const arr1 = [1,2,3];
const arr2 = [4,5,6];
const arr3 = arr1.concate(arr2);
console.log(arr3);    // [1,2,3,4,5,6]
```

2. 'push()': It is used to add one or more elements to the end of an array.

It modifies the original array & returns the new length of the array.

Ex:
```
const arr = [1,2,3];
Const length = arr.push(4,5);

console.log(arr);    //[1,2,3,4,5]
console.log(length);    //5
```

3. 'pop()': It is used to remove the last element from an array.
It modifies the original array and returns the removed element.

Ex: const arr = [1, 2, 3];
const lastElement = arr.pop();

console.log(arr); // [1, 2]
console.log(lastElement); // 3

4. 'shift()': It is used to remove the first element from an array.
It modifies the original array & returns the removed element.

Ex: const arr = [1, 2, 3];
const firstElement = arr.shift();

console.log(arr); // [2, 3]
console.log(firstElement); // 1

5. 'unshift()': It is used to add one or more elements to the beginning of an array.
It modifies the original array & returns the new length of an array.

Ex: const arr = [1, 2, 3];
const length = arr.unshift(4, 5);

console.log(arr); // [4, 5, 1, 2, 3]
console.log(length); // 5.