Html Interview question website https://www.interviewbit.com/html-interview-questions/

Java interview question website <https://www.interviewbit.com/java-interview-questions/>

<https://acchou.github.io/html-css-cheat-sheet/html-css-cheat-sheet.html>

https://www.interviewbit.com/java-cheat-sheet/

Following are imp links for botstrap

<link rel=*"stylesheet"*

href=*"https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"*

integrity=*"sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"*

crossorigin=*"anonymous"*>

<link

href=*"https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"*

rel=*"stylesheet"*>

<script

src=*"https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.6/dist/umd/popper.min.js"*></script>

<script

src=*"https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js"*></script>

1 Html मे starting को <DOCTYPE html > add करणेका

2 <head > add करणे का

3 head tag मे meta tag and link tag(मे . css file add करते )

4 </head>close करको <body> tag open करणे का उसमे <main> tag लागणे का

𝗙𝗼𝘂𝗻𝗱𝗮𝘁𝗶𝗼𝗻𝘀 𝗼𝗳 𝗪𝗲𝗯 𝗗𝗲𝘃𝗲𝗹𝗼𝗽𝗺𝗲𝗻𝘁

Basic HTML

- Form

- Import

- Elements

- Attributes

- Semantics

- Multimedia

- Block element

Sure, here's a brief explanation of each of those terms with an example:

1.**Form** HTML forms are used to collect user input. They typically contain input fields like text fields, checkboxes, radio buttons, submit buttons, etc. Users can input data into these fields and submit it to a server for processing.

**Example:**

<html>

<form action="/submit\_form.php" method="post">

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

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

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

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

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

</form>

2.**Import:** HTML imports are used to include external resources, such as CSS stylesheets or JavaScript files, into an HTML document.

Example:

*<html>*

*<link rel="stylesheet" type="text/css" href="styles.css">*

*<script src="script.js"></script>*

**3.Elements**: HTML elements are the building blocks of HTML pages. They define the structure and content of a webpage. Each element has a specific purpose and may contain other elements, attributes, or text.

**Example:**

<html>

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

<h1>This is a heading element.</h1>

<ul>

<li>Item 1</li>

<li>Item 2</li>

</ul>

**4.Attributes:** HTML attributes provide additional information about HTML elements. They are used to modify the behavior or appearance of an element.

Example:

<html>

<img src="image.jpg" alt="A beautiful image">

<a href="https://www.example.com" target="\_blank">Visit Example</a>

5.**Semantics**: HTML semantics refers to the meaning and purpose of HTML elements. Semantically correct HTML helps improve accessibility, search engine optimization, and maintainability of web pages.

Example:

<html>

<header>

<h1>Website Title</h1>

<nav>

<ul>

<li><a href="/">Home</a></li>

<li><a href="/about">About</a></li>

<li><a href="/contact">Contact</a></li>

</ul>

</nav>

</header>

**6.Multimedia**: HTML provides elements to embed multimedia content, such as images, videos, and audio, into web pages.

Example:

<html>

<img src="image.jpg" alt="A beautiful image">

<video width="320" height="240" controls>

<source src="video.mp4" type="video/mp4">

Your browser does not support the video tag.

</video>

<audio controls>

<source src="audio.mp3" type="audio/mpeg">

Your browser does not support the audio element.

</audio>

**7.Block Element:** Block-level elements are HTML elements that typically start on a new line and stretch out to the left and right as far as they can. They create "blocks" of content on a webpage.

**Example:**

<html>

<div>

<p>This is a block-level paragraph element.</p>

<p>This is another block-level paragraph element.</p>

</div>

In this example, the `<div>` element is a block-level element, and it contains two `<p>` elements, also block-level. Each `<p>` element starts on a new line and takes up the full width of its containing `<div>`.

**CSS**

Certainly! Here's a set of interview Q&A for basic CSS concepts with examples:

\*\*1. Box Model:\*\*

Q: What is the CSS Box Model and its components?

A: The CSS Box Model describes the layout and rendering of elements in HTML. It consists of four parts: Content, Padding, Border, and Margin.

Example:

```html

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 200px;

height: 100px;

padding: 20px;

border: 2px solid black;

margin: 10px;

}

</style>

</head>

<body>

<div>This is a box with content, padding, border, and margin.</div>

</body>

</html>

```

\*\*2. Pseudo Classes:\*\*

Q: Explain what Pseudo Classes are in CSS and provide examples of some commonly used ones.

A: Pseudo Classes are used to define the special state of an element. Some common examples include :hover, :active, :focus.

Example:

```html

<!DOCTYPE html>

<html>

<head>

<style>

a:hover {

color: red;

}

input:focus {

background-color: lightblue;

}

</style>

</head>

<body>

<a href="#">Hover over me</a>

<input type="text" placeholder="Click here">

</body>

</html>

```

\*\*3. Class and other selectors:\*\*

Q: Differentiate between class selectors and other CSS selectors.

A: Class selectors target elements with a specific class attribute, while other selectors like tag selectors target specific HTML tags and ID selectors target elements with a specific ID attribute.

Example:

```html

<!DOCTYPE html>

<html>

<head>

<style>

/\* Class Selector \*/

.highlight {

color: blue;

}

/\* Tag Selector \*/

p {

font-size: 16px;

}

/\* ID Selector \*/

#unique {

background-color: yellow;

}

</style>

</head>

<body>

<p class="highlight">This paragraph has a class applied to it.</p>

<p>This paragraph is styled using tag selector.</p>

<div id="unique">This div has an ID applied to it.</div>

</body>

</html>

```

\*\*4. CSS types - Flex, Grid, normal:\*\*

Q: Explain the differences between Flexbox, CSS Grid, and normal flow layout.

A: Flexbox is a one-dimensional layout method for laying out items in rows or columns, CSS Grid is a two-dimensional layout method for laying out items in rows and columns, and normal flow layout is the default layout method where elements are displayed according to the HTML structure.

Example:

```html

<!DOCTYPE html>

<html>

<head>

<style>

/\* Flexbox Example \*/

.flex-container {

display: flex;

justify-content: space-between;

}

/\* Grid Example \*/

.grid-container {

display: grid;

grid-template-columns: 1fr 1fr 1fr;

gap: 10px;

}

</style>

</head>

<body>

<!-- Flexbox Example -->

<div class="flex-container">

<div>Item 1</div>

<div>Item 2</div>

<div>Item 3</div>

</div>

<!-- Grid Example -->

<div class="grid-container">

<div>Item 1</div>

<div>Item 2</div>

<div>Item 3</div>

</div>

</body>

</html>

```

These examples should provide a clear understanding of the concepts, making it easier for candidates to grasp during an interview.

**JS**

Sure, here's how you can structure an interview question and answer based on the provided concepts:

\*\*Interview Question:\*\*

Explain the fundamental concepts of JavaScript programming that every developer should understand.

\*\*Answer:\*\*

JavaScript, as a versatile and widely used programming language, relies on several key concepts for effective development. Let's delve into some of these fundamental concepts:

1. \*\*Scope\*\*:

Scope in JavaScript defines the accessibility of variables. Understanding scope is crucial for managing variable lifetimes and avoiding naming conflicts.

- Example Question: What will be the output of the following code?

```javascript

function testScope() {

if (true) {

var x = 10;

}

console.log(x);

}

testScope();

```

- Example Answer: The output will be `10`. Despite `x` being declared inside the `if` block, it's accessible due to function scope when using `var`.

2. \*\*Closure\*\*:

Closures allow functions to retain access to variables from their parent scopes even after those scopes have closed. They are powerful constructs often used in JavaScript for data privacy and encapsulation.

- Example Question: How can you create a closure in JavaScript? Provide an example.

- Example Answer:

```javascript

function outerFunction() {

let outerVar = 10;

function innerFunction() {

console.log(outerVar);

}

return innerFunction;

}

let closureExample = outerFunction();

closureExample(); // Output: 10

```

3. \*\*Functions\*\*:

Functions in JavaScript are first-class citizens, meaning they can be assigned to variables, passed as arguments, and returned from other functions.

- Example Question: Differentiate between function declarations and function expressions in JavaScript.

- Example Answer:

```javascript

// Function declaration

function greet(name) {

return "Hello, " + name + "!";

}

// Function expression

let farewell = function(name) {

return "Goodbye, " + name + "!";

};

```

4. \*\*Data Types\*\*:

JavaScript supports various data types including primitive types like numbers, strings, booleans, null, and undefined, as well as complex types like objects and arrays.

- Example Question: What are the primitive data types in JavaScript? Give examples.

- Example Answer: Primitive data types include numbers (`let num = 10;`), strings (`let str = "Hello";`), booleans (`let bool = true;`), null (`let nul = null;`), and undefined (`let und = undefined;`).

5. \*\*Event Loop\*\*:

The event loop is a critical component of JavaScript's asynchronous nature, managing tasks such as I/O operations without blocking the execution of other code.

- Example Question: Explain how the event loop works in JavaScript and provide an example of an asynchronous operation.

- Example Answer:

```javascript

console.log("Start");

setTimeout(function() {

console.log("Async operation completed");

}, 2000);

console.log("End");

// Output: Start

// End

// (After 2 seconds) Async operation completed

```

These concepts form the backbone of JavaScript programming and are essential for building robust and efficient applications. Understanding them thoroughly is key to becoming proficient in JavaScript development.