← HTML QUESTIONS →

1. It is not a tag in HTML, it is called a Document Type Declaration. It is used to specify the version of HTML or XHTML .

—----------

-

2. Semantic tags are special elements that carry meaning and describe the structure and purpose of the content they enclose. They are:-

    1. <header>
    2. <nav>
    3. <article>
    4. <section>
    5. <footer>

—----------

--

3. HTML tags define the structure and markup of an HTML document.

- HTML elements consist of tags, attributes, and the content enclosed within the tags.

—----------

--

4. https://github.com/jatinfoujdar/placement-assignment-jatin-foujdar/tree/master/Html

—----------

--

6. a.New Semantic Elements

    b.multimedia support

c.canvas and svg

d.improved form and validation

e.improved performance

---

7. https://github.com/jatinfoujdar/placement-assignment-jatin-foujdar/blob/master/Html/song.html

---

8. Img tag is used to embed an image within an HTML document. It is a self-closing tag that does not require a closing tag. The `<img>`tag requires the `src` attribute, which specifies the source (URL) of the image file.

```
<img src="image.jpg" alt="Description of the image">
```

The `<figure>` tag is used to encapsulate media content, such as images, illustrations, diagrams, audio, video

```
 <figure>
  <img src="image.jpg" alt="Description of the image">
  <figcaption>Caption or description of the image</figcaption>
</figure>
```

← CSS QUESTIONS →

1. The Box Model is a fundamental concept in CSS (Cascading Style Sheets) that defines how elements are rendered and displayed on a web page. It describes the structure and layout of an element by conceptualizing it as a rectangular box. (Content,Padding,Border,Margin).

---

2. There are several types of selector : type selector (h1) element ,Class selector: (.container) , Id selector:(#header)

---

3. VW (Viewport Width) and VH (Viewport Height) are CSS units that are relative to the size of the browser's viewport.

---

4. Inline: elements do not start on a new line and only take up as much horizontal space as necessary to accommodate their content. `<span>`, `<a>`, `<strong>`, and `<em>`.

Inline-block: Inline-block elements share characteristics of both inline and block elements. `<img>`, `<input>`, and `<button>`.

Block:Block elements start on a new line and occupy the full available width by default. `<div>`, `<p>`, `<h1>` to `<h6>`, and `<section>`.

---

5. The `content-box` value is the default behavior of the box model.

The `border-box` value changes the way the box model is calculated.

6.the `z-index` property controls the stacking order of positioned elements along the z-axis, which determines which elements appear in front of or behind

others. The `z-index` property is used to specify a numeric value that determines the element's position in the stacking order.

7.Flexbox is a one-dimensional layout system, while Grid is a two-dimensional layout system.

Flexbox focuses on arranging elements along a single axis, while Grid enables control over both horizontal and vertical axes simultaneously.

9.
https://github.com/jatinfoujdar/placement-assignment-jatin-foujdar/tree/master/Css/Layout

---

10.
https://github.com/jatinfoujdar/placement-assignment-jatin-foujdar/tree/master/Css/mobilelayout

---

11.
https://github.com/jatinfoujdar/placement-assignment-jatin-foujdar/tree/master/Css/ineuron

---

12.
Pseudo-classes target elements based on certain states or conditions. They are prefixed with a colon ":" and are used to style elements that are in a particular state or meet specific criteria.
Pseudo-elements create virtual elements that do not exist in the HTML markup. They are also prefixed with a double colon "::" (although some pseudo-elements, such as `::before` and `::after`

← JavaScript Questions →

1. Hoisting is a JavaScript behavior that allows variable and function declarations to be moved to the top of their respective scopes during the compilation phase, before the code is executed.

---

2. In JavaScript, higher-order functions are functions that can take other functions as arguments or return functions as results.

   The `map()` method applies a provided function to each element in an array and returns a new array with the results.

   The `forEach()` method executes a provided function once for each element in an array. It does not create a new array but instead performs a specified action on each element of the array.

---

3. The `.call()` method invokes a function with a specified `this` value and individual arguments passed as comma-separated values.

```
 const person = {
 name: "John Doe",
 greet: function (greeting) {
   console.log(greeting + ", " + this.name);
 },
};

const otherPerson = {
```

```
  name: "Jane Smith",
};

person.greet.call(otherPerson, "Hello");
// Output: Hello, Jane Smith
```

The `.apply()` method invokes a function with a specified `this` value and an array (or an array-like object) of arguments.

```
const person = {
  name: "John Doe",
  greet: function (greeting) {
    console.log(greeting + ", " + this.name);
  },
};

const otherPerson = {
  name: "Jane Smith",
};

person.greet.apply(otherPerson, ["Hello"]);
// Output: Hello, Jane Smith
```

The `.bind()` method returns a new function with a specified `this` value and any pre-defined arguments. It does not immediately invoke the function but rather creates a new function that can be invoked later

```
const person = {
  name: "John Doe",
  greet: function (greeting) {
    console.log(greeting + ", " + this.name);
  },
};

const boundGreet = person.greet.bind(person, "Hello");
boundGreet();
// Output: Hello, John Doe
```

---

4. Event bubbling is the default behavior in which an event is first triggered on the innermost element and then propagated to its parent elements in the DOM hierarchy.

   Event capturing is the reverse mechanism where an event is triggered on the outermost element and then propagated to its innermost descendants before reaching the target element.

---

5. Function currying is a technique in functional programming where a function with multiple arguments is transformed into a series of functions, each taking a single argument. The transformed functions can be called one by one, each accepting an argument until all the arguments are satisfied, and then finally returning the result.

---

6. The execution starts from the top. The first line logs "First" to the console.
   The `setTimeout` function is called, which schedules the execution of the callback function (`() => console.log('Second')`) to occur after a minimum delay of 0 milliseconds. However, it does not immediately execute the callback and moves on to the next line.
   The third line logs "Third" to the console.

   First
   Third
   Second

   The execution starts from the top. The first line logs "First" to the console.
   The `secondCall` function is defined, but not immediately executed.
   The first `setTimeout` function is called, scheduling the execution of `secondCall` after a delay of 2000 milliseconds (2 seconds).

The second `setTimeout` function is called, scheduling the execution of `() => console.log('Third')` after a minimum delay of 0 milliseconds.
The final line logs "Third" to the console.
At this point, the initial execution is complete, and the JavaScript engine checks if there are any scheduled tasks in the event queue. There are no tasks with a minimum delay of 0 milliseconds, so it moves on.

First
Third
Second
Third

---

7. Promises are a feature in JavaScript that provide a way to handle asynchronous operations and manage the resulting values or errors. They allow you to write asynchronous code in a more readable and manageable way, avoiding the use of nested callbacks commonly known as "callback hell."

```
function fetchData() {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      const data = { id: 1, name: 'John Doe' };
      // Simulating a successful asynchronous operation
      resolve(data); // Resolve the promise with the data
      // Simulating an error during the asynchronous operation
      // reject(new Error('Failed to fetch data')); // Reject the promise with an error
    }, 2000);
  });
}

fetchData()
  .then((data) => {
    console.log('Data:', data);
  })
```

```
  .catch((error) => {
    console.error('Error:', error);
  });
```

---

8. In JavaScript, the `this` keyword refers to the context within which a function is executed. It represents the object that the function is bound to or the object that is currently being operated on. The value of `this` is determined dynamically based on how a function is invoked.

```
const obj = {
  prop: 'Hello',
  method: function() {
    console.log(this.prop);
  }
};

const obj2 = {
  prop: 'Goodbye'
};

obj.method(); // Output: Hello

const methodRef = obj.method;
methodRef(); // Output: undefined

obj2.method = obj.method;
obj2.method(); // Output: Goodbye
```

---

9. Call Stack: The call stack is like a stack of function calls. Whenever a function is invoked, it gets added to the top of the call stack.

Event Loop: The event loop is a mechanism in JavaScript that ensures the smooth execution of asynchronous code.

Callback Queue: The callback queue (also known as the task queue) is a queue that holds callback functions.

Microtask Queue: The microtask queue (also known as the Promise queue) is a special queue that holds microtasks.

—----------

--

10. Debouncing is a technique used in programming to control the frequency of function calls, particularly in scenarios where an event or action may trigger multiple consecutive function invocations in a short period.

—----------

--

11. A closure is created when a function is defined inside another function and has access to the variables, parameters, and inner functions of its outer function, even after the outer function has finished executing.

—----------

--

12. https://github.com/jatinfoujdar/placement-assignment-jatin-foujdar/tree/master/js

—----------

--

← React Questions →

1. **Component-based architecture, Virtual DOM,Declarative syntax**.

---

2. The Virtual DOM (VDOM) is a concept in React that provides a virtual representation of the actual Document Object Model (DOM) in memory.

   Efficiency , Faster rendering , Simplified programming model

---

3. **Mounting Phase , Updating Phase , Unmounting Phase**

---

4. Functional Components:

- Functional components are defined as JavaScript functions.
- They have a simpler and more concise syntax compared to class components.

Class Components:

- Class components are defined as JavaScript classes that extend the React.Component class.
- They have a more verbose syntax compared to functional components.

---

5. They provide a way to reuse stateful logic without using class components.

   useState , useEffect ,useContext , useReducer.

---

6. **Control over component, behaviorOptimized, renderingSide , effects and asynchronous, tasksCleanup and resource management**

---

7. The `useState` hook is a built-in hook in React that allows functional components to have state. It provides a way to declare state variables and update them within a functional component.

   Simplified state, management , Easy to understand and use ,Functional programming approach , Multiple state variables.

---

8. The `useEffect` hook is a React hook that allows you to perform side effects in functional components. Side effects are actions that have an impact on the outside world, such as fetching data from an API or making a network request

---

9. `useReducer` is a hook provided by React, a JavaScript library for building user interfaces. It is an alternative to the `useState` hook and is used for managing state in more complex scenarios.

   Complex state management , Predictable state transitions , Better code organization

---

10. https://github.com/jatinfoujdar/Pin-Notes

12.https://github.com/jatinfoujdar/placement-assignment-jatin-foujdar/tree/master/c_alculator-app

---

13. https://github.com/jatinfoujdar/javascript-component/tree/main/counter

---

14. https://github.com/jatinfoujdar/tic-tac-toe

---

15. Prop drilling refers to the process of passing down props through multiple layers of components, even if some intermediate components do not need those props.

Readability and Maintainability , Code Duplication , Brittleness

---

← Express Question →

1. https://github.com/jatinfoujdar/placement-assignment-jatin-foujdar/tree/master/exblog

---

3. https://github.com/jatinfoujdar/blog-MERN-JWT

---

6. https://github.com/jatinfoujdar/blog-MERN-JWT

---

7. Hashing password

---

8. Event Loop Phases , Event Queue , Event Loop Iteration , Non-Blocking Operations

---

9. https://github.com/jatinfoujdar/drip-ecommerse-MERN