Interview Preparation

**\*JavaScript\***

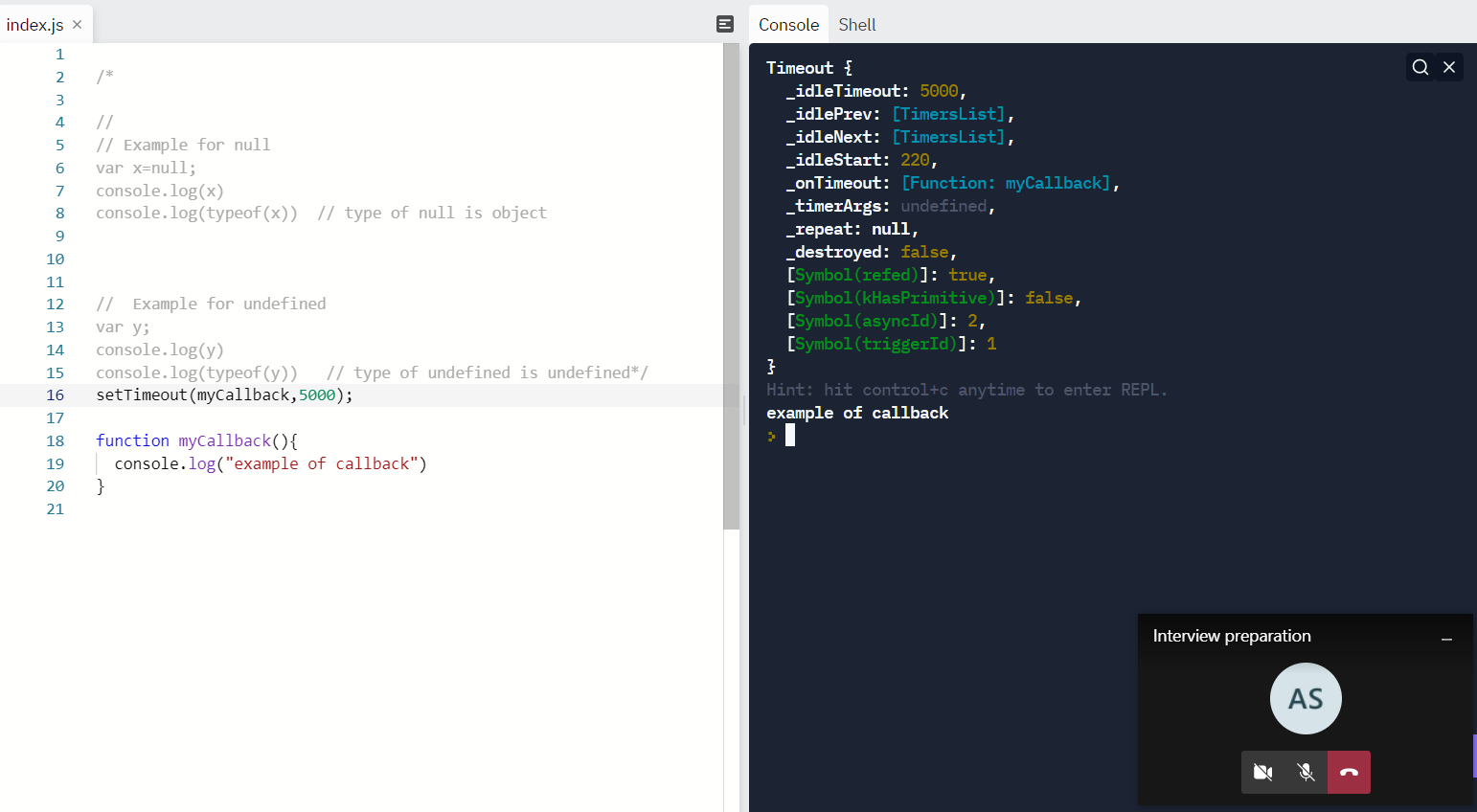
1) Closures in JavaScript :

A **closure** is the combination of a function bundled together (enclosed) with references to its surrounding state (the **lexical environment**). In other words, a closure gives you access to an outer function's scope from an inner function. In JavaScript, closures are created every time a function is created, at function creation time.

2) Promises in JavaScript :

Promises and callbacks both are use to perform asynchronous operations.

**Callback:** when function is passed as argument to another function is a callback function.

****

In above code myCallback is passed as argument in setTimeout function.

To avoid call back hell we prefer to use promises over callback function

**Callback hell :**  nesting of callback is called as callabck hell.

**3)Session Storage and Local Storage:**

The read-only **sessionStorage** property accesses a session [Storage](https://developer.mozilla.org/en-US/docs/Web/API/Storage) object for the current [origin](https://developer.mozilla.org/en-US/docs/Glossary/Origin). sessionStorage is similar to [localStorage](https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage" \o "localStorage); the difference is that while data in localStorage doesn't expire, data in sessionStorage is cleared when the page session ends.

* Whenever a document is loaded in a particular tab in the browser, a unique page session gets created and assigned to that particular tab. That page session is valid only for that particular tab.
* A page session lasts as long as the tab or the browser is open, and survives over page reloads and restores.
* Opening a page in a new tab or window creates a new session with the value of the top-level browsing context, which differs from how session cookies work.

**4) What are cookies in Javascript:**

A cookie is an amount of information that persists between a server-side and a client-side. A web browser stores this information at the time of browsing.

A cookie contains the information as a string generally in the form of a name-value pair separated by semi-colons. It maintains the state of a user and remembers the user's information among all the web pages.

In JavaScript, we can create, read, update and delete a cookie by using **document.cookie** property.

**\*\*React\*\***

**1) Lifecycle methods in react:** React components pass through three lifecycles: Mounting, Updating and Unmounting.

* **Mounting:**  it is simply putting elements in the DOM. At this stage, react has built in methods which are:

1. constructor() => useState()
2. getDerivedStateFromProps()
3. render()
4. componentDidMount() =>useEffect()

At this stage, the render method will always be called in class components. The other methods will be called if you define them.

* **Updating:** A component is updated, whenever there is a change in the component. This could be either via props or state. The react built in method for updated components are:

1. getDerivedStateFromProps()
2. shouldComponentUpdate()
3. render()
4. getSnapshotBeforeUpdate()
5. componentDidUpdate()

* **Unmounting:** A component is unmounted when it is removed from the DOM. There is just one built in method here, which is componentWillUnmount. These component lifecycles are only available when writing class components. With functional components, some of these lifecycle methods are mimicked with react hooks. Mostly useState() and useEffect() hook.

### **2)** controlled and uncontrolled components in JavaScript :

**Controlled component:**In a controlled component, the value of the input element is controlled by React. We store the state of the input element inside the code, and by using event-based callbacks, any changes made to the input element will be reflected in the code as well.

When a user enters data inside the input element of a controlled component, onChange function gets triggered and inside the code, we check whether the value entered is valid or invalid. If the value is valid, we change the state and re-render the input element with the new value.

**Uncontrolled component:** In an uncontrolled component, the value of the input element is handled by the DOM itself. Input elements inside uncontrolled components work just like normal HTML input form elements.

The state of the input element is handled by the DOM. Whenever the value of the input element is changed, event-based callbacks are not called. Basically, react does not perform any action when there are changes made to the input element.

**3) Prop Drilling in React.js:**

Sometimes while developing React applications, there is a need to pass data from a component that is higher in the hierarchy to a component that is deeply nested. To pass data between such components, we pass props from a source component and keep passing the prop to the next component in the hierarchy till we reach the deeply nested component.

The **disadvantage** of using prop drilling is that the components that should otherwise be not aware of the data have access to the data.

**\*\*MongoDB Database\*\***

**1)What is Mongo dB :-**

* MongoDB is an open-source NoSQL database written in C++ language. It uses JSON-like documents with optional schemas.
* It provides easy scalability and is a cross-platform, document-oriented database.
* MongoDB works on the concept of Collection and Document.
* It combines the ability to scale out with features such as secondary indexes, range queries, sorting, aggregations, and geospatial indexes.

2**)Document and Collections in MongoDB: -**

**Document :-**

A Document in MongoDB is an ordered set of keys with associated values. It is represented by a map, hash, or dictionary. In JavaScript, documents are represented as objects,

Example :- 1) { “name” : “Vaibhav”}

2) { “name” : “Vaibhav”,

“g-mail”:[”vai@gmail.com](mailto:vai@gmail.com)”}

**Collections: -**

A collection in MongoDB is a group of documents. If a document is the MongoDB analog of a row in a relational database, then a collection can be thought of as the analog to a table.  
Documents within a single collection can have any number of different “shapes.”, i.e. collections have dynamic schemas.   
For example, both of the following documents could be stored in a single collection:

**3) How to add data in mongo dB?**

- Basic method of adding data to mongo dB is “inserts”.

- to add single document in mongo dB we can use “insertOne” method.

- to add multiple document in mongo dB we can use “insertMany” method.

Example-

db.inventory.insertOne(

{ "item" : "canvas",

"qty" : 100,

"tags" : ["cotton"],

"size" : { "h" : 28, "w" : 35.5, "uom" : "cm" }

}

)

**\*\*Node Js\*\***

**1)What is Node Js :**

**-** Node.js is a cross-platform runtime environment and library for running JavaScript applications outside the browser. It is used for creating server-side and networking web applications

-Node.js is a virtual machine that uses JavaScript as its scripting language and runs Chrome’s V8 JavaScript engine.

-Basically, Node.js is based on an event-driven architecture where I/O runs asynchronously making it lightweight and efficient.

-It is being used in developing desktop applications as well with a popular framework called electron as it provides API to access OS-level features such as file system, network, etc.

**2) How to create Simple server using node.js ?**