Interview Preparation

**\*JavaScript\***

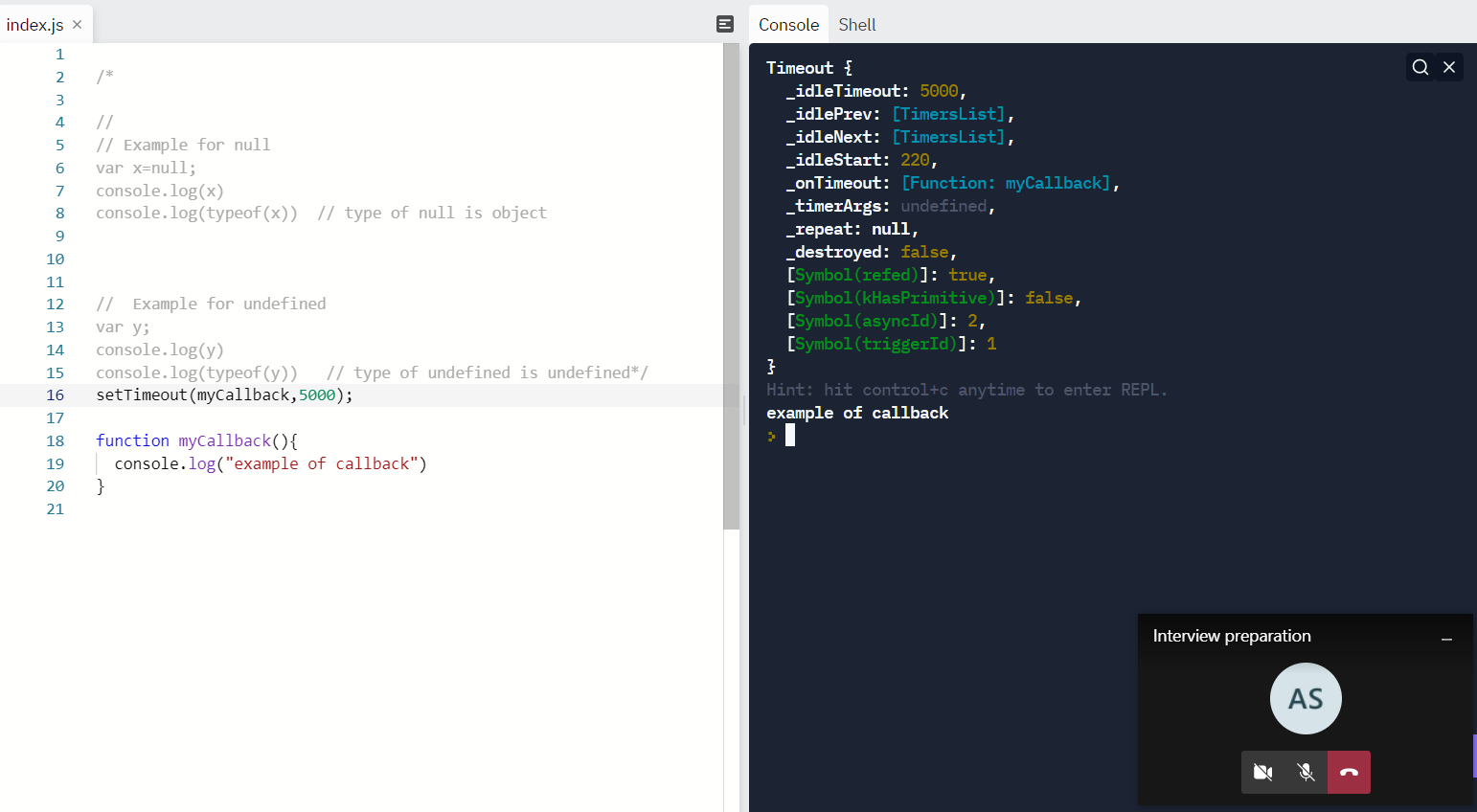
1) Closures in JavaScript :

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.

**\*\*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.