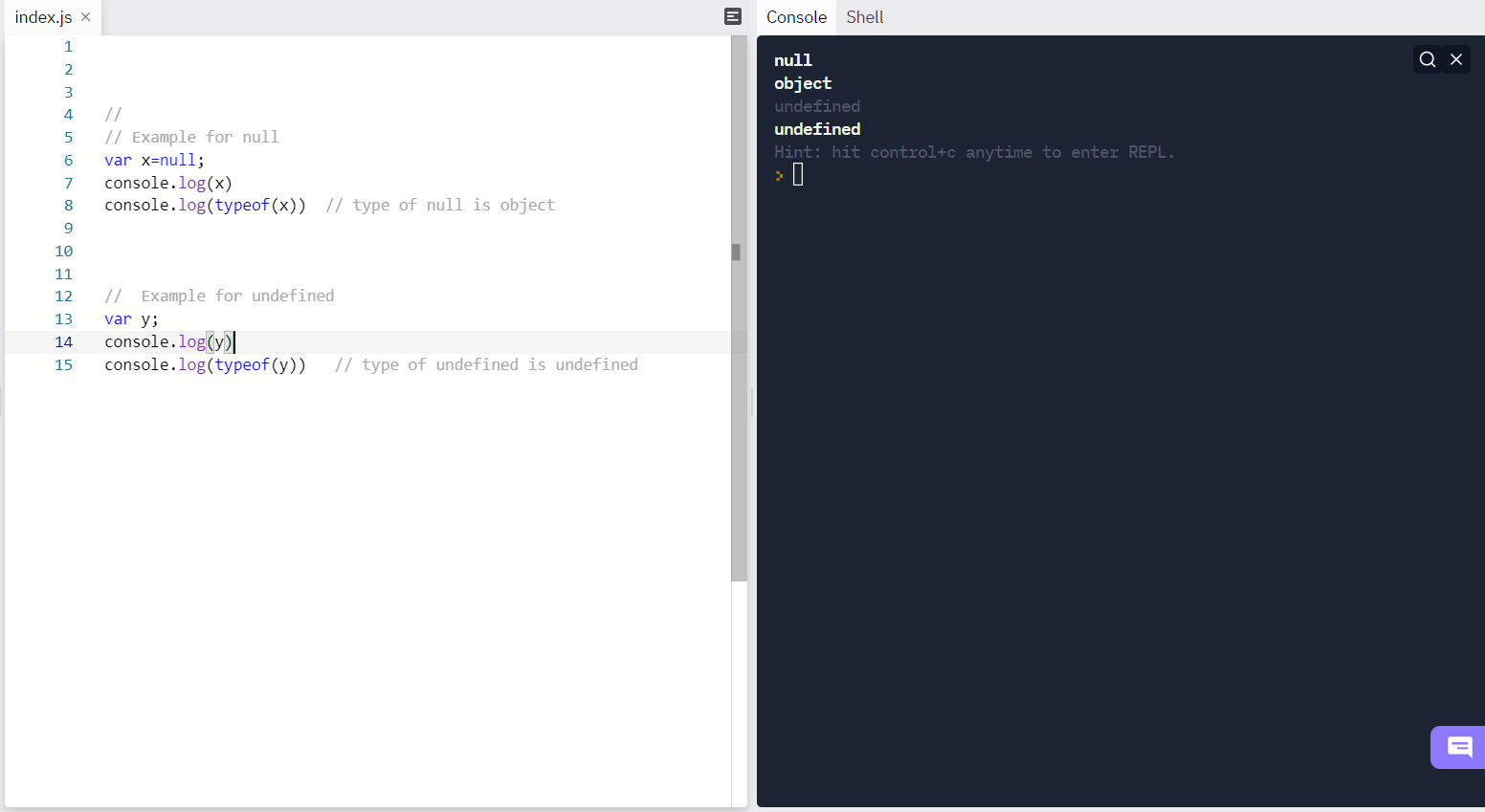
* **JAVASCRIPT :**

1. Difference between null and undefined?

null : An empty or non-existent value. Which assigned to the variable but not represented any value.

Undefined : variable is defined but not assigned any value which results as undefined.



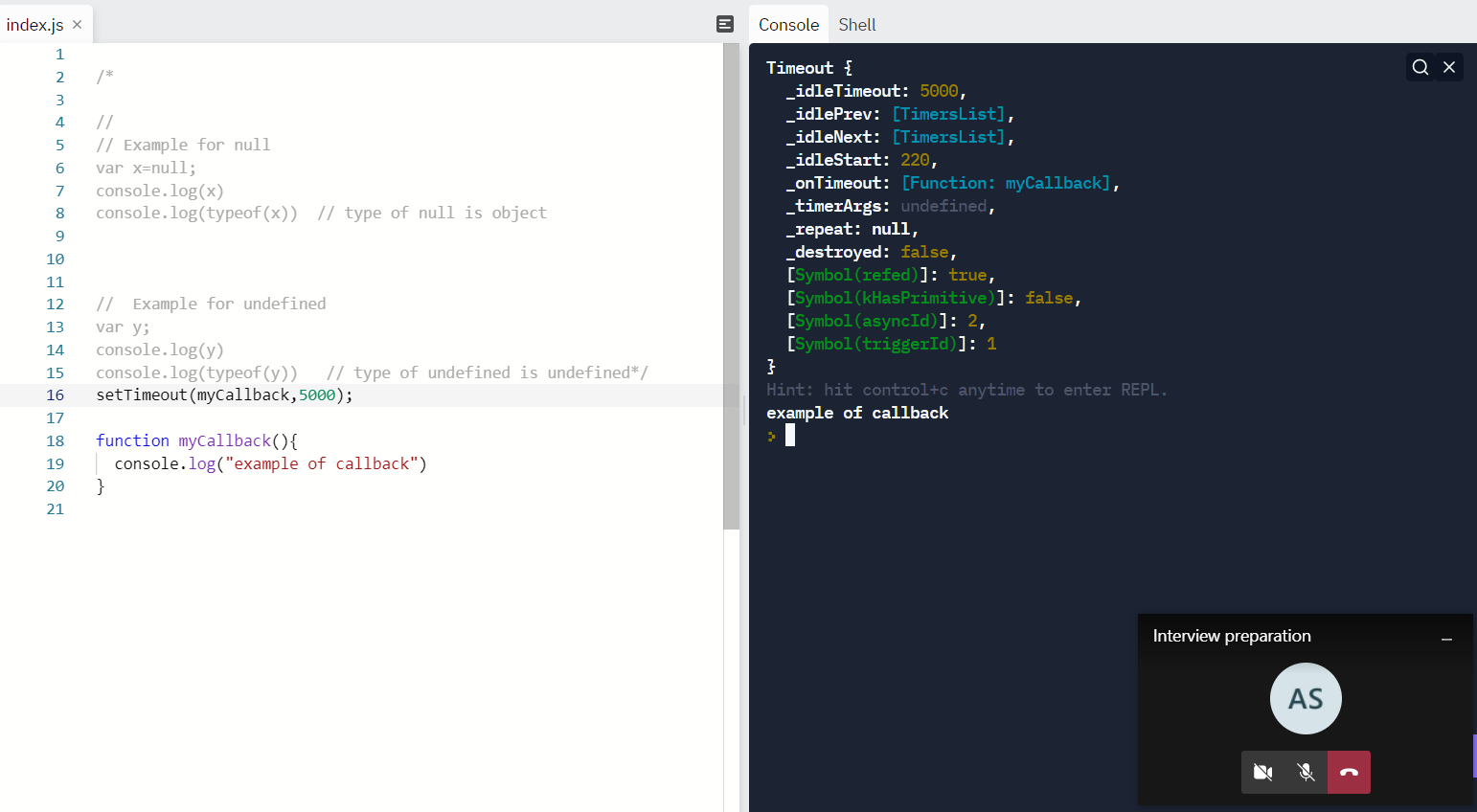
**null VS undefined Difference :**

* data type of null is object and data type of undefined in undefined.
* When used in arithmetic operations, **undefined** will result in **NaN** (not a number), whereas **null** will be converted to **0** behind the screens.
* Null !=== undefined but undefined===null
* Both are falsy values when use in Boolean operations

2) **Promises and callback 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.

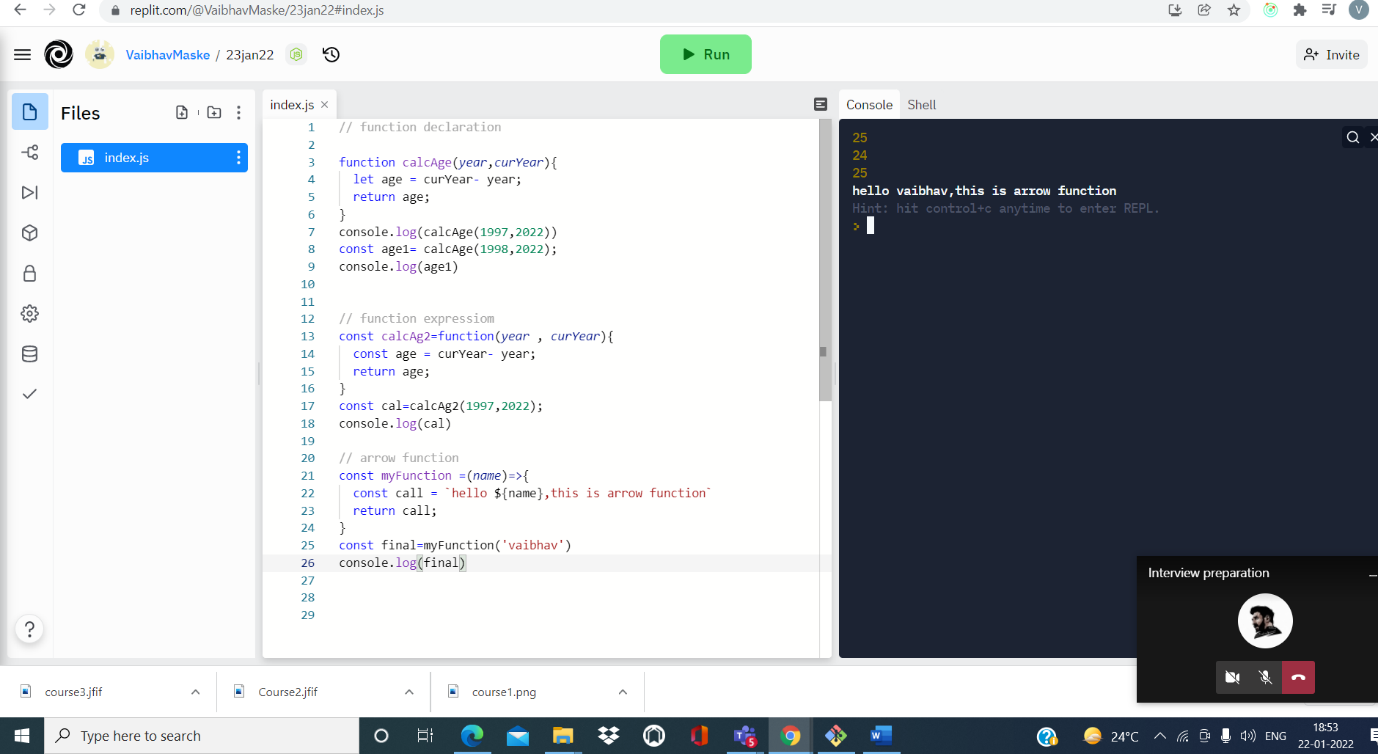
****

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)Function in JavaScript :**



**Regular function and arrow function:**

In regular function can be define in two ways A] function declaration

B] function expression.

**Function declaration :**

Syntax :

function Myfunction(parameters){

// code here

return;

}

**function expression :**

syntax :

const myFunction = function(parameters){

// code here;

Return;

}

**Arrow Fuction :**

Syntax :

const myFunction =()=>{

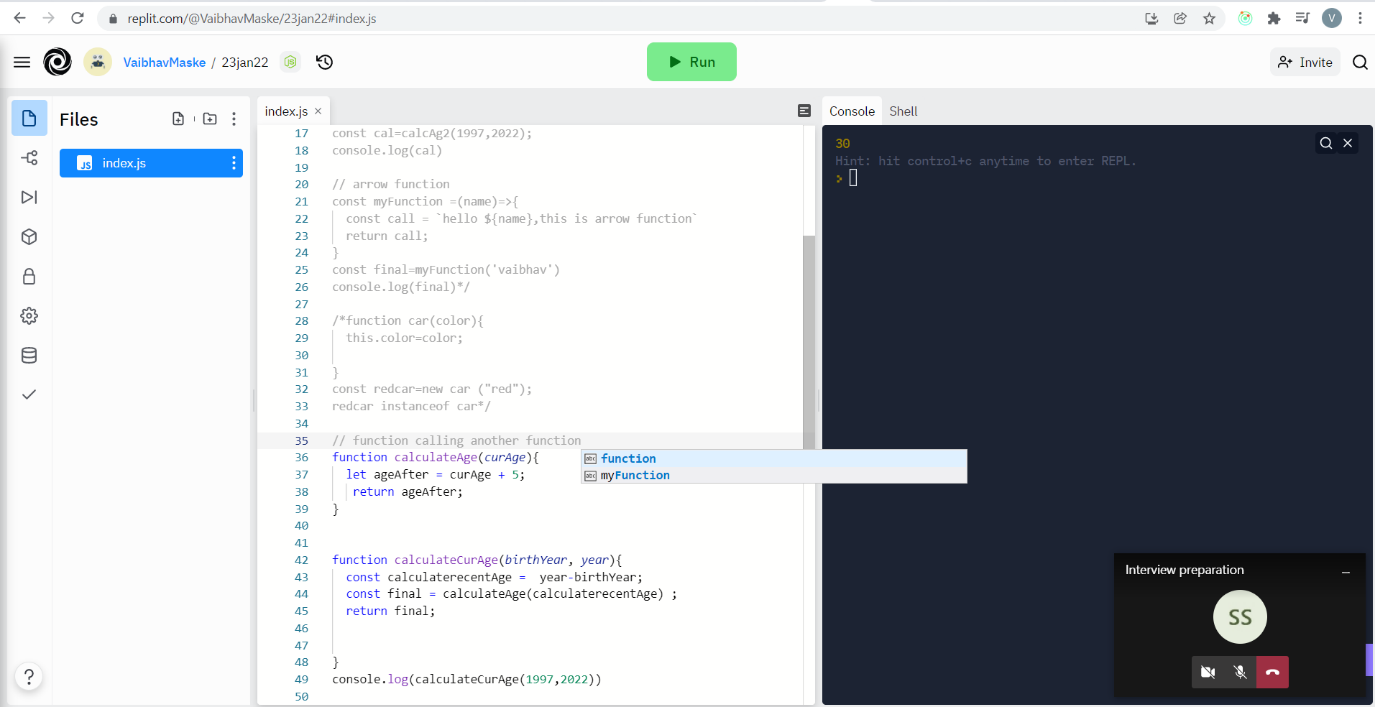
// code here ;

return;}

**Difference between Standard function & arrow function in JavaScript:**

1. Arrow functions are least human readable than standard function.
2. Arrow function does not support this keyword. While standard function support this keyword

**Function calling another function :**

****

**React:**

1. **What is react ?**

* ReactJS is a **declarative**, **efficient**, and flexible **JavaScript library** for building reusable UI components. It is an open-source, component-based front end library which is responsible only for the view layer of the application

1. **Advantages of React Js :**

-It uses virtual DOM (JavaScript object), which improves the performance of the app. The JavaScript virtual DOM is faster than the regular DOM.

- We can use ReactJS on the client and server-side as well as with other frameworks.

-It uses component and data patterns that improve readability and helps to maintain larger apps.

* **Use of Virtual DOM to improve efficiency:**React uses virtual DOM to render the view. As the name suggests, virtual DOM is a virtual representation of the real DOM. Each time the data changes in a react app, a new virtual DOM gets created. Creating a virtual DOM is much faster than rendering the UI inside the browser. Therefore, with the use of virtual DOM, the efficiency of the app improves.
* **Gentle learning curve:** React has a gentle learning curve when compared to frameworks like Angular. Anyone with little knowledge of javascript can start building web applications using React.
* **SEO friendly:** React allows developers to develop engaging user interfaces that can be easily navigated in various search engines. It also allows server-side rendering, which boosts the SEO of an app.
* **Reusable components:**React uses component-based architecture for developing applications. Components are independent and reusable bits of code. These components can be shared across various applications having similar functionality. The re-use of components increases the pace of development.
* **Huge ecosystem of libraries to choose from:**React provides you with the freedom to choose the tools, libraries, and architecture for developing an application based on your requirement

**3) How virtual DOM works :**

- Virtual DOM is act as a Blueprint of real DOM .

- when object gets updated every time new virtual DOM create and it compare the updated object with the previous state of object .

-after comparison it update the changes to the real DOM, after that react render the only updated part .in this way virtual DOM avoid the inefficient updating.

Webpage 🡺 virtual DOM 🡺 real DOM

**4)What is state in React :**

-The state is an instance of React Component Class can be defined as an object of a set of **observable** properties that control the behavior of the component.

-In other words, the State of a component is an object that holds some information that may change over the lifetime of the component.

**5) what are the props in React:**

-Props stand for "**Properties**."

-They are **read-only** components.

-It is an object which stores the value of attributes of a tag and work similar to the HTML attributes. It gives a way to pass data from one component to other components.-

It is similar to function arguments.

-Props are passed to the component in the same way as arguments passed in a function.

**6)difference between state and props:**

* Props are immutable i.e. once set the props cannot be changed, while State is an observable object that is to be used to hold data that may change over time and to control the behavior after each change.
* States can be used in Class Components, Functional components with the use of React Hooks (useState and other methods) while Props don’t have this limitation.
* While Props are set by the parent component, State is generally updated by event handlers.

**7) prop drilling in React :**

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

**8) React hooks:**

Hooks are functions that let us “hook into” React state and lifecycle features from a **functional component.**

React Hooks**cannot** be used in class components. They let us write components without class.

**9)Why were hooks introduced in react :**

React hooks were introduced in the 16.8 version of React. Previously, functional components were called stateless components. Only class components were used for state management and lifecycle methods. The need to change a functional component to a class component, whenever state management or lifecycle methods were to be used, led to the development of Hooks.

Example of a hook:**useState hook:**

In functional components, the useState hook lets us define a state for a component

**10) Lifecycle methods in react:**

React components pass through three lifecycles: Mounting, Updating and Unmounting.

* **Mounting:** here, 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

### 11) What is the use of useEffect React Hooks?

The useEffect React Hook is used for performing the side effects in functional components. With the help of useEffect, you will inform React that your component requires something to be done after rendering the component or after a state change. The function you have passed(can be referred to as “effect”) will be remembered by React and call afterwards the performance of DOM updates is over. Using this, we can perform various calculations such as data fetching, setting up document title, manipulating DOM directly, etc, that don’t target the output value. The useEffect hook will run by default after the first render and also after each update of the component. React will guarantee that the DOM will be updated by the time when the effect has run by it.

The useEffect React Hook will accept 2 arguments: useEffect(callback[, dependencies]);

Where the first argument callback represents the function having the logic of side-effect and it will be immediately executed after changes were being pushed to DOM. The second argument dependencies represent an optional array of dependencies. The useEffect() will execute the callback only if there is a change in dependencies in between renderings.