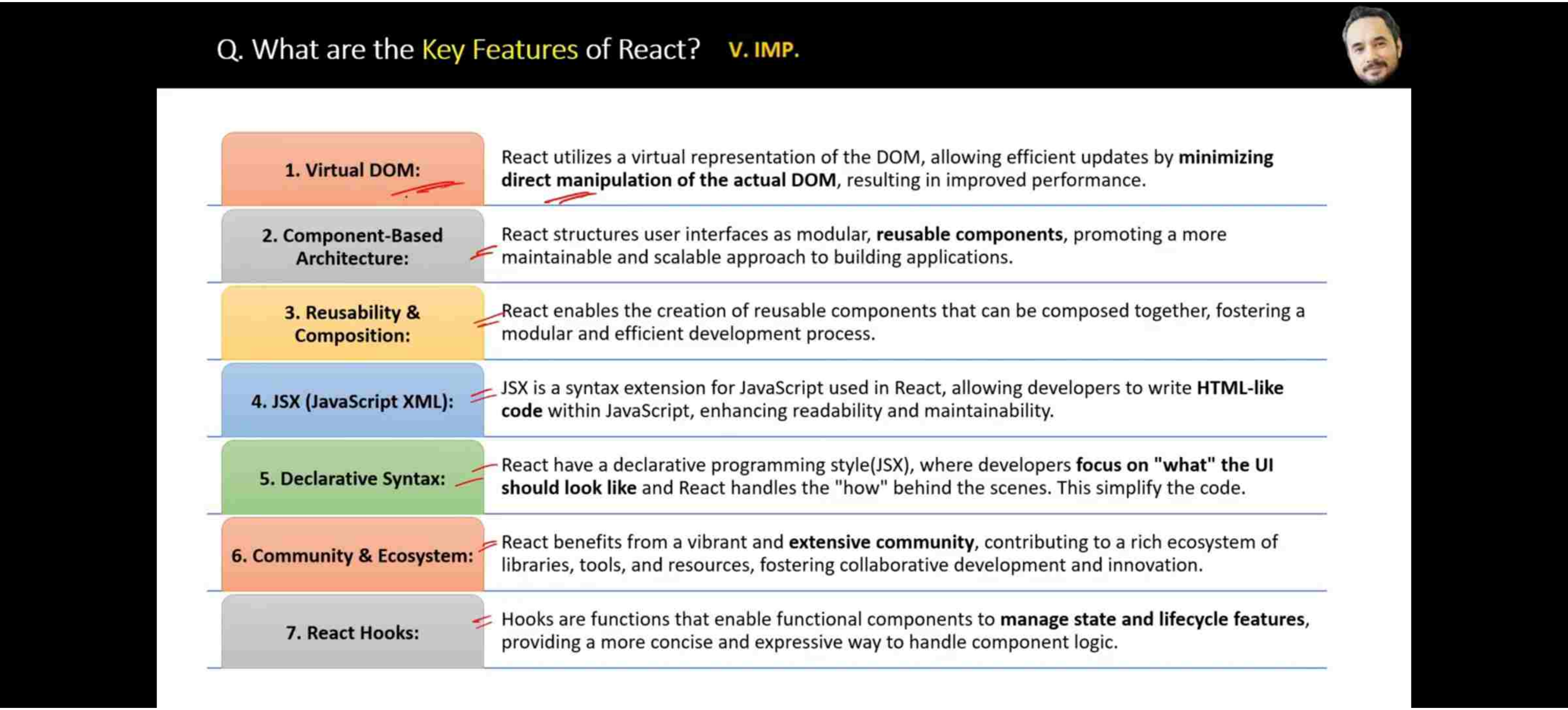


Q. What are the Key Features of React? V. IMP.

1. Virtual DOM: React utilizes a virtual representation of the DOM, allowing efficient updates by **minimizing direct manipulation of the actual DOM**, resulting in improved performance.

2. Component-Based Architecture: React structures user interfaces as modular, **reusable components**, promoting a more maintainable and scalable approach to building applications.

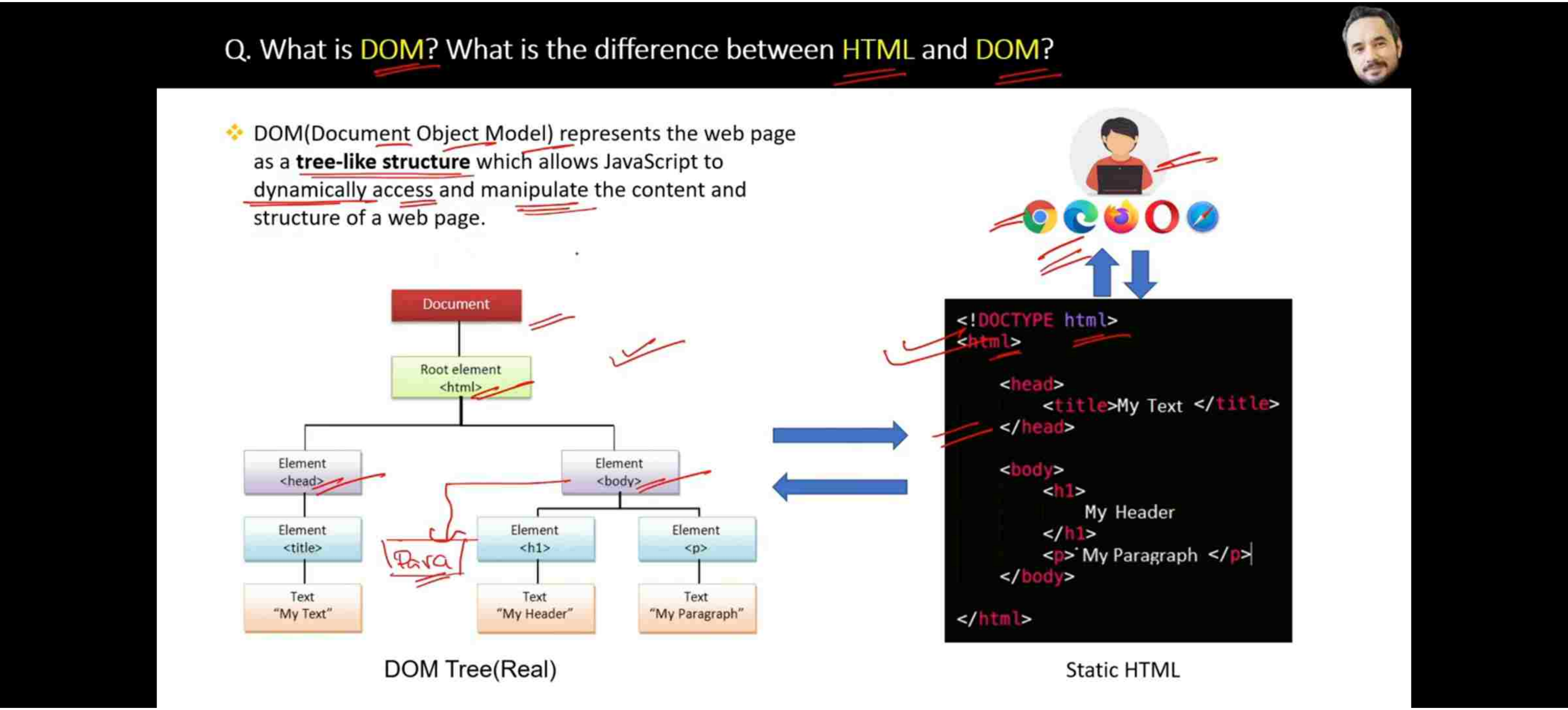
3. Reusability & Composition: React enables the creation of reusable components that can be composed together, fostering a modular and efficient development process.

4. JSX (JavaScript XML): JSX is a syntax extension for JavaScript used in React, allowing developers to write **HTML-like code** within JavaScript, enhancing readability and maintainability.

5. Declarative Syntax: React have a declarative programming style(JSX), where developers **focus on "what" the UI should look like** and React handles the "how" behind the scenes. This simplify the code.

6. Community & Ecosystem: React benefits from a vibrant and **extensive community**, contributing to a rich ecosystem of libraries, tools, and resources, fostering collaborative development and innovation.

7. React Hooks: Hooks are functions that enable functional components to **manage state and lifecycle features**, providing a more concise and expressive way to handle component logic.



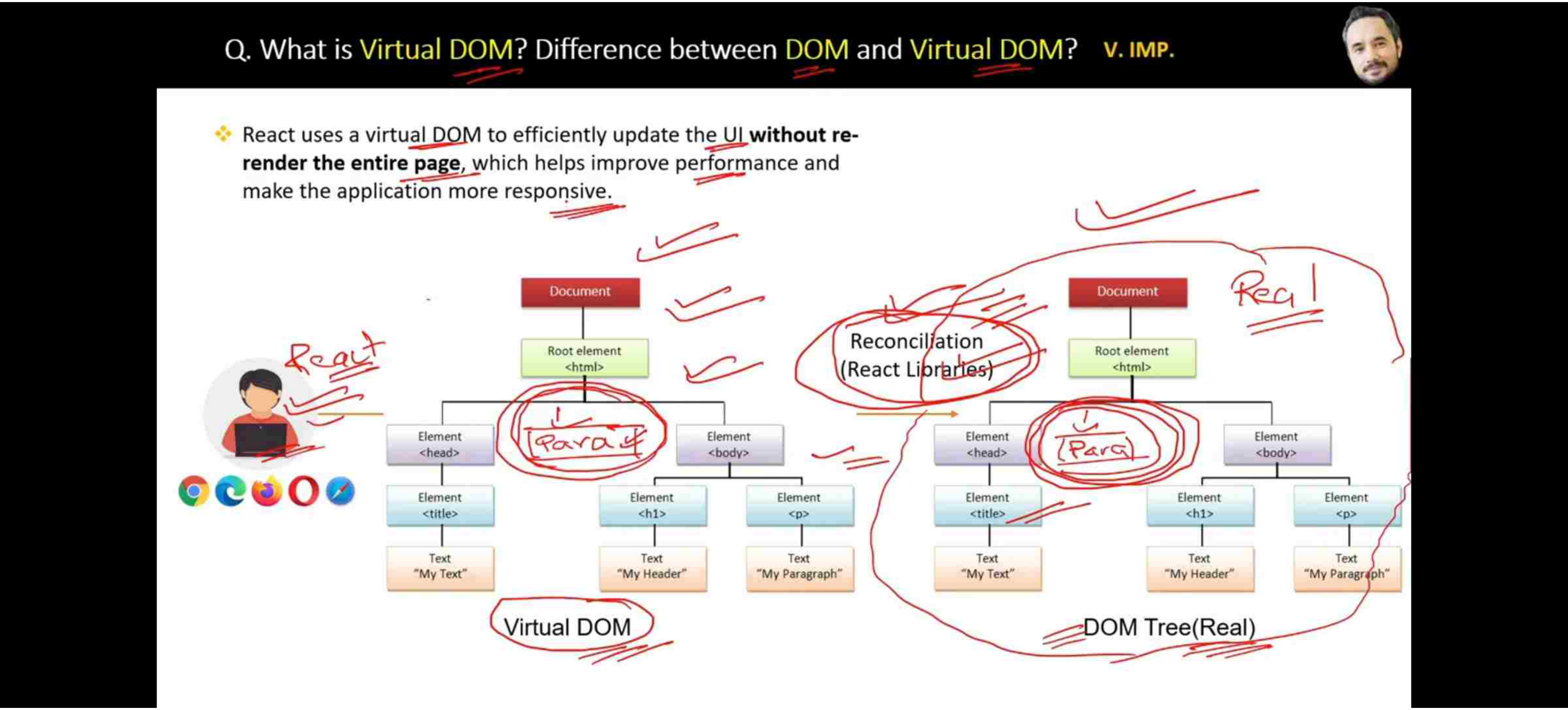
Q. What is DOM? What is the difference between HTML and DOM?

- DOM(Document Object Model) represents the web page as a **tree-like structure** which allows JavaScript to dynamically access and manipulate the content and structure of a web page.

DOM Tree(Real)

```
<!DOCTYPE html>
<html>

    <head>
        <title>My Text </title>
    </head>

    <body>
        <h1>
            My Header
        </h1>
        <p> My Paragraph </p>
    </body>

</html>
```

Static HTML



Q. What is Virtual DOM? Difference between DOM and Virtual DOM? V. IMP.

- React uses a virtual DOM to efficiently update the UI **without re-render the entire page**, which helps improve performance and make the application more responsive.

React

Document

Root element <html>

Element <head>

Element <title>

Text "My Text"

Element <body>

Element <h1>

Text "My Header"

Element <p>

Text "My Paragraph"

Para

Virtual DOM

Reconciliation (React Libraries)

Real

Document

Root element <html>

Element <head>

Element <title>

Text "My Text"

Element <body>

Element <h1>

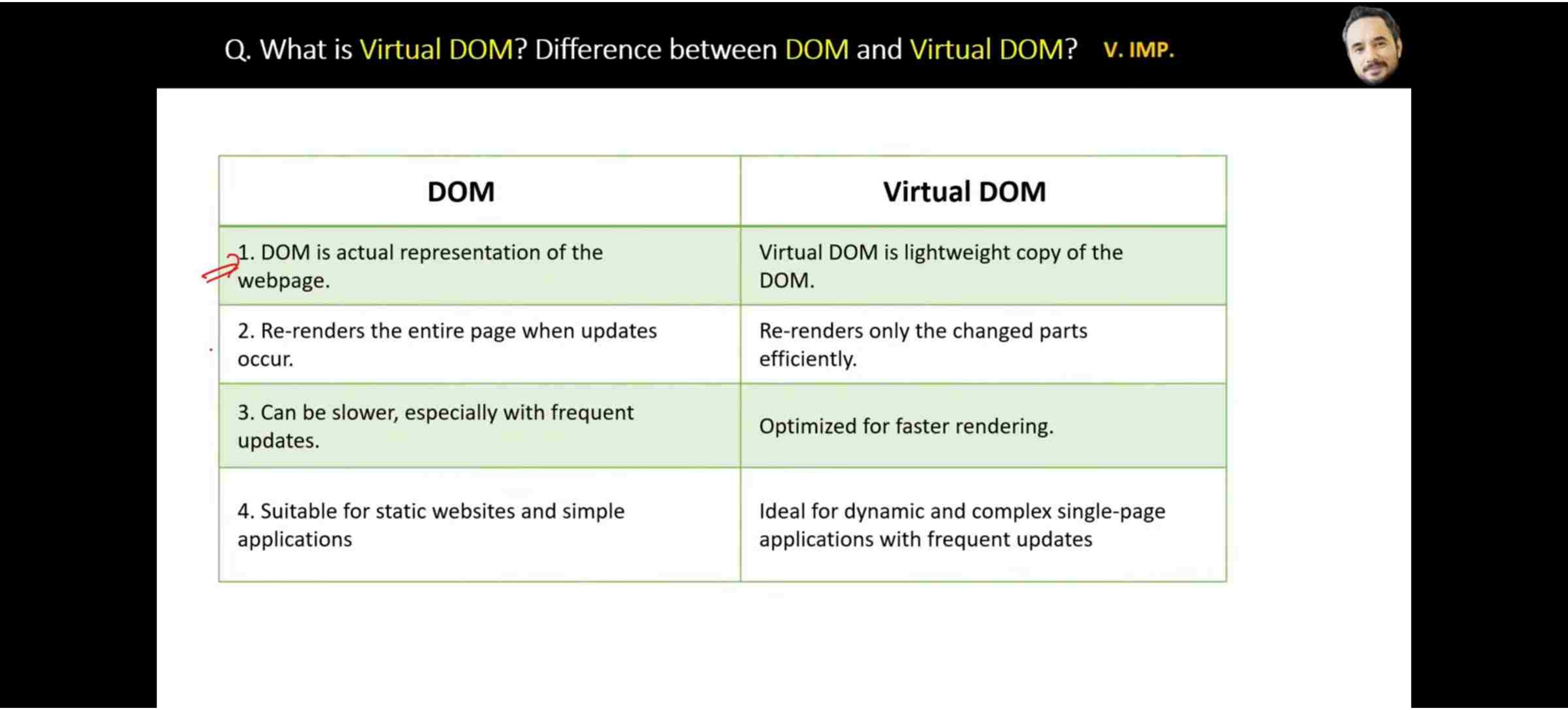
Text "My Header"

Element <p>

Text "My Paragraph"

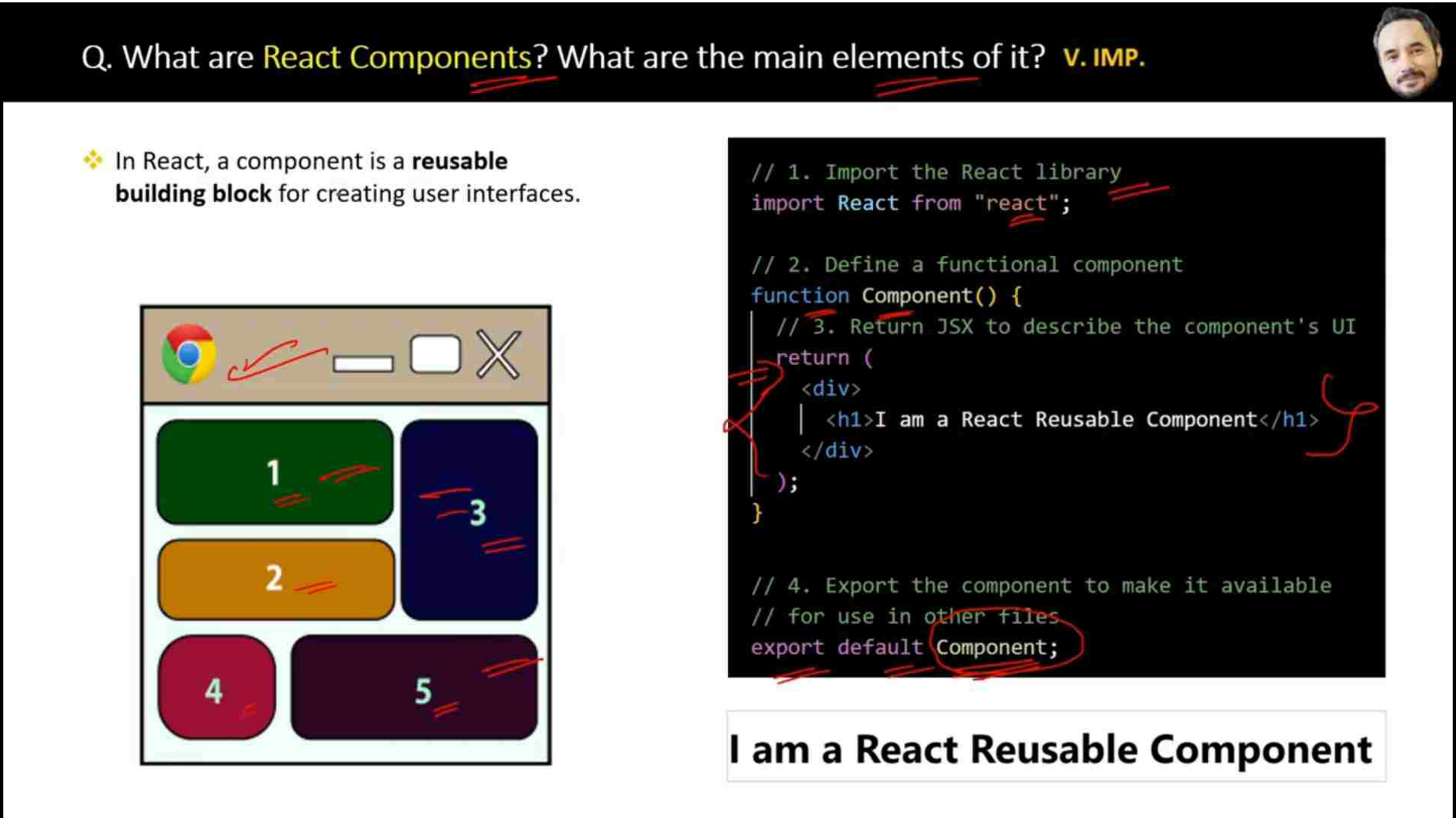
Para

DOM Tree(Real)



Q. What is Virtual DOM? Difference between DOM and Virtual DOM? V. IMP.

DOM	Virtual DOM
1. DOM is actual representation of the webpage.	Virtual DOM is lightweight copy of the DOM.
2. Re-renders the entire page when updates occur.	Re-renders only the changed parts efficiently.
3. Can be slower, especially with frequent updates.	Optimized for faster rendering.
4. Suitable for static websites and simple applications	Ideal for dynamic and complex single-page applications with frequent updates



Q. What are React Components? What are the main elements of it? V. IMP.

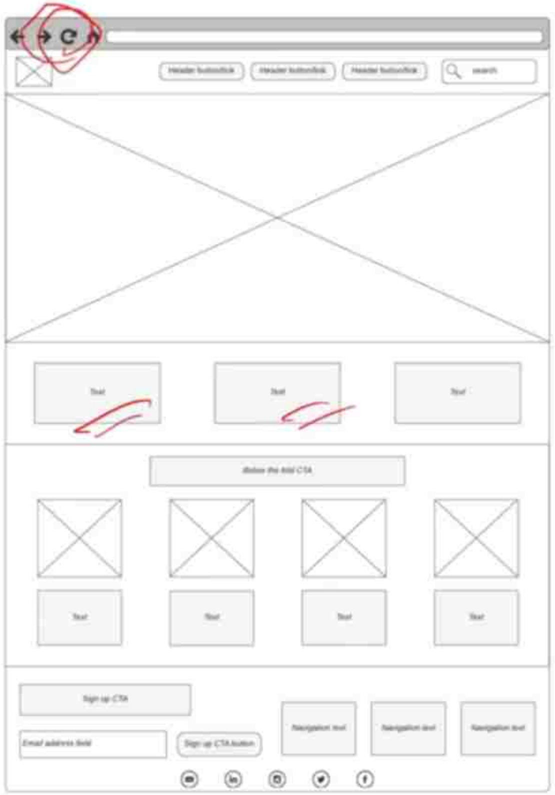
- In React, a component is a **reusable building block** for creating user interfaces.

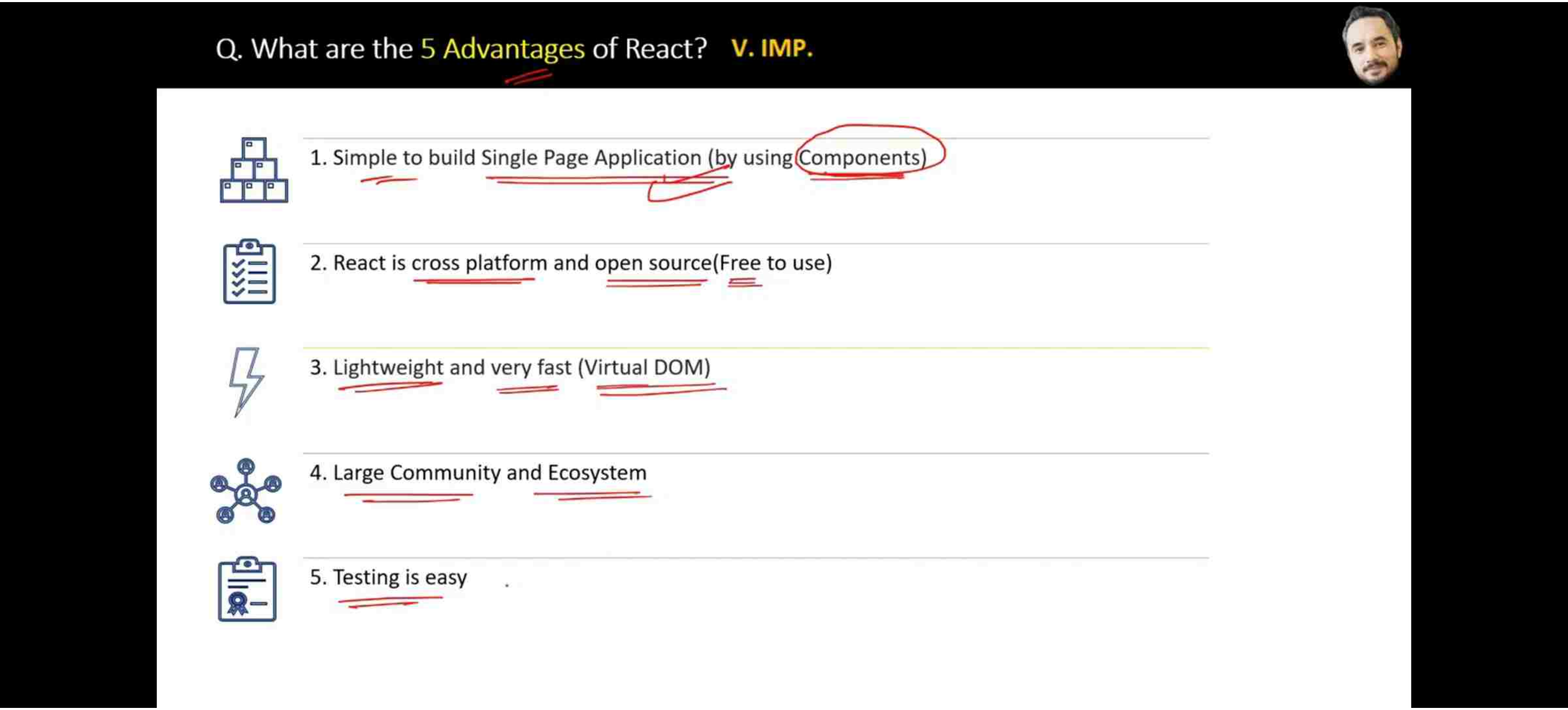
```
// 1. Import the React library
import React from "react";

// 2. Define a functional component
function Component() {
  // 3. Return JSX to describe the component's UI
  return (
    <div>
      <h1>I am a React Reusable Component</h1>
    </div>
  );
}

// 4. Export the component to make it available
// for use in other files
export default Component;
```

I am a React Reusable Component

Q. What is SPA(Single Page Application)?

- A Single Page Application (SPA) is a web application that have only one **single web page**.
- Whenever user do some action on the website, then in response content is dynamically updated without refreshing or loading a new page.





Q. What are the 5 Advantages of React? **V. IMP.**

1. Simple to build Single Page Application (by using Components)

2. React is cross platform and open source(Free to use)

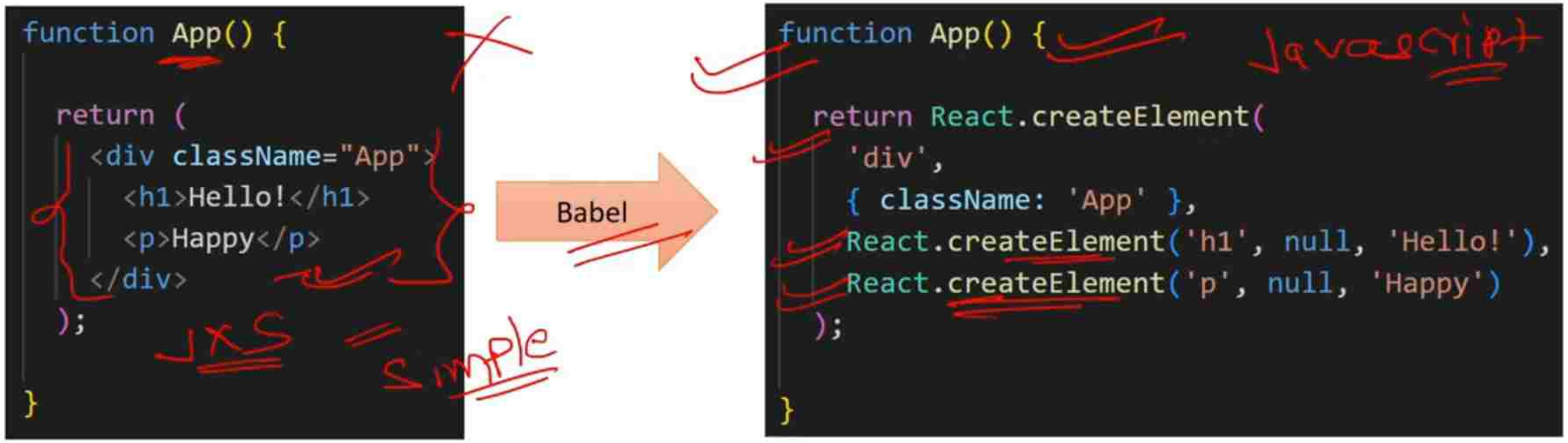
3. Lightweight and very fast (Virtual DOM)

4. Large Community and Ecosystem

5. Testing is easy

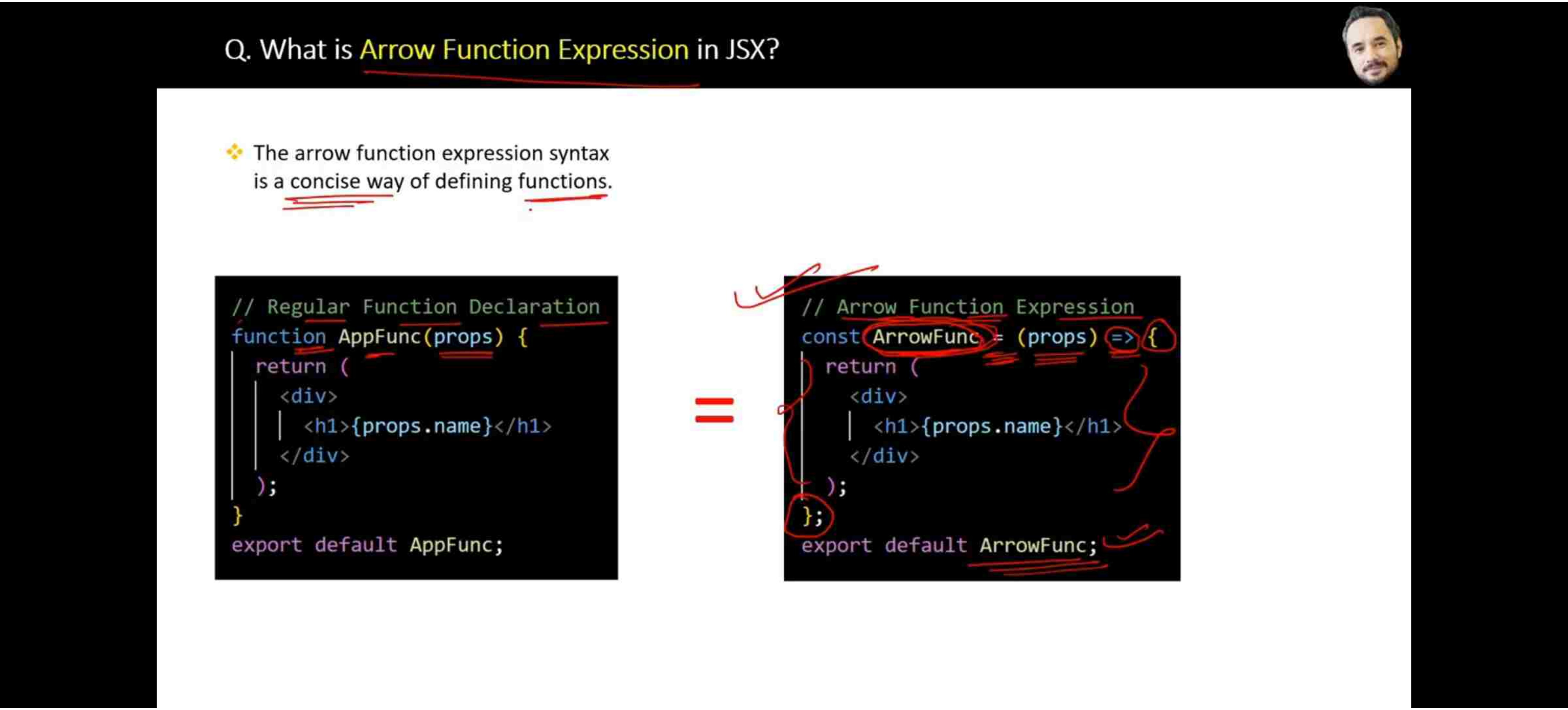
Q. What is the role of JSX in React? (3 points) **V. IMP.**

1. JSX stands for **JavaScript XML**.
2. JSX is used by React to write **HTML-like code**.
3. JSX is converted to JavaScript via tools like **Babel**.
   Because Browsers understand JavaScript not JSX.



```
function App() {

  return (
    <div className="App">
      <h1>Hello!</h1>
      <p>Happy</p>
    </div>
  );

}
```

```
function App() {

  return React.createElement(
    'div',
    { className: 'App' },
    React.createElement('h1', null, 'Hello!'),
    React.createElement('p', null, 'Happy')
  );

}
```



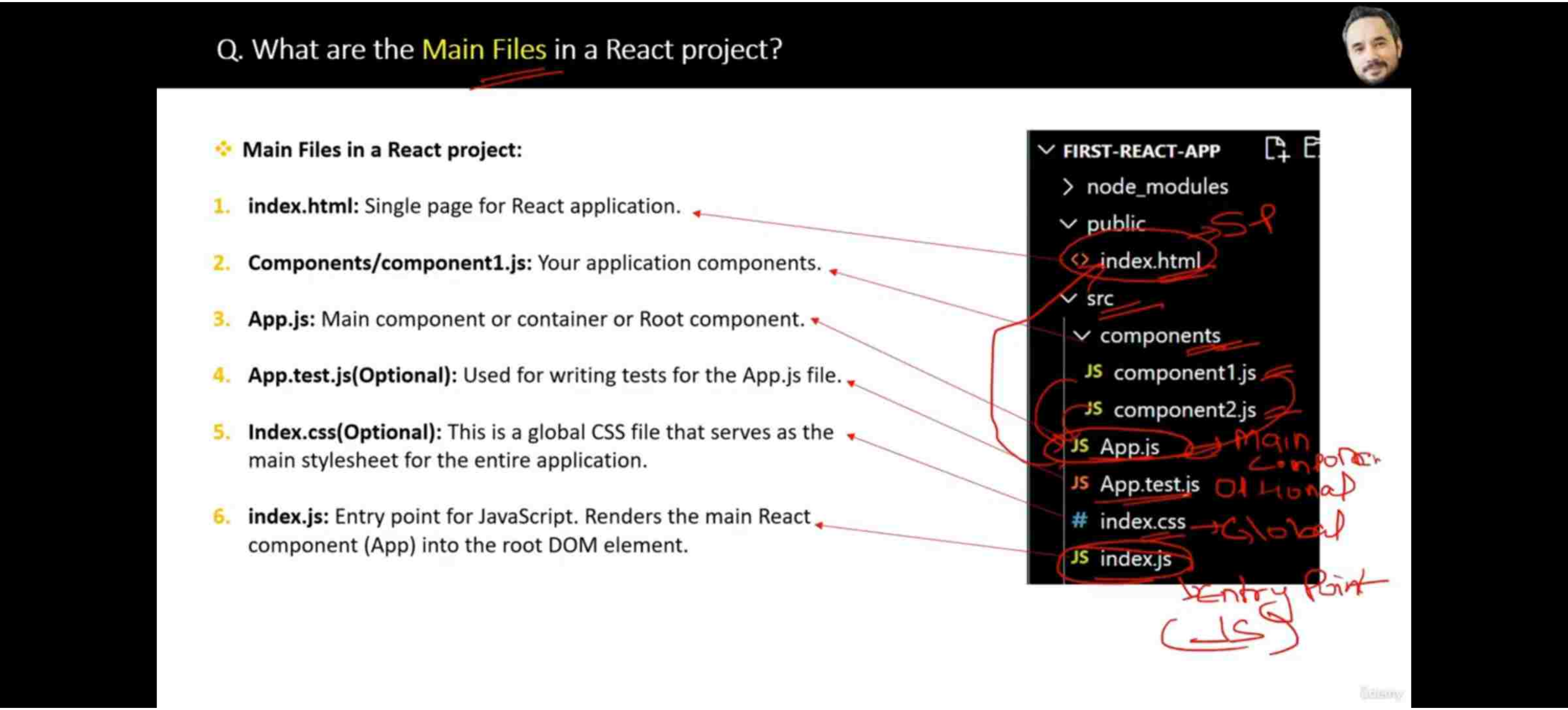
Q. What is Arrow Function Expression in JSX?

- The arrow function expression syntax is a concise way of defining functions.

```
// Regular Function Declaration
function AppFunc(props) {
  return (
    <div>
      <h1>{props.name}</h1>
    </div>
  );
}
export default AppFunc;
```

=

```
// Arrow Function Expression
const ArrowFunc = (props) => {
  return (
    <div>
      <h1>{props.name}</h1>
    </div>
  );
};
export default ArrowFunc;
```

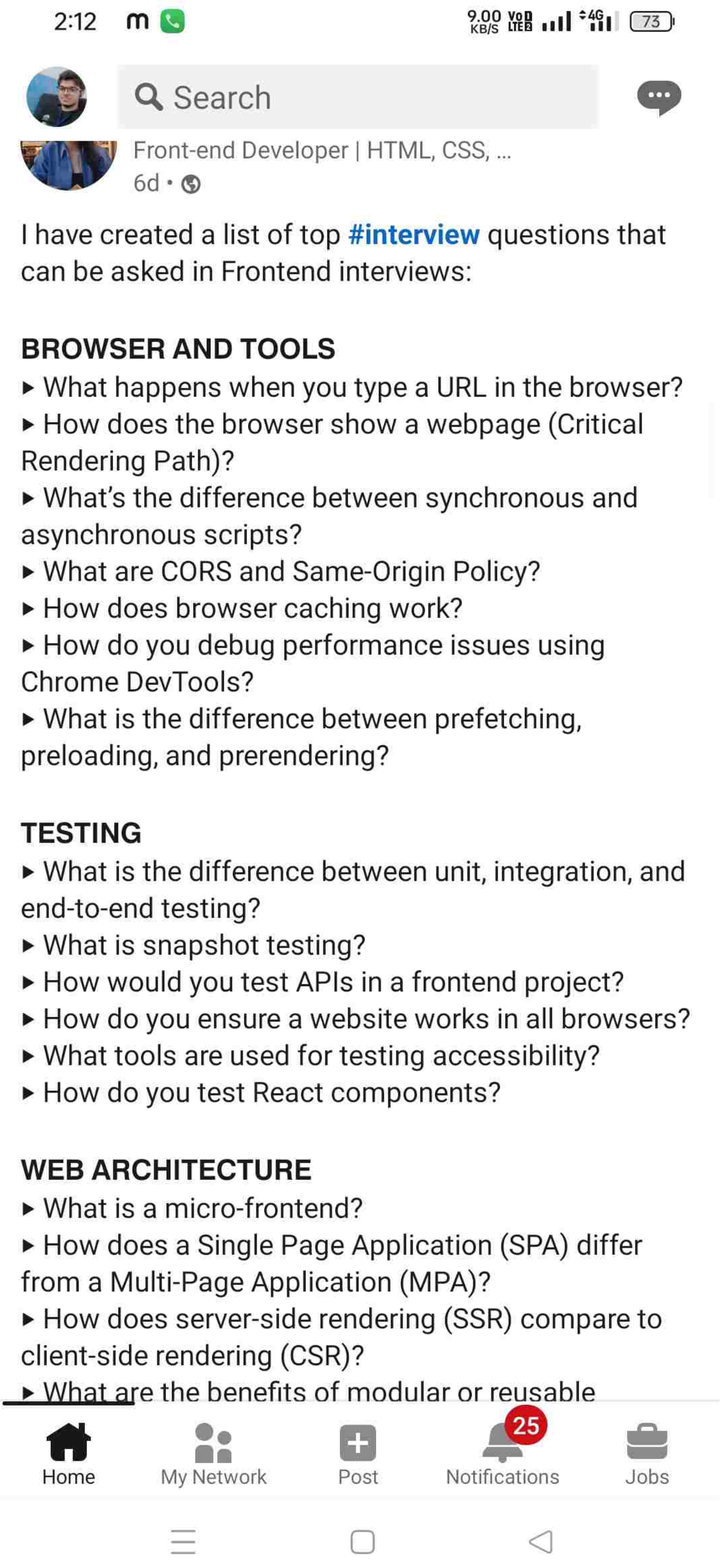


Q. What are the Main Files in a React project?

Main Files in a React project:

1. **index.html:** Single page for React application.
2. **Components/component1.js:** Your application components.
3. **App.js:** Main component or container or Root component.
4. **App.test.js(Optional):** Used for writing tests for the App.js file.
5. **Index.css(Optional):** This is a global CSS file that serves as the main stylesheet for the entire application.
6. **index.js:** Entry point for JavaScript. Renders the main React component (App) into the root DOM element.

```
FIRST-REACT-APP
  > node_modules
  v public
     <> index.html
  v src
     v components
        JS component1.js
        JS component2.js
     JS App.js
     JS App.test.js
     #  index.css
     JS index.js
```



Front-end Developer | HTML, CSS, ...
6d •

I have created a list of top #interview questions that can be asked in Frontend interviews:

BROWSER AND TOOLS

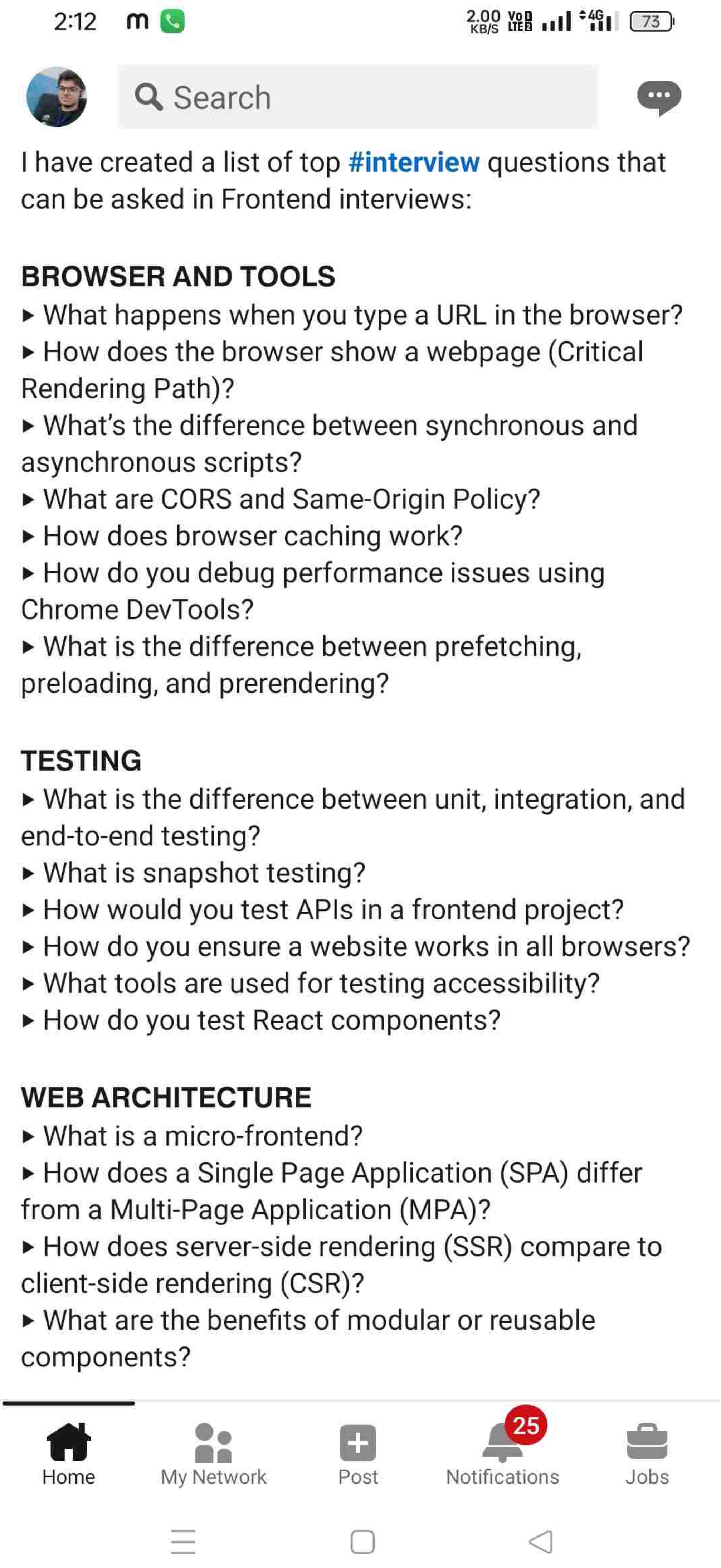
▸ What happens when you type a URL in the browser?
▸ How does the browser show a webpage (Critical Rendering Path)?
▸ What's the difference between synchronous and asynchronous scripts?
▸ What are CORS and Same-Origin Policy?
▸ How does browser caching work?
▸ How do you debug performance issues using Chrome DevTools?
▸ What is the difference between prefetching, preloading, and prerendering?

TESTING

▸ What is the difference between unit, integration, and end-to-end testing?
▸ What is snapshot testing?
▸ How would you test APIs in a frontend project?
▸ How do you ensure a website works in all browsers?
▸ What tools are used for testing accessibility?
▸ How do you test React components?

WEB ARCHITECTURE

▸ What is a micro-frontend?
▸ How does a Single Page Application (SPA) differ from a Multi-Page Application (MPA)?
▸ How does server-side rendering (SSR) compare to client-side rendering (CSR)?
▸ What are the benefits of modular or reusable



I have created a list of top #interview questions that can be asked in Frontend interviews:

BROWSER AND TOOLS

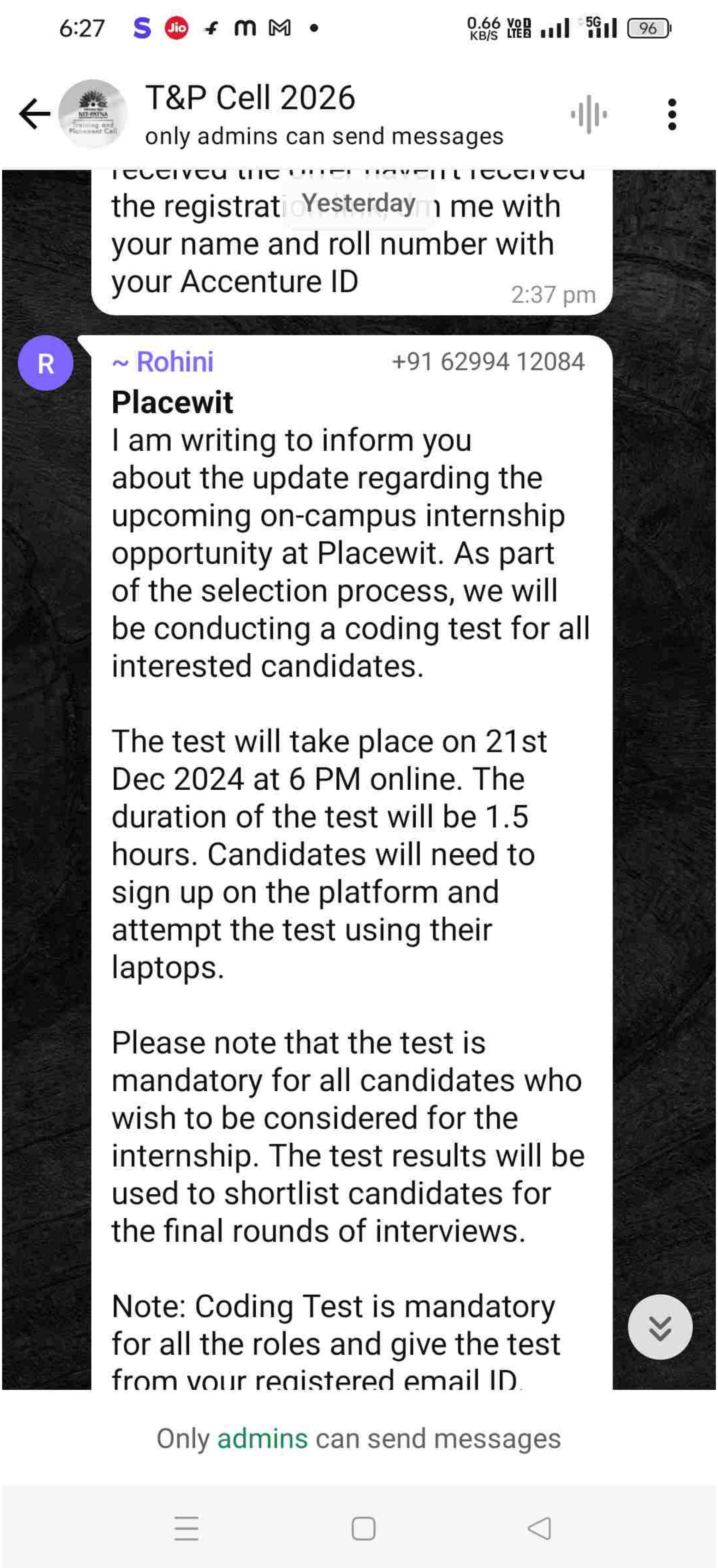
▸ What happens when you type a URL in the browser?
▸ How does the browser show a webpage (Critical Rendering Path)?
▸ What's the difference between synchronous and asynchronous scripts?
▸ What are CORS and Same-Origin Policy?
▸ How does browser caching work?
▸ How do you debug performance issues using Chrome DevTools?
▸ What is the difference between prefetching, preloading, and prerendering?

TESTING

▸ What is the difference between unit, integration, and end-to-end testing?
▸ What is snapshot testing?
▸ How would you test APIs in a frontend project?
▸ How do you ensure a website works in all browsers?
▸ What tools are used for testing accessibility?
▸ How do you test React components?

WEB ARCHITECTURE

▸ What is a micro-frontend?
▸ How does a Single Page Application (SPA) differ from a Multi-Page Application (MPA)?
▸ How does server-side rendering (SSR) compare to client-side rendering (CSR)?
▸ What are the benefits of modular or reusable components?



T&P Cell 2026

only admins can send messages

the registrati Yesterday n me with your name and roll number with your Accenture ID

2:37 pm

R

~ Rohini +91 62994 12084

Placewit

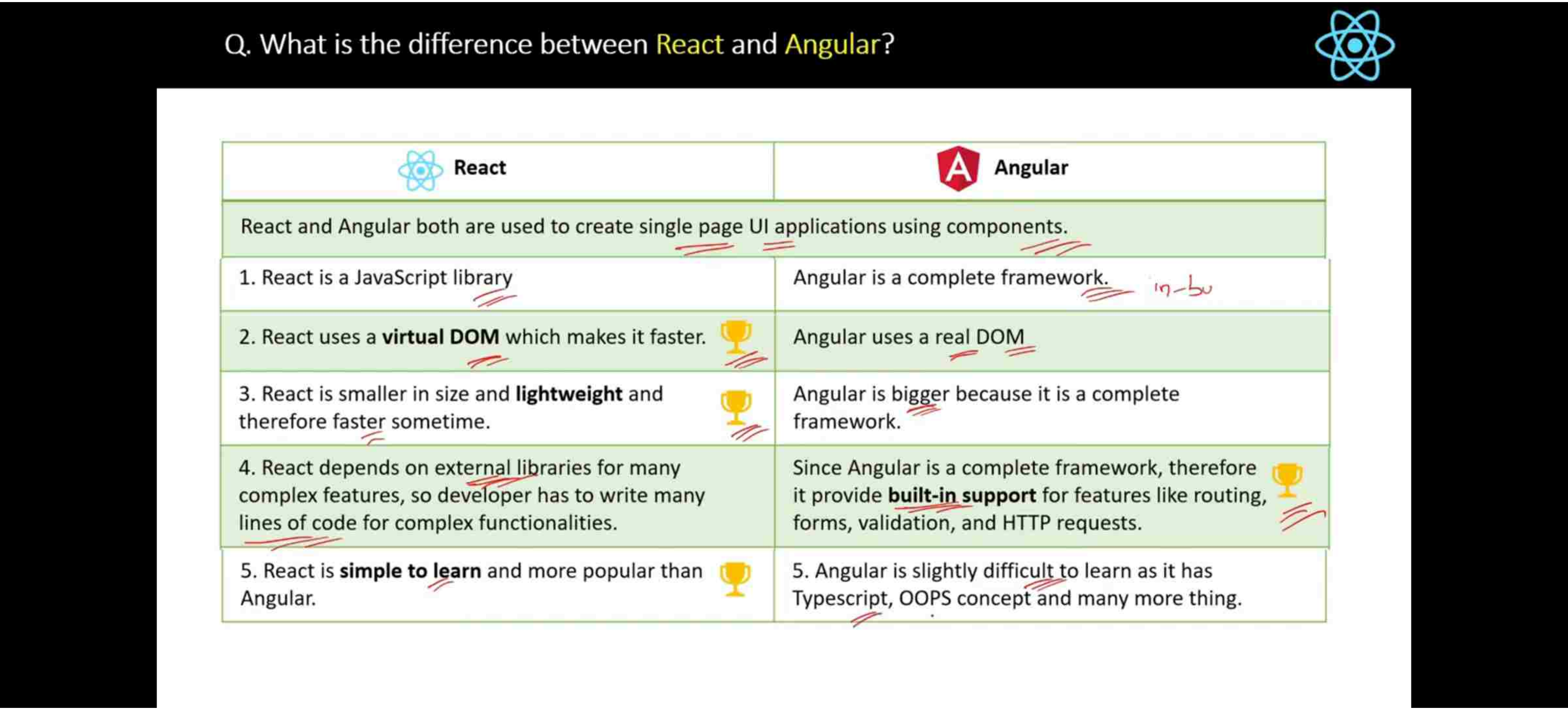
I am writing to inform you about the update regarding the upcoming on-campus internship opportunity at Placewit. As part of the selection process, we will be conducting a coding test for all interested candidates.

The test will take place on 21st Dec 2024 at 6 PM online. The duration of the test will be 1.5 hours. Candidates will need to sign up on the platform and attempt the test using their laptops.

Please note that the test is mandatory for all candidates who wish to be considered for the internship. The test results will be used to shortlist candidates for the final rounds of interviews.

Note: Coding Test is mandatory for all the roles and give the test from your registered email ID.

Only admins can send messages

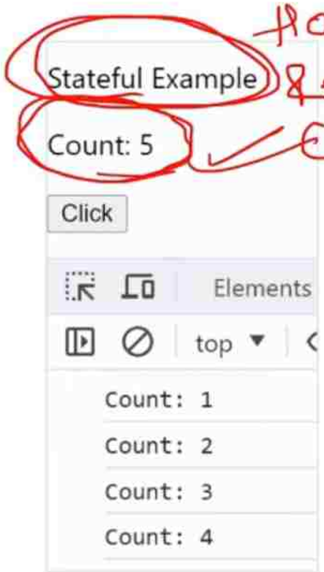


Q. What is the difference between React and Angular?

React	Angular
React and Angular both are used to create single page UI applications using components.	
1. React is a JavaScript library	Angular is a complete framework. in-bu
2. React uses a **virtual DOM** which makes it faster. 🏆	Angular uses a real DOM
3. React is smaller in size and **lightweight** and therefore faster sometime. 🏆	Angular is bigger because it is a complete framework.
4. React depends on external libraries for many complex features, so developer has to write many lines of code for complex functionalities.	Since Angular is a complete framework, therefore it provide **built-in support** for features like routing, forms, validation, and HTTP requests. 🏆
5. React is **simple to learn** and more popular than Angular. 🏆	5. Angular is slightly difficult to learn as it has Typescript, OOPS concept and many more thing.

Q. What are state, stateless, stateful and state management terms?

- "state" refers to the current data of the component.
- Stateful or state management means, when a user performs some actions on the UI, then the React application should be able to **update and re-render that data or state** on the UI.

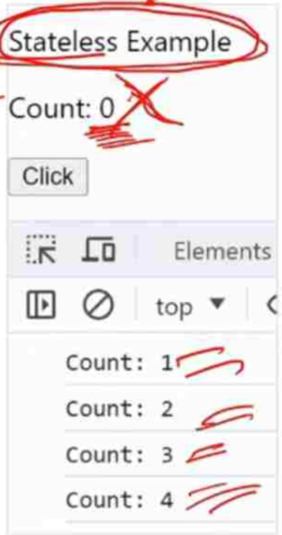


Stateful Example

Count: 5

```
Count: 1
Count: 2
Count: 3
Count: 4
```

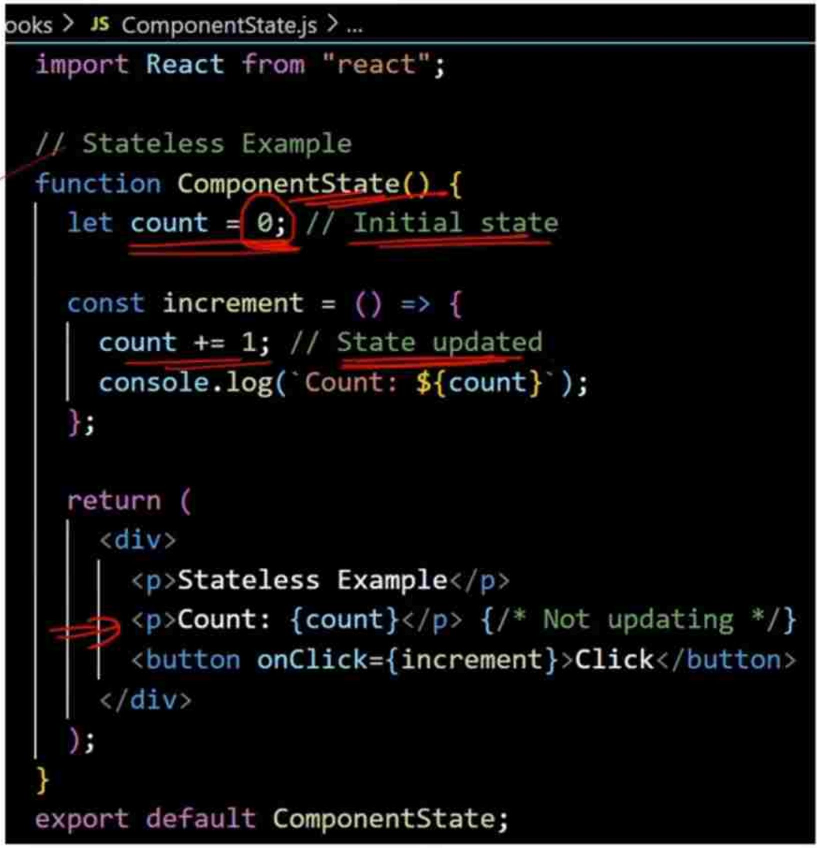
Hooks
Redux
Lifecycle



Stateless Example

Count: 0

```
Count: 1
Count: 2
Count: 3
Count: 4
```

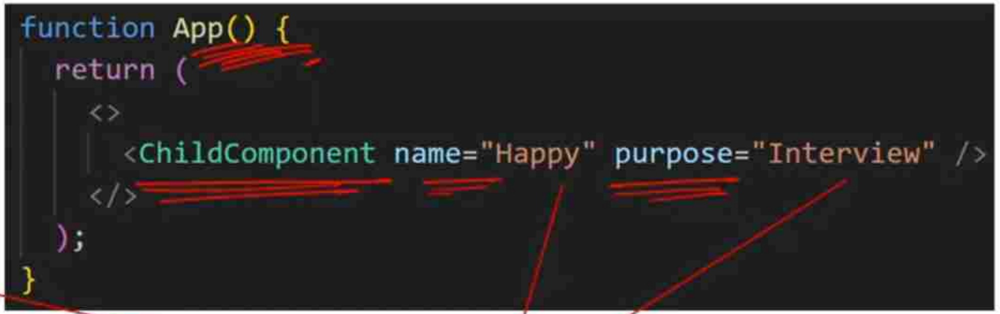


ooks > JS ComponentState.js > ...

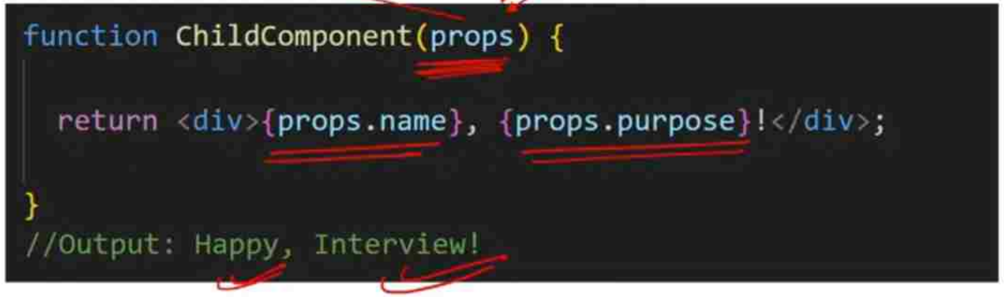
```
import React from "react";

// Stateless Example
function ComponentState() {
  let count = 0; // Initial state

  const increment = () => {
    count += 1; // State updated
    console.log(`Count: ${count}`);
  };

  return (
    <div>
      <p>Stateless Example</p>
      <p>Count: {count}</p> {/* Not updating */}
      <button onClick={increment}>Click</button>
    </div>
  );
}
export default ComponentState;
```

Q. What are Props n JSX? **V. IMP.**

- props (properties) are a way to **pass data** from a parent component to a child component.



```
function App() {
  return (
    <>
      <ChildComponent name="Happy" purpose="Interview" />
    </>
  );
}
```

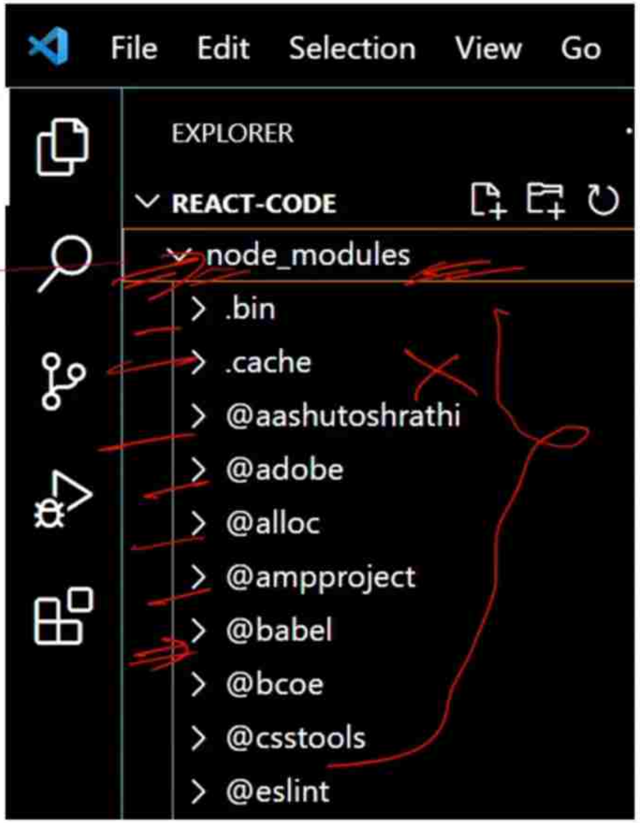


```
function ChildComponent(props) {

  return <div>{props.name}, {props.purpose}!</div>;

}
//Output: Happy, Interview!
```

Q. What is NPM? What is the role of node_modules folder? V. IMP.

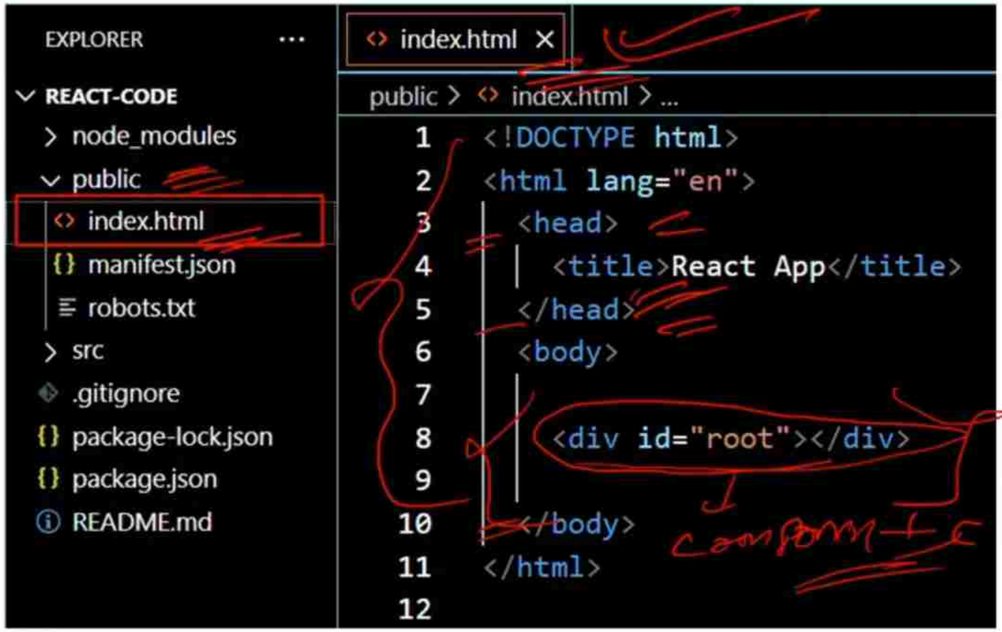
- NPM(Node Package Manager) is used to manage the **dependencies** for your React project, including the **React library** itself.

- node_modules folder contains all the dependencies of the project, including the React libraries.



Q. What is the role of index.html page in React?

- index.html file is the main HTML file(SPA) in React applcication.
- Here the div with “id=root” will be replaced by the component inside index.js file.

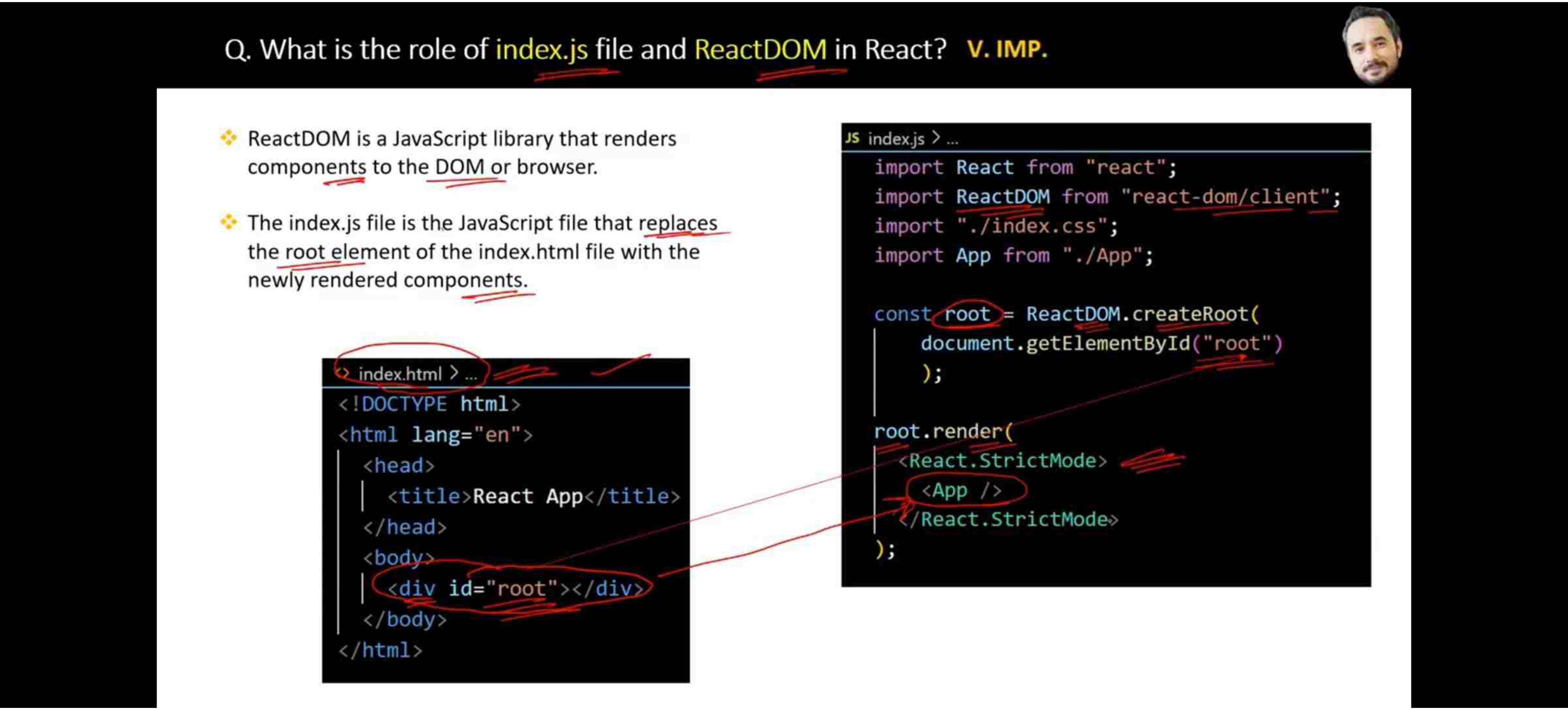


```
EXPLORER
∨ REACT-CODE
  > node_modules
  ∨ public
    index.html
    manifest.json
    robots.txt
  > src
  .gitignore
  package-lock.json
  package.json
  README.md
```

index.html ×

public > index.html > ...

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>React App</title>
  </head>
  <body>

    <div id="root"></div>

  </body>
</html>

```



Q. What is the role of index.js file and ReactDOM in React? V. IMP.

- ReactDOM is a JavaScript library that renders components to the DOM or browser.
- The index.js file is the JavaScript file that replaces the root element of the index.html file with the newly rendered components.

index.html > ...

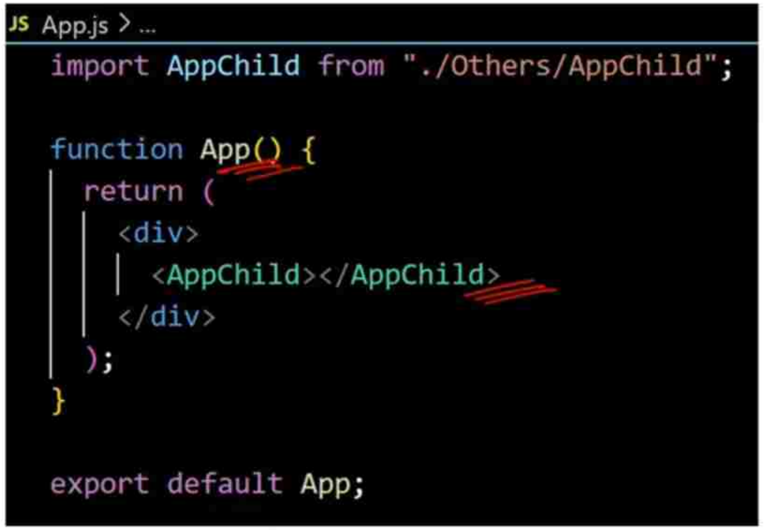
```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>React App</title>
  </head>
  <body>
    <div id="root"></div>
  </body>
</html>
```

index.js > ...

```javascript
import React from "react";
import ReactDOM from "react-dom/client";
import "./index.css";
import App from "./App";

const root = ReactDOM.createRoot(
    document.getElementById("root")
    );

root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

Q. What is the role of App.js file in React? V. IMP.

- App.js file contain the **root component(App)** of React application.
- App component is like a **container** for other components.
- App.js defines the structure, layout, and routing in the application.

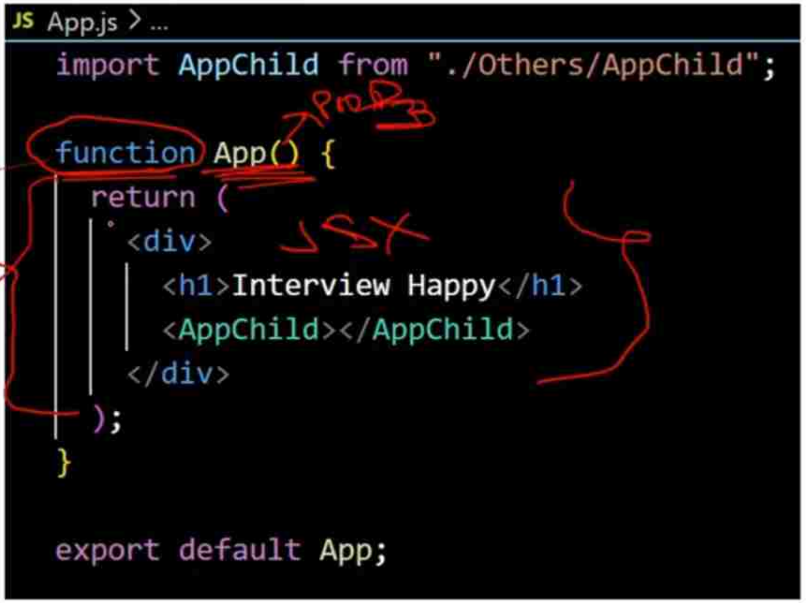


JS App.js > ...

```
import AppChild from "./Others/AppChild";

function App() {
  return (
    <div>
      <AppChild></AppChild>
    </div>
  );
}

export default App;
```

Q. What is the role of function and return inside App.js?

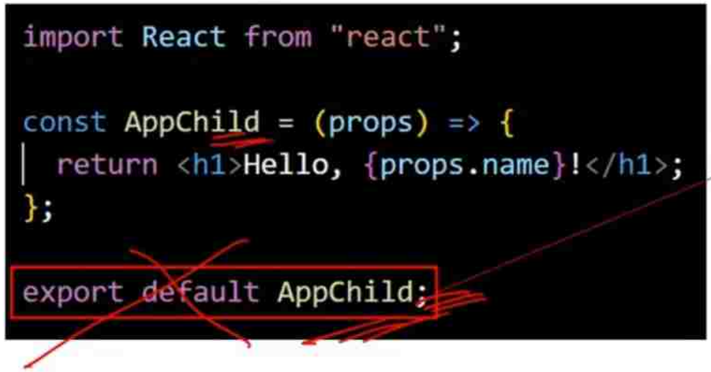
- The function keyword is used to **define a JavaScript function** that represents your React component.
- function is like a placeholder which contains all the code or logic of component.
- The function takes in props as its argument (if needed) and returns JSX



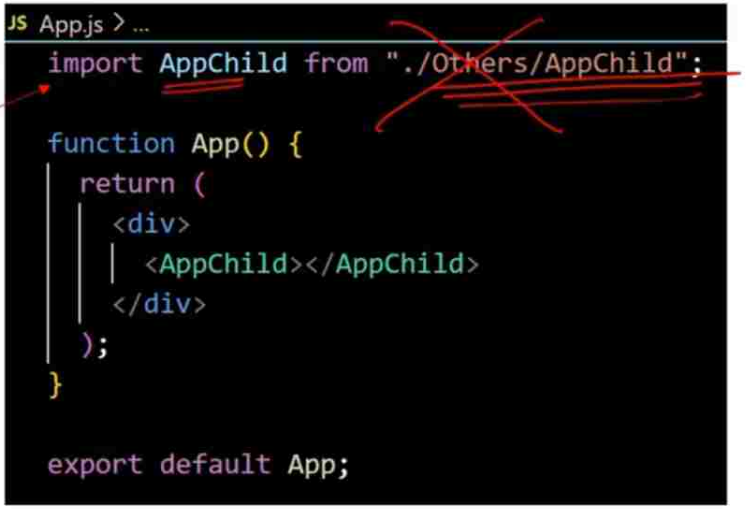
```
import AppChild from "./Others/AppChild";

function App() {
  return (
    <div>
      <h1>Interview Happy</h1>
      <AppChild></AppChild>
    </div>
  );
}

export default App;
```

Q. What is the role of export default inside App.js?

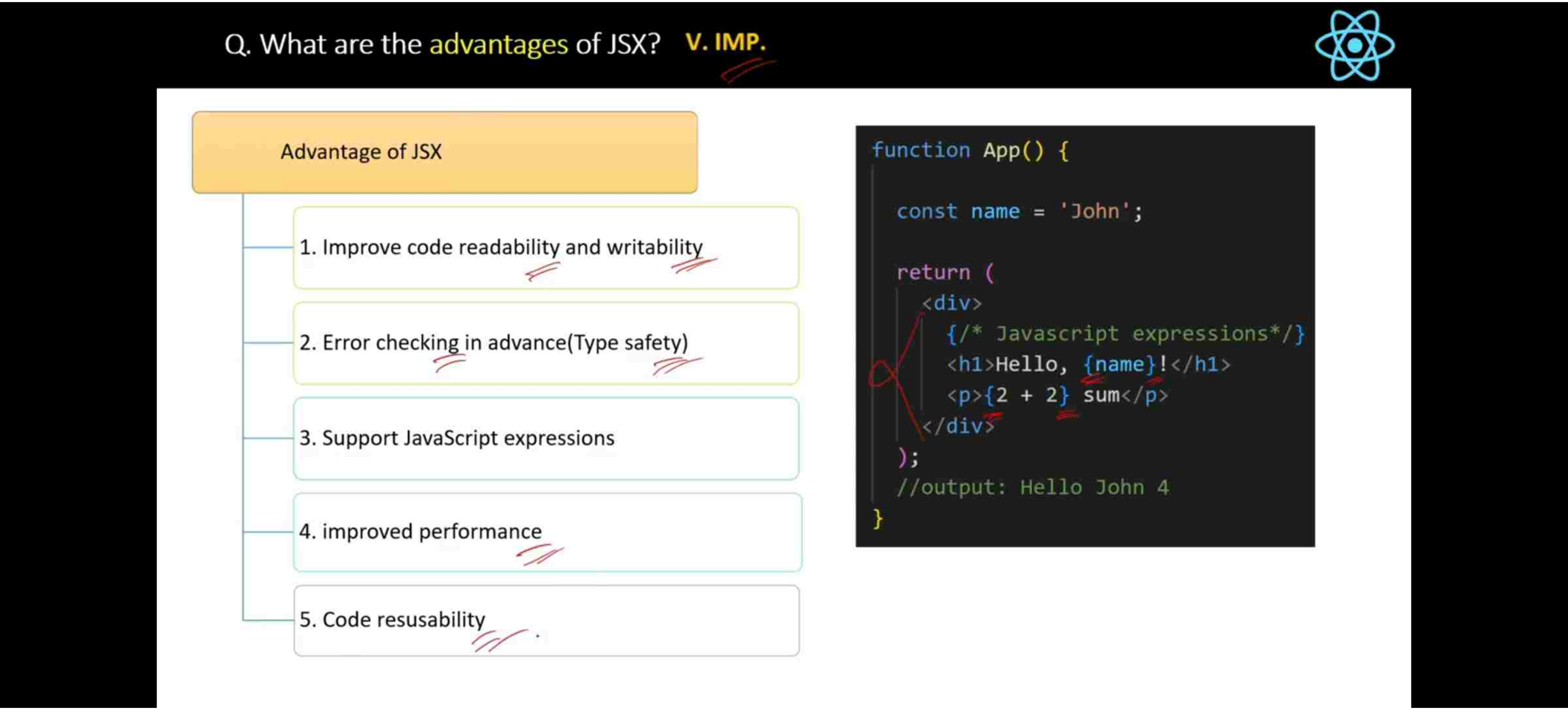
- Export statement is used to make a component available for import using “import” statement in other files.



```
import React from "react";

const AppChild = (props) => {
  return <h1>Hello, {props.name}!</h1>;
};

export default AppChild;
```



```
import AppChild from "./Others/AppChild";

function App() {
  return (
    <div>
      <AppChild></AppChild>
    </div>
  );
}

export default App;
```



Q. What are the advantages of JSX? V. IMP.

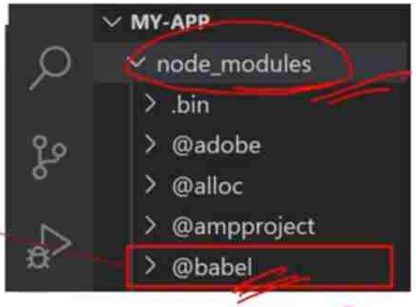
Advantage of JSX

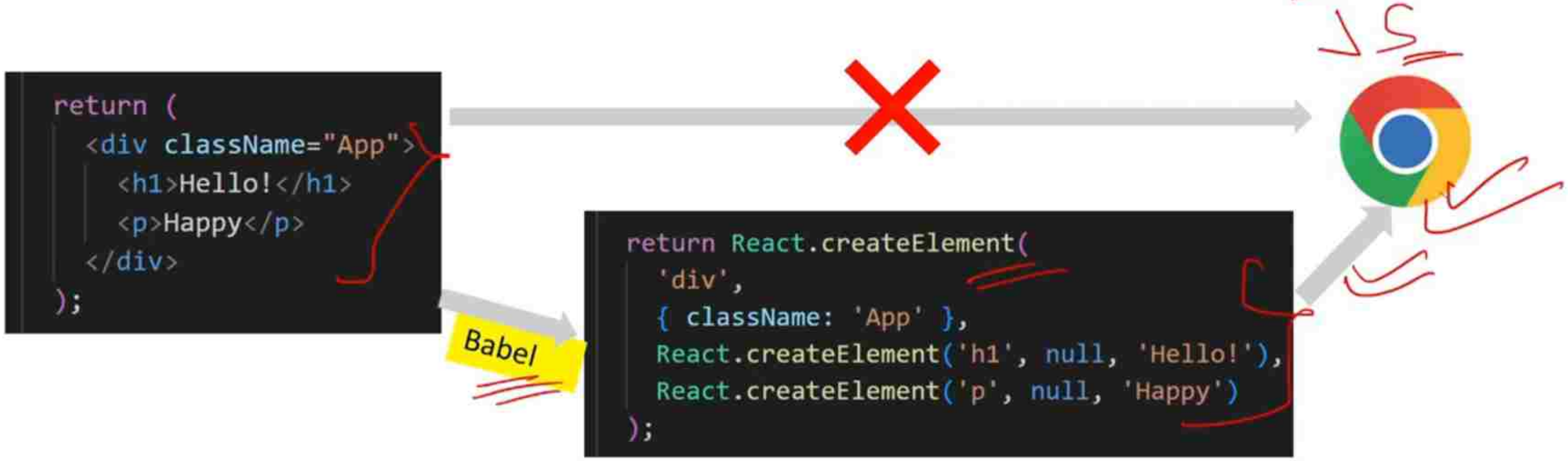
1. Improve code readability and writability
2. Error checking in advance(Type safety)
3. Support JavaScript expressions
4. improved performance
5. Code resusability

```
function App() {

  const name = 'John';

  return (
    <div>
      {/* Javascript expressions*/}
      <h1>Hello, {name}!</h1>
      <p>{2 + 2} sum</p>
    </div>
  );
  //output: Hello John 4
}
```

Q. What is Babel?

- Babel in React is used to transpile JSX syntax into regular JavaScript which browser can understand.



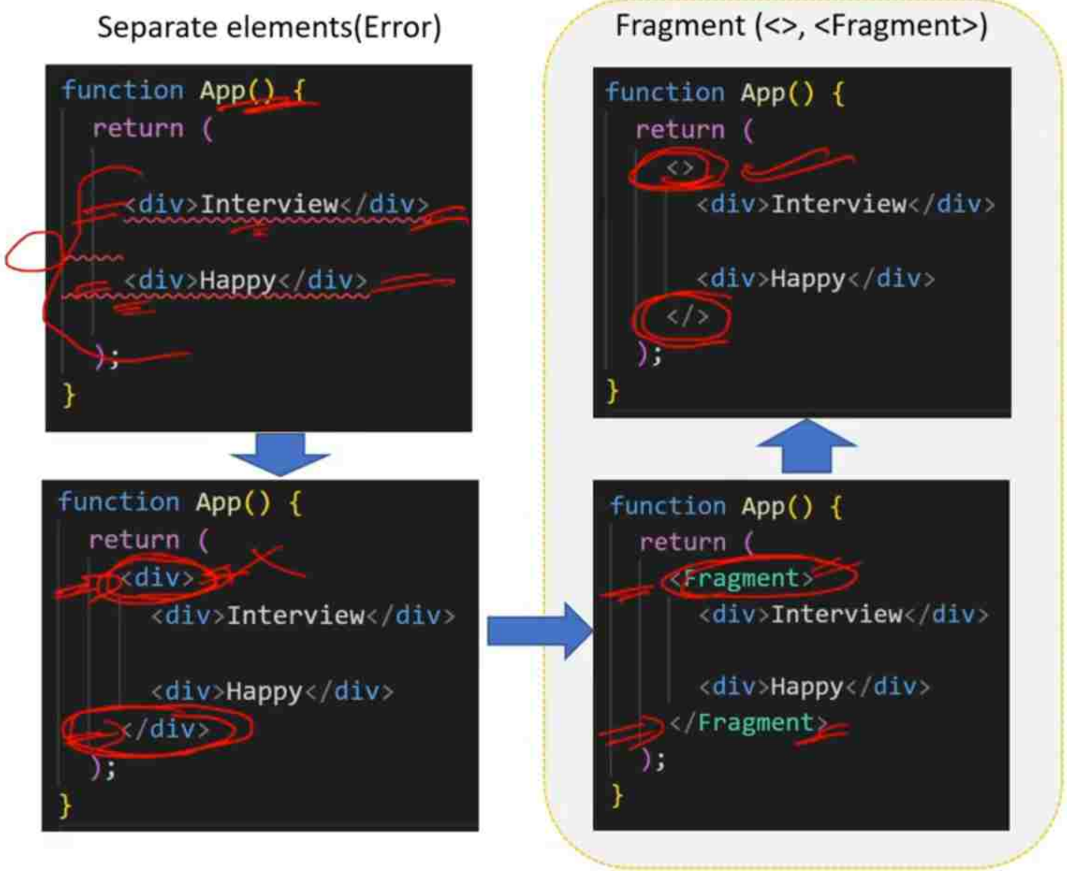


```
return (
  <div className="App">
    <h1>Hello!</h1>
    <p>Happy</p>
  </div>
);
```

```
return React.createElement(
  'div',
  { className: 'App' },
  React.createElement('h1', null, 'Hello!'),
  React.createElement('p', null, 'Happy')
);
```

Q. What is the role of Fragment in JSX? V. IMP.

- In React, a fragment is a way to **group multiple children's** elements.
- Using a Fragment prevents the addition of unnecessary nodes to the DOM.



Separate elements(Error)

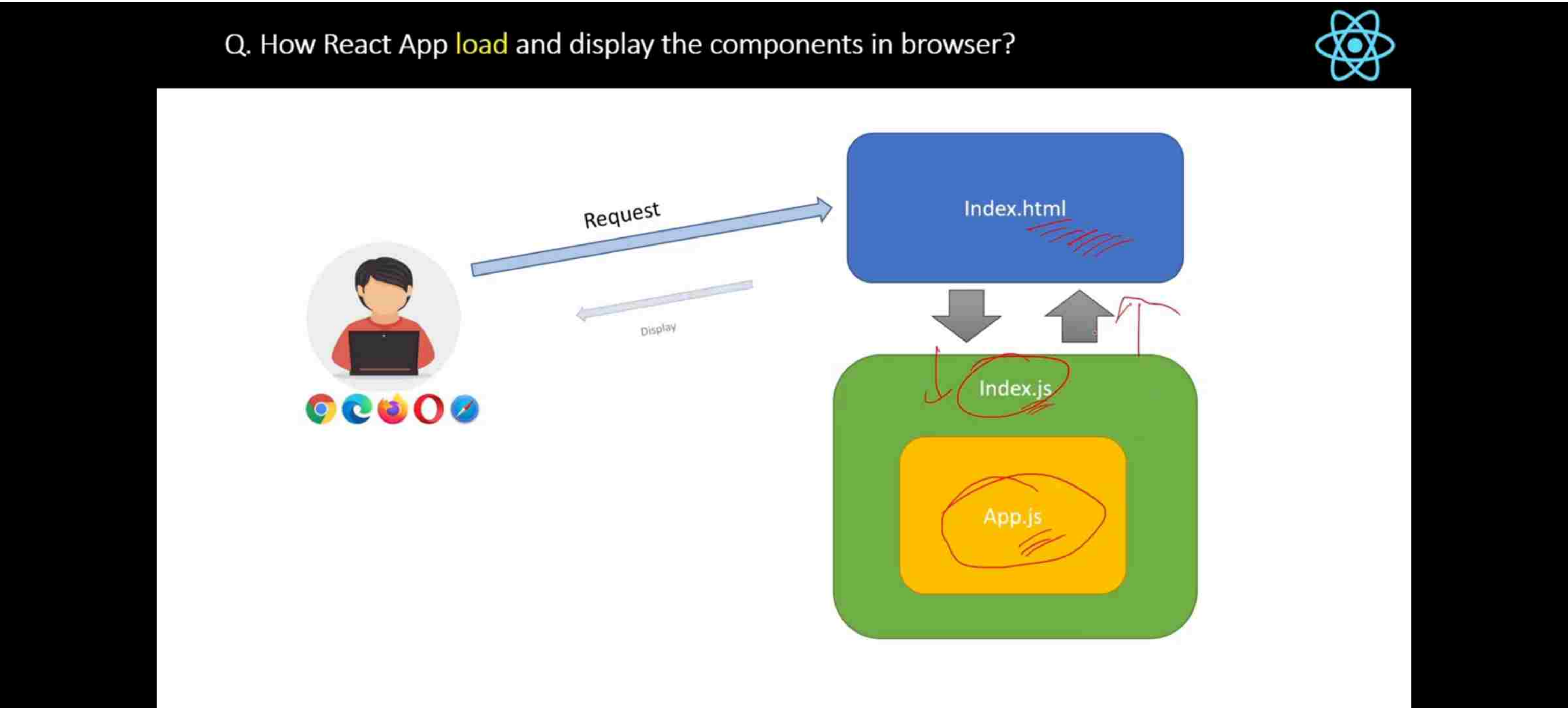
```
function App() {
  return (

      <div>Interview</div>

      <div>Happy</div>

  );
}
```

```
function App() {
  return (
    <div>
      <div>Interview</div>

      <div>Happy</div>
    </div>
  );
}
```

Fragment (<>, <Fragment>)

```
function App() {
  return (
    <Fragment>
      <div>Interview</div>

      <div>Happy</div>
    </Fragment>
  );
}
```

```
function App() {
  return (
    <>
      <div>Interview</div>

      <div>Happy</div>
    </>
  );
}
```



Q. How React App load and display the components in browser?

Request

Display

Index.html

Index.js

App.js

Q. What is the difference between Declarative & Imperative syntax?

1. Declarative syntax focuses on **describing the desired result** without specifying the step-by-step process.
2. **JSX** in React is used to write declarative syntax.

```
// Declarative syntax using JSX
function App() {

  return <h1>Interview Happy</h1>;

}
```

focus ⇒ output

1. Imperative syntax involves **step by step process** to achieve a particular goal.
2. **JavaScript** has an imperative syntax.

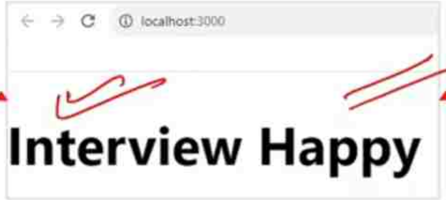
```
// Imperative syntax(non-React) using JavaScript
function App() {
  const element = document.createElement("h1");
  element.textContent = "Interview Happy";
  document.body.appendChild(element);
}
```

step

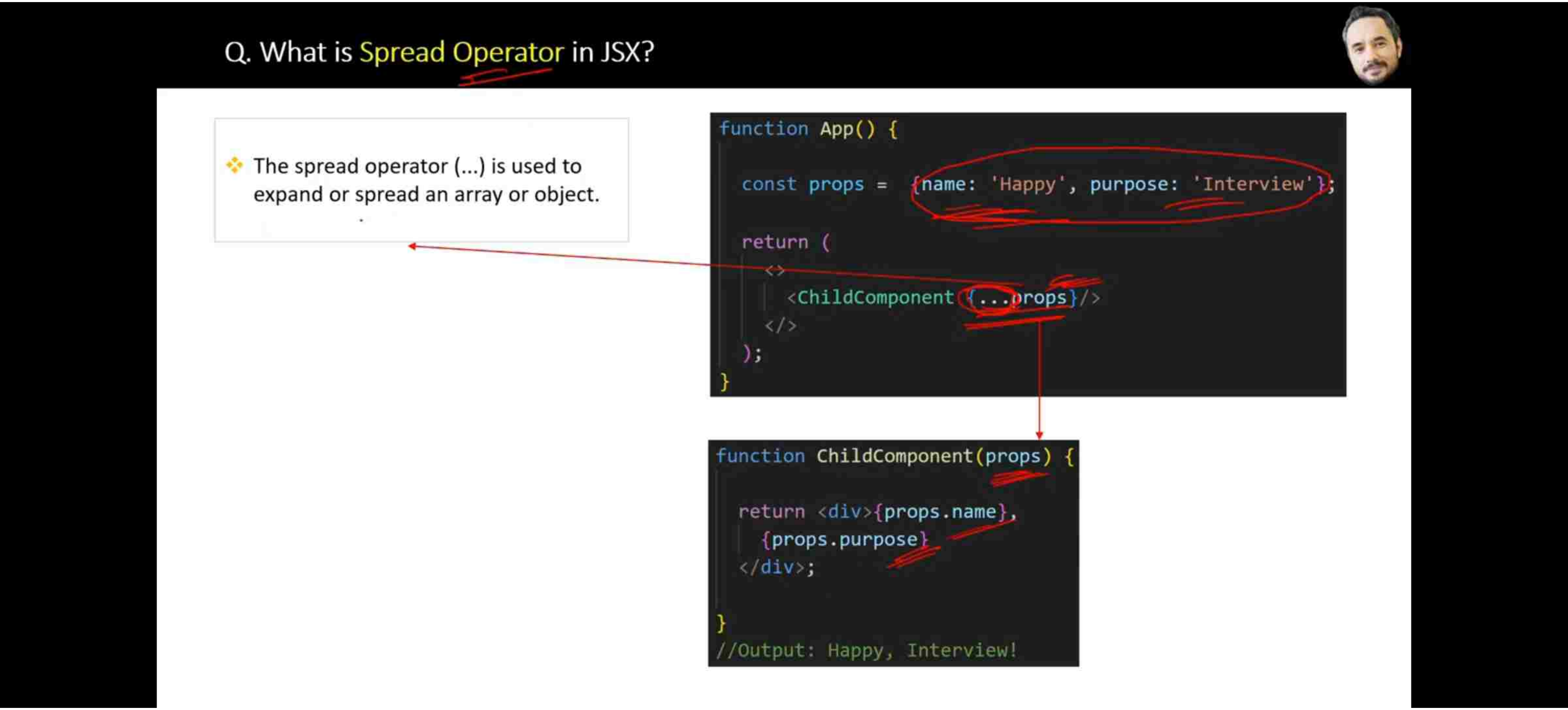
1

2

3



Interview Happy

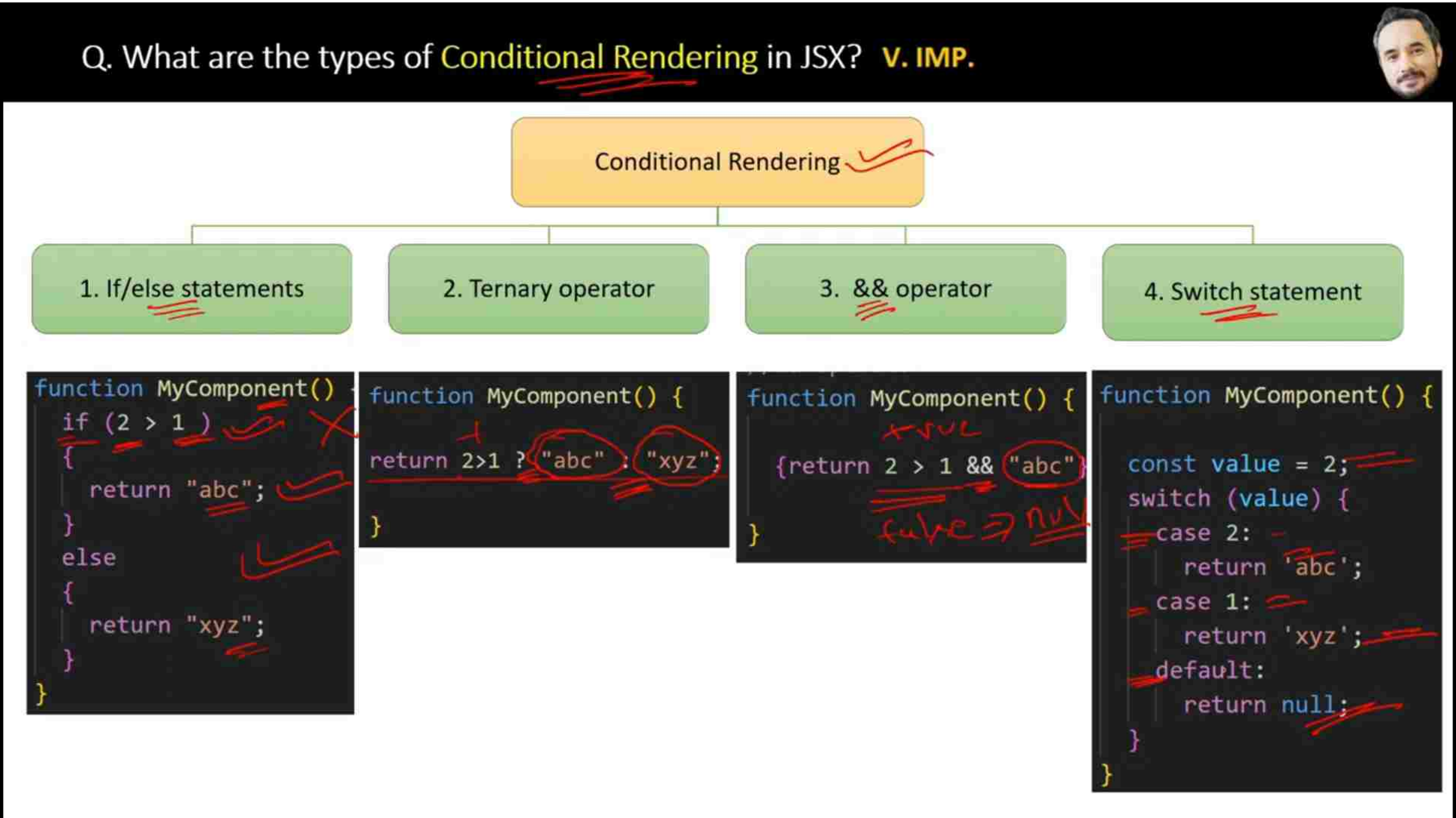


Q. What is Spread Operator in JSX?

- The spread operator (...) is used to expand or spread an array or object.

```
function App() {

  const props = {name: 'Happy', purpose: 'Interview'};

  return (
    <>
      <ChildComponent {...props}/>
    </>
  );
}
```

```
function ChildComponent(props) {

  return <div>{props.name},
    {props.purpose}
  </div>;

}
//Output: Happy, Interview!
```



Q. What are the types of Conditional Rendering in JSX? **V. IMP.**

Conditional Rendering

1. If/else statements

2. Ternary operator

3. && operator

4. Switch statement

```
function MyComponent() {
  if (2 > 1 )
  {
    return "abc";
  }
  else
  {
    return "xyz";
  }
}
```

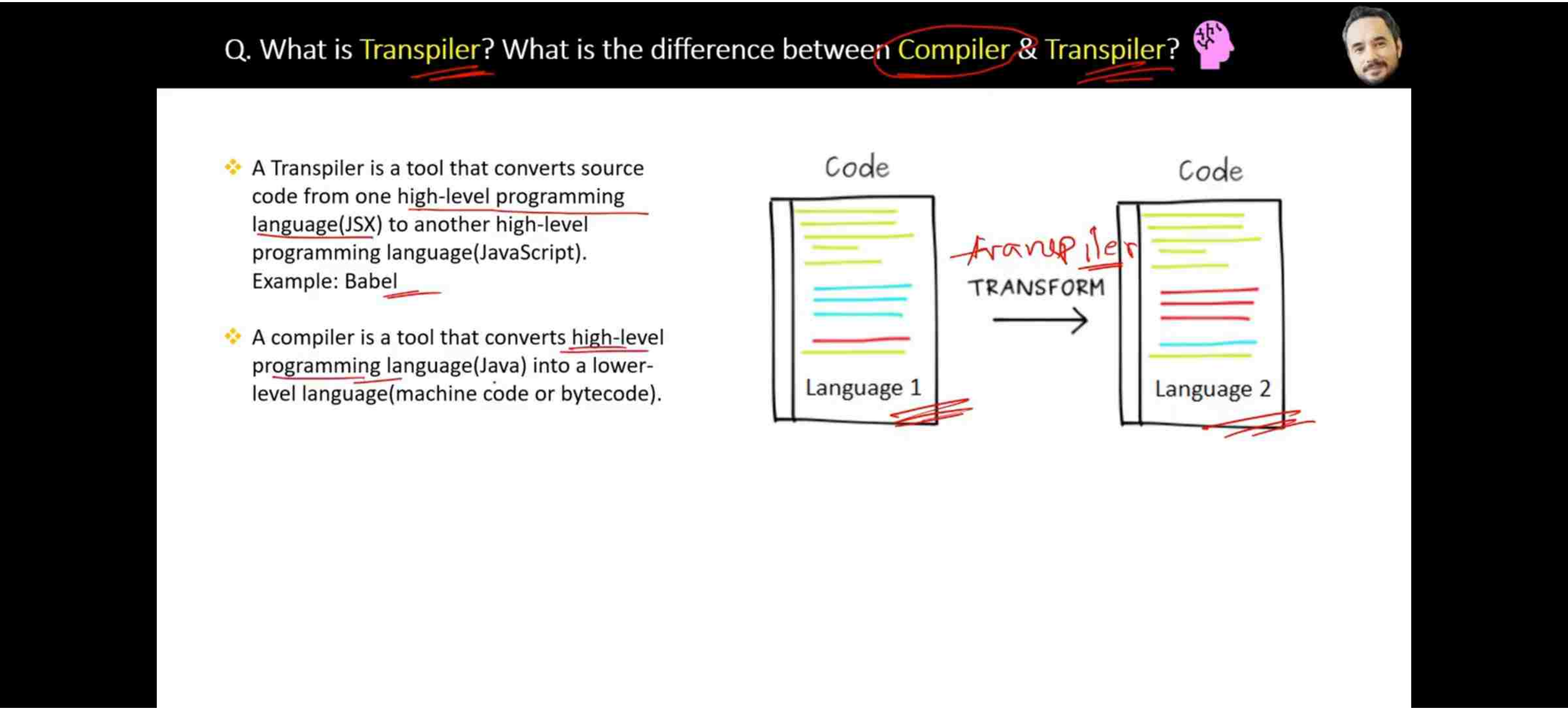
```
function MyComponent() {

return 2>1 ? "abc" : "xyz";

}
```

```
function MyComponent() {

  {return 2 > 1 && "abc"}

}
```

true

false ⇒ null

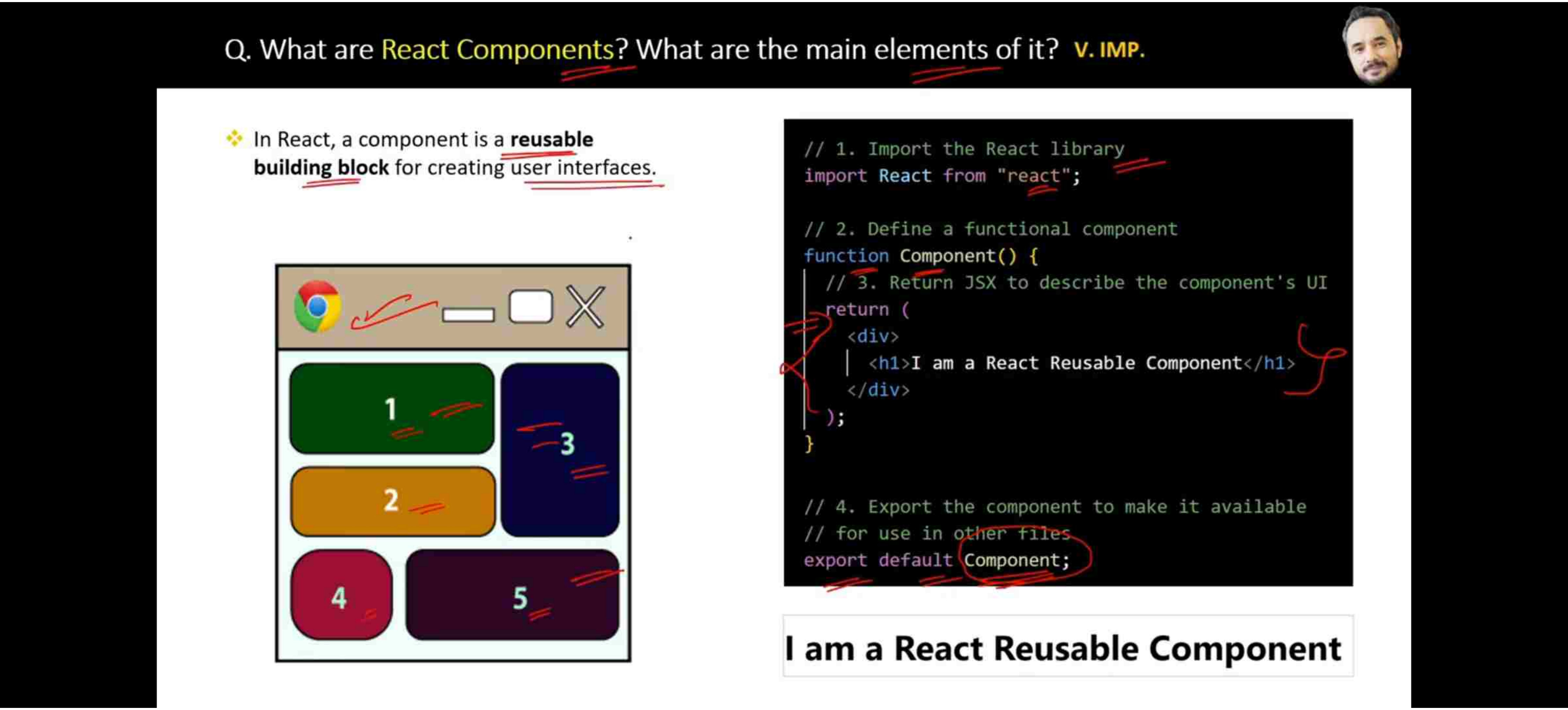
```
function MyComponent() {

  const value = 2;
  switch (value) {
    case 2:
      return 'abc';
    case 1:
      return 'xyz';
    default:
      return null;
  }
}
```



Q. What is Transpiler? What is the difference between Compiler & Transpiler?

- A Transpiler is a tool that converts source code from one high-level programming language(JSX) to another high-level programming language(JavaScript). Example: Babel

- A compiler is a tool that converts high-level programming language(Java) into a lower-level language(machine code or bytecode).

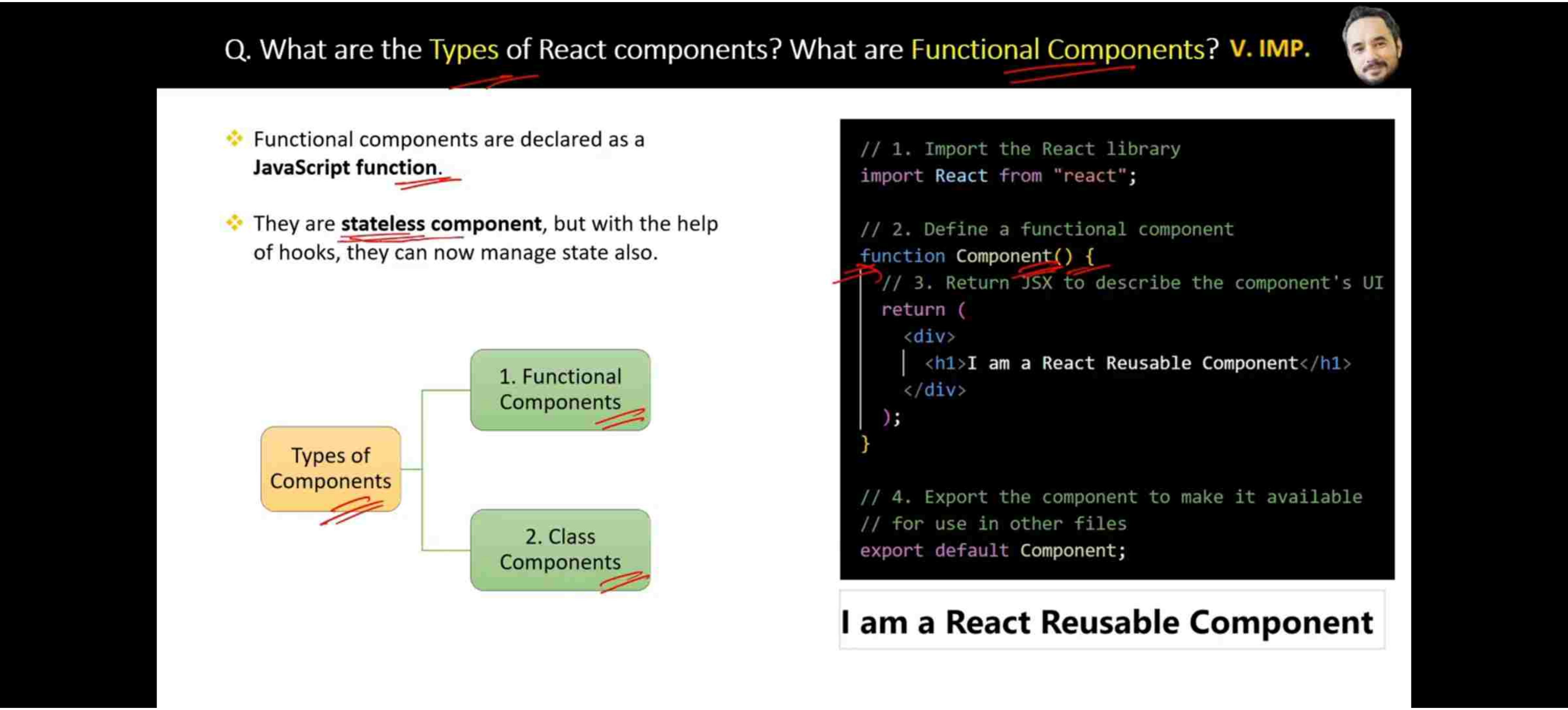


Q. What are React Components? What are the main elements of it? **V. IMP.**

- In React, a component is a **reusable building block** for creating user interfaces.

```
// 1. Import the React library
import React from "react";

// 2. Define a functional component
function Component() {
  // 3. Return JSX to describe the component's UI
  return (
    <div>
      <h1>I am a React Reusable Component</h1>
    </div>
  );
}


// 4. Export the component to make it available
// for use in other files
export default Component;
```

I am a React Reusable Component



Q. What are the Types of React components? What are Functional Components? V. IMP.

- Functional components are declared as a **JavaScript function**.
- They are **stateless component**, but with the help of hooks, they can now manage state also.

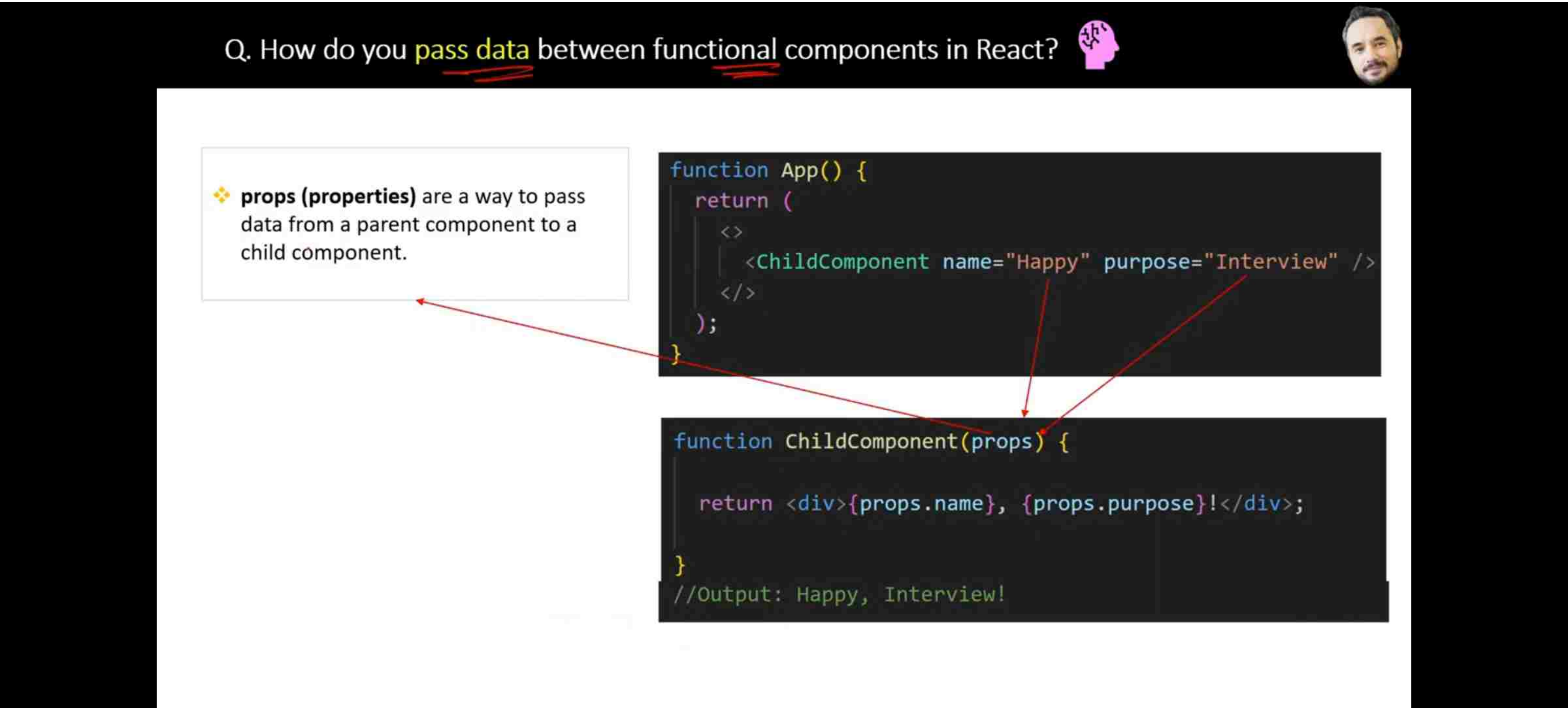
Types of Components

1. Functional Components

2. Class Components

```
// 1. Import the React library
import React from "react";

// 2. Define a functional component
function Component() {
  // 3. Return JSX to describe the component's UI
  return (
    <div>
      <h1>I am a React Reusable Component</h1>
    </div>
  );
}

// 4. Export the component to make it available
// for use in other files
export default Component;
```

I am a React Reusable Component

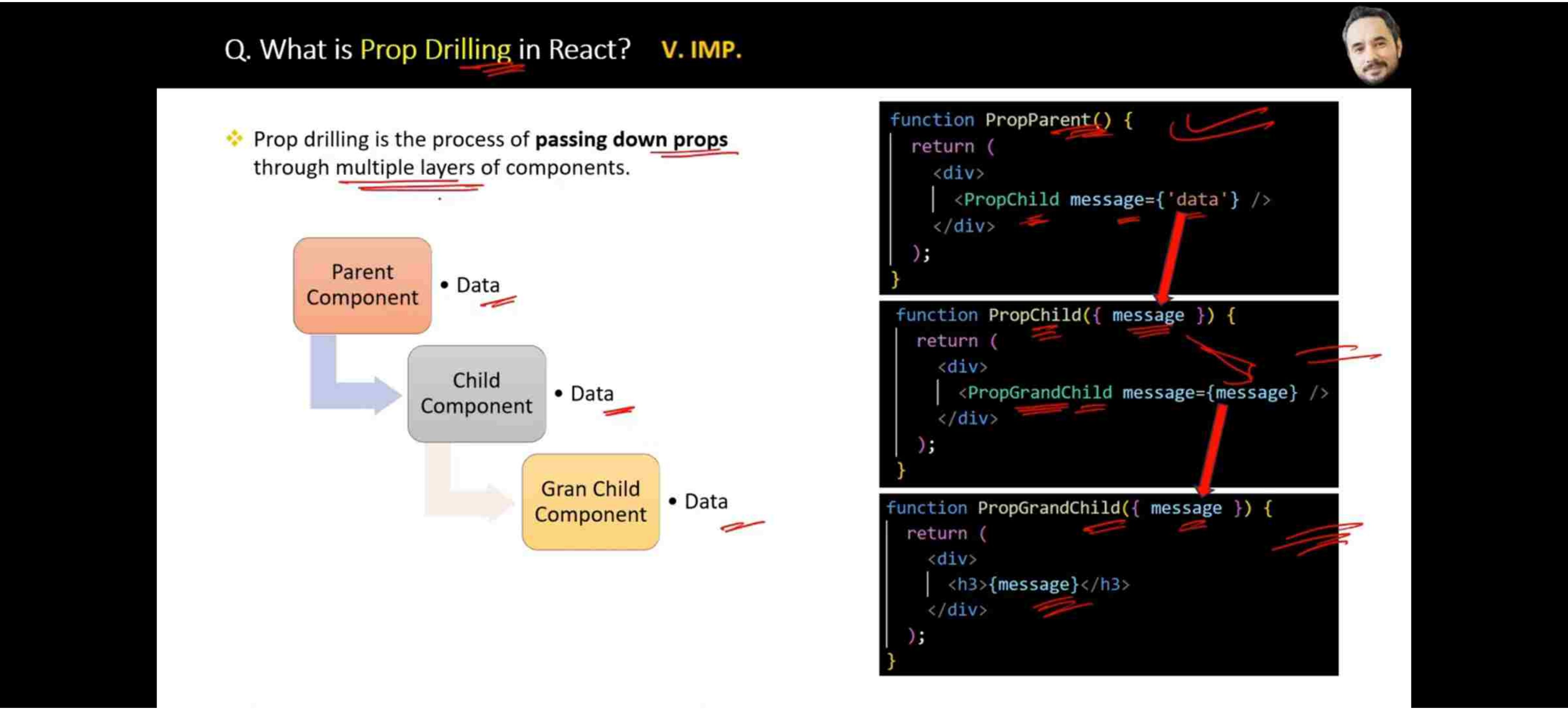


Q. How do you pass data between functional components in React?

- **props (properties)** are a way to pass data from a parent component to a child component.

```
function App() {
  return (
    <>
      <ChildComponent name="Happy" purpose="Interview" />
    </>
  );
}
```

```
function ChildComponent(props) {

  return <div>{props.name}, {props.purpose}!</div>;

}
//Output: Happy, Interview!
```



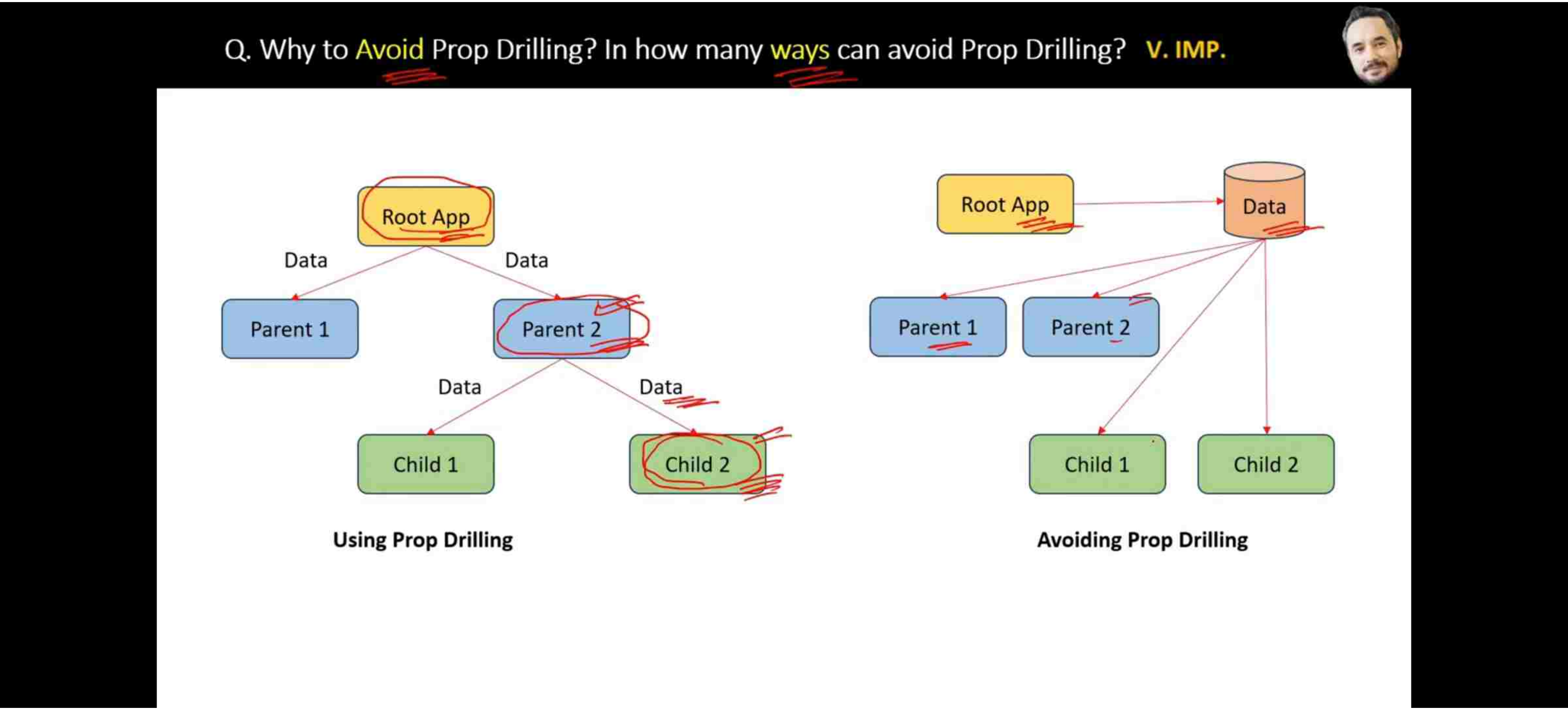
Q. What is Prop Drilling in React? V. IMP.

- Prop drilling is the process of **passing down props** through multiple layers of components.

```
function PropParent() {
  return (
    <div>
      <PropChild message={'data'} />
    </div>
  );
}
```

```
function PropChild({ message }) {
  return (
    <div>
      <PropGrandChild message={message} />
    </div>
  );
}
```

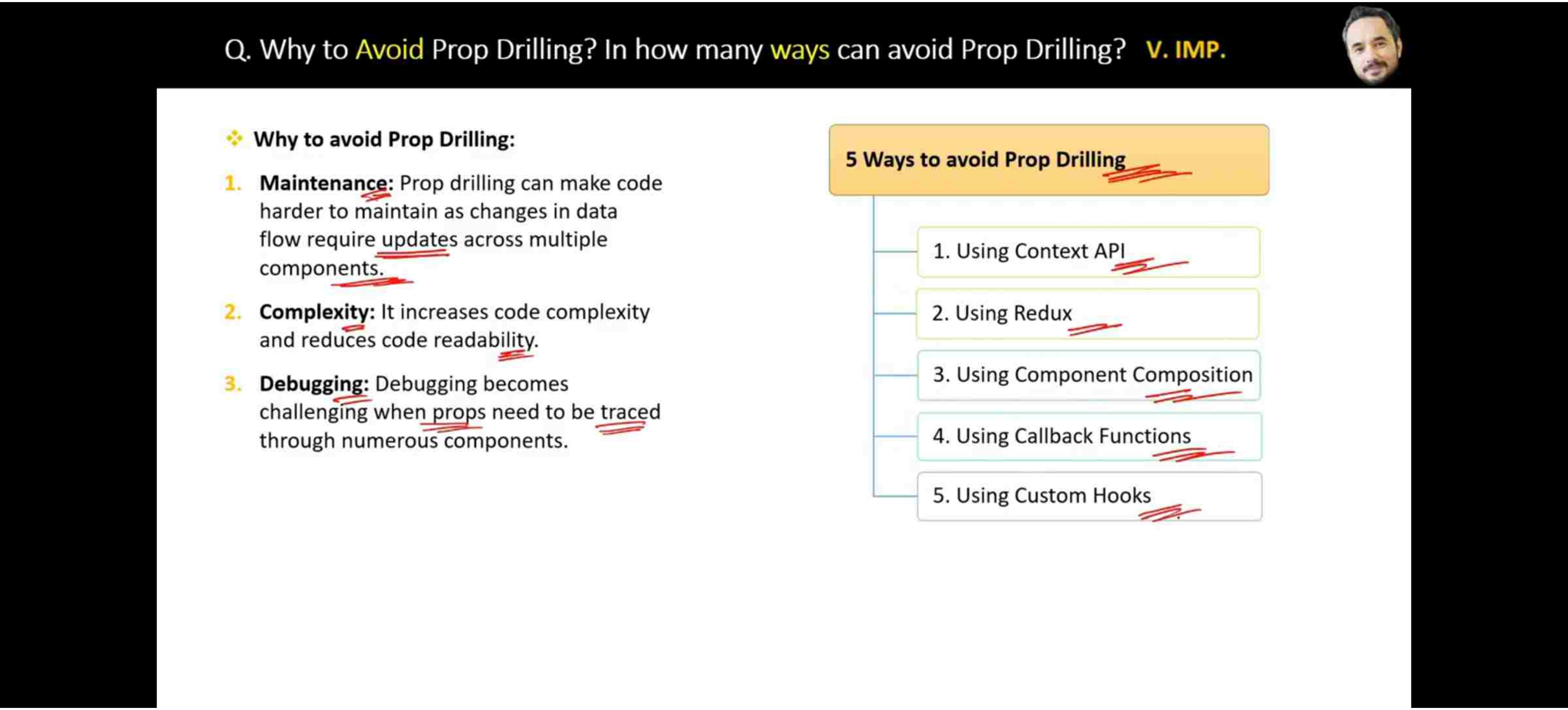
```
function PropGrandChild({ message }) {
  return (
    <div>
      <h3>{message}</h3>
    </div>
  );
}
```



Q. Why to Avoid Prop Drilling? In how many ways can avoid Prop Drilling? V. IMP.

Using Prop Drilling

Avoiding Prop Drilling



Q. Why to Avoid Prop Drilling? In how many ways can avoid Prop Drilling? V. IMP.

Why to avoid Prop Drilling:

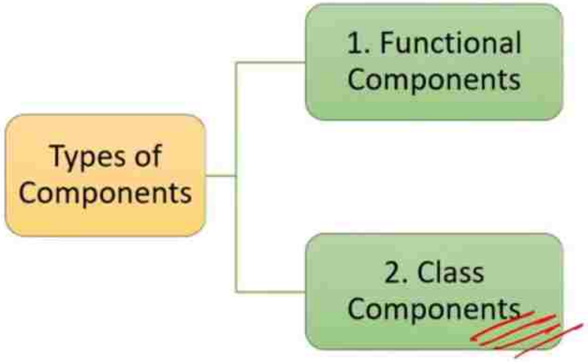
1. **Maintenance:** Prop drilling can make code harder to maintain as changes in data flow require updates across multiple components.
2. **Complexity:** It increases code complexity and reduces code readability.
3. **Debugging:** Debugging becomes challenging when props need to be traced through numerous components.

5 Ways to avoid Prop Drilling

1. Using Context API
2. Using Redux
3. Using Component Composition
4. Using Callback Functions
5. Using Custom Hooks

Q. What are Class Components In React? **V. IMP.**

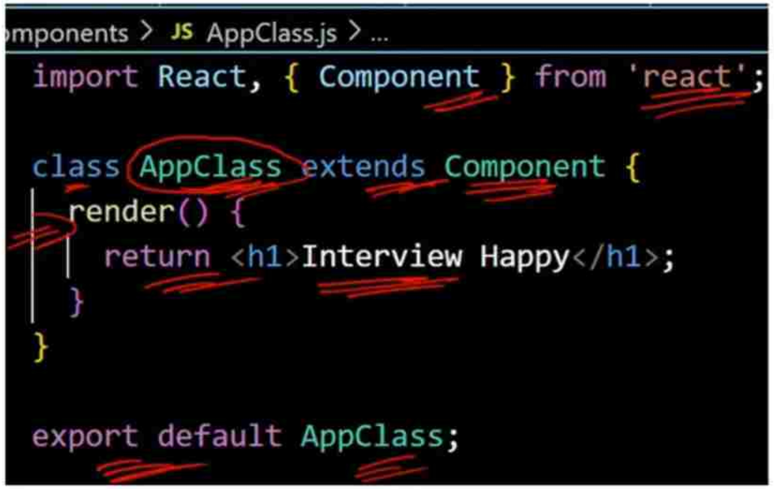
- Class components are defined using **JavaScript classes**.
- They are stateful components by using the lifecycle methods.
- The render method in a class component is responsible for returning JSX.



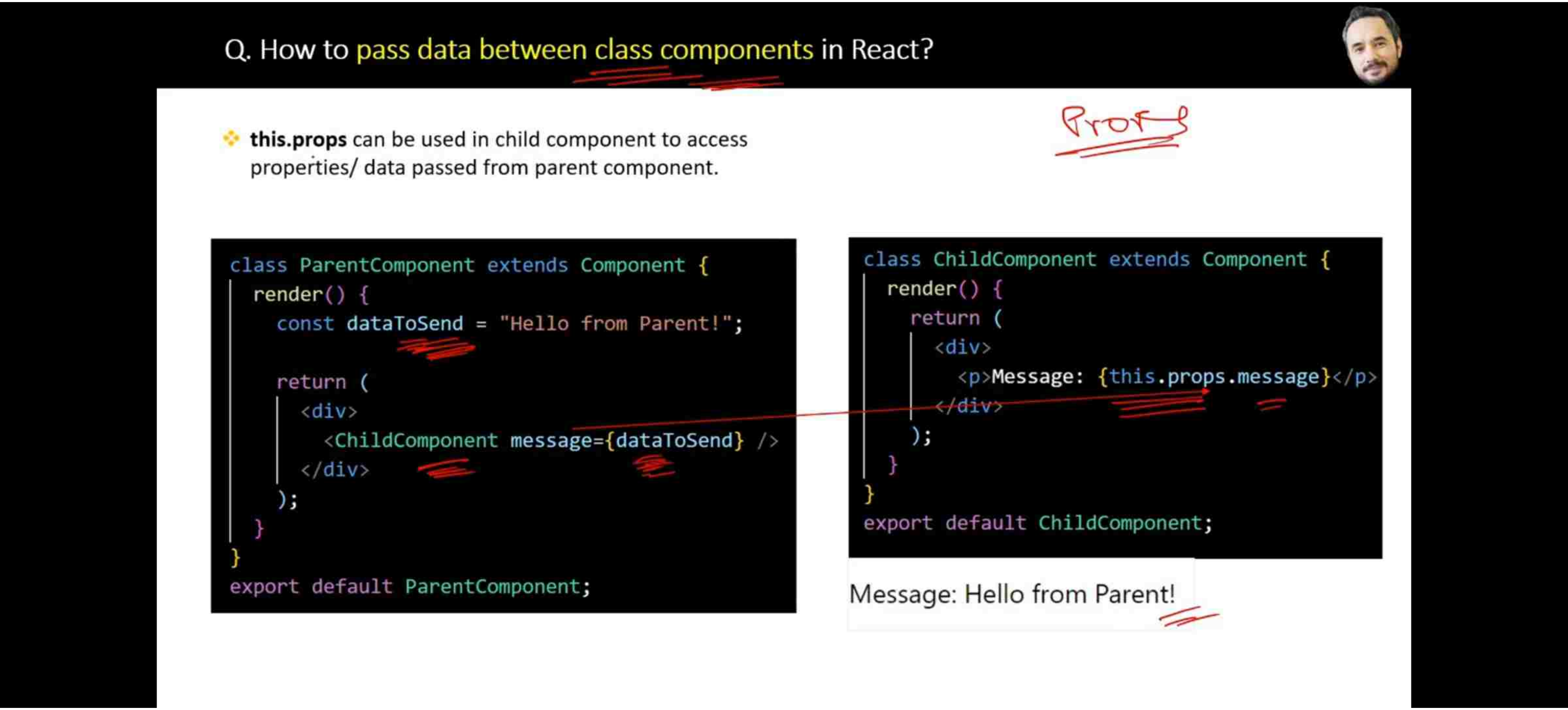
Types of Components

1. Functional Components

2. Class Components



```
import React, { Component } from 'react';

class AppClass extends Component {
  render() {
    return <h1>Interview Happy</h1>;
  }
}

export default AppClass;
```



Q. How to pass data between class components in React?

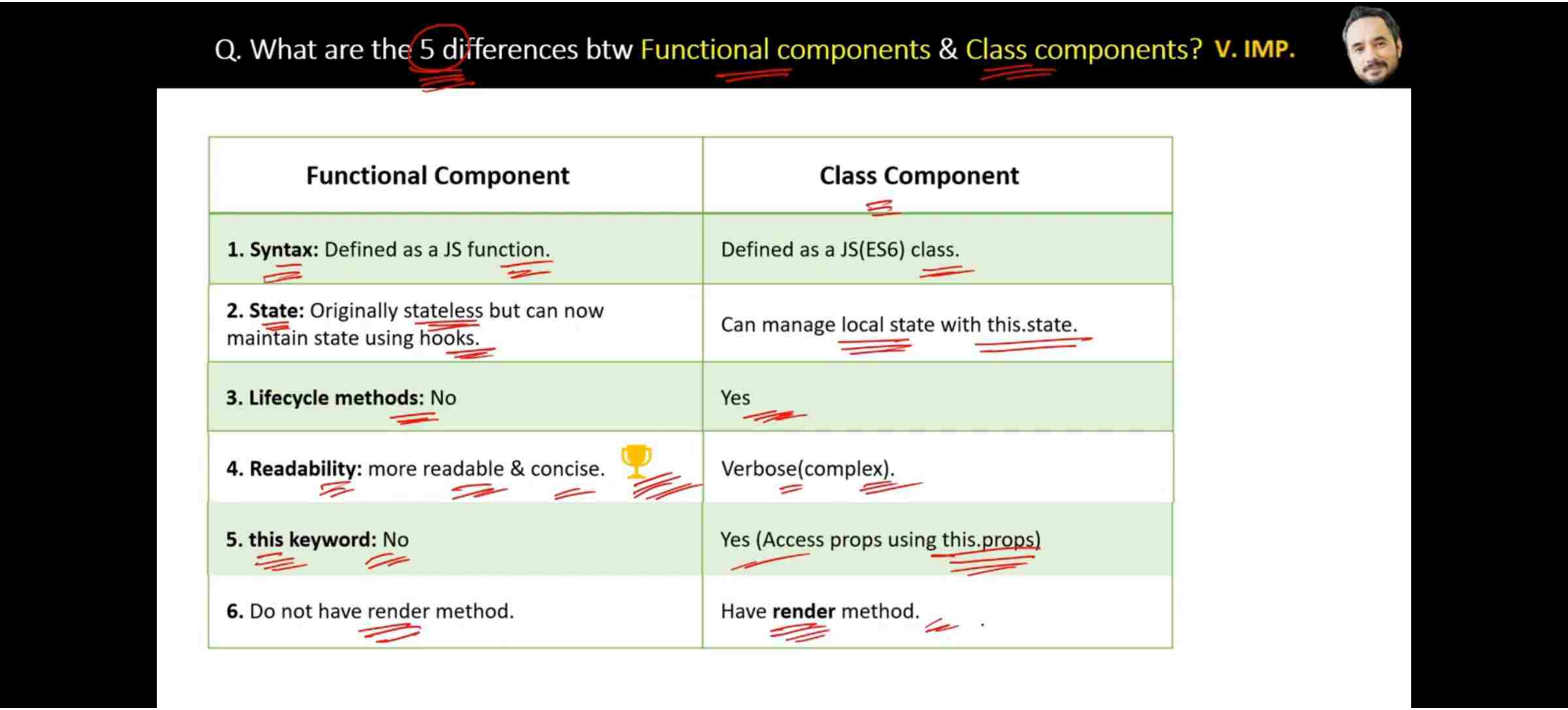
Props

- **this.props** can be used in child component to access properties/ data passed from parent component.

```
class ParentComponent extends Component {
  render() {
    const dataToSend = "Hello from Parent!";

    return (
      <div>
        <ChildComponent message={dataToSend} />
      </div>
    );
  }
}
export default ParentComponent;
```

```
class ChildComponent extends Component {
  render() {
    return (
      <div>
        <p>Message: {this.props.message}</p>
      </div>
    );
  }
}
export default ChildComponent;
```

```
Message: Hello from Parent!
```



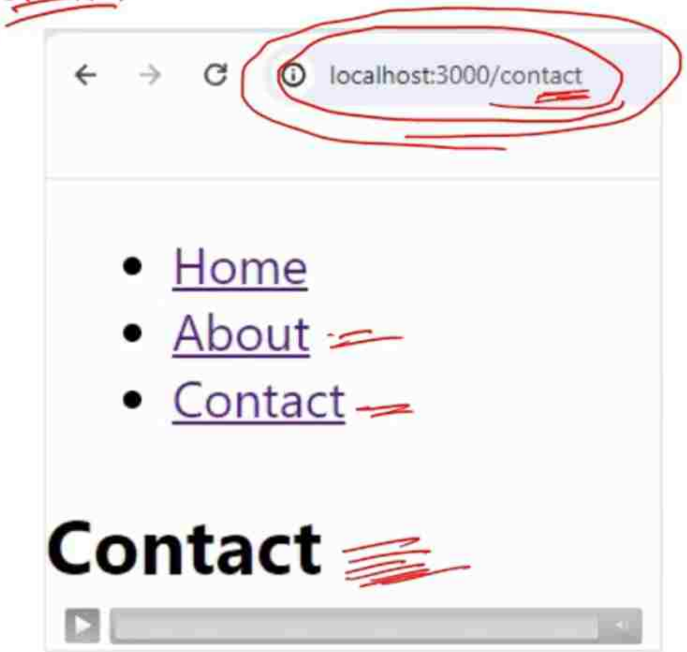
Q. What are the 5 differences btw Functional components & Class components? V. IMP.

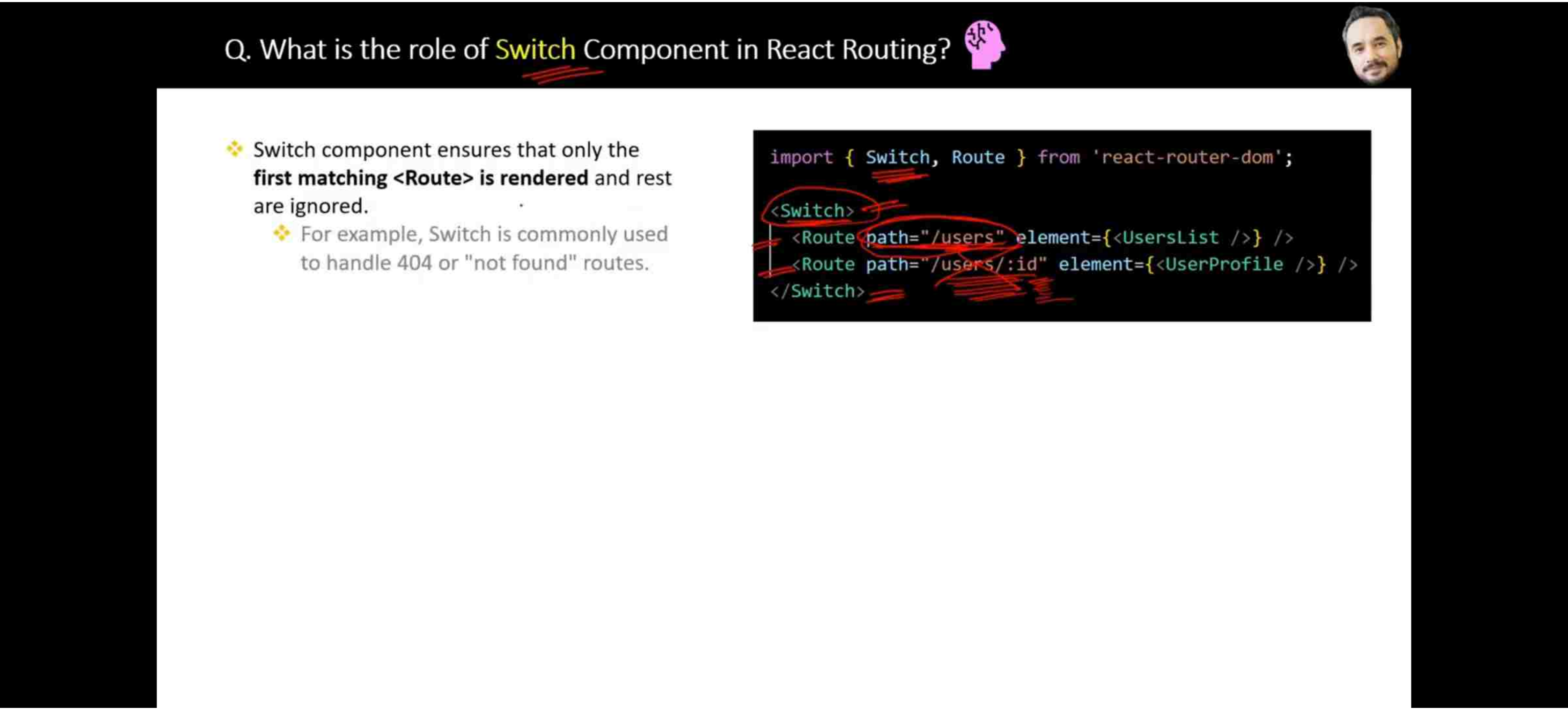
Functional Component	Class Component
1. Syntax: Defined as a JS function.	Defined as a JS(ES6) class.
2. State: Originally stateless but can now maintain state using hooks.	Can manage local state with this.state.
3. Lifecycle methods: No	Yes
4. Readability: more readable & concise. 🏆	Verbose(complex).
5. this keyword: No	Yes (Access props using this.props)
6. Do not have render method.	Have **render** method.

Q. What is Routing and Router in React? V. IMP.

Navigation

Routing	Routing allows you to create a single-page web application with **navigation**, without the need for a full-page refresh.
React Router	React Router is a library for handling routing and enables navigation and rendering of different components **based on the URL**.

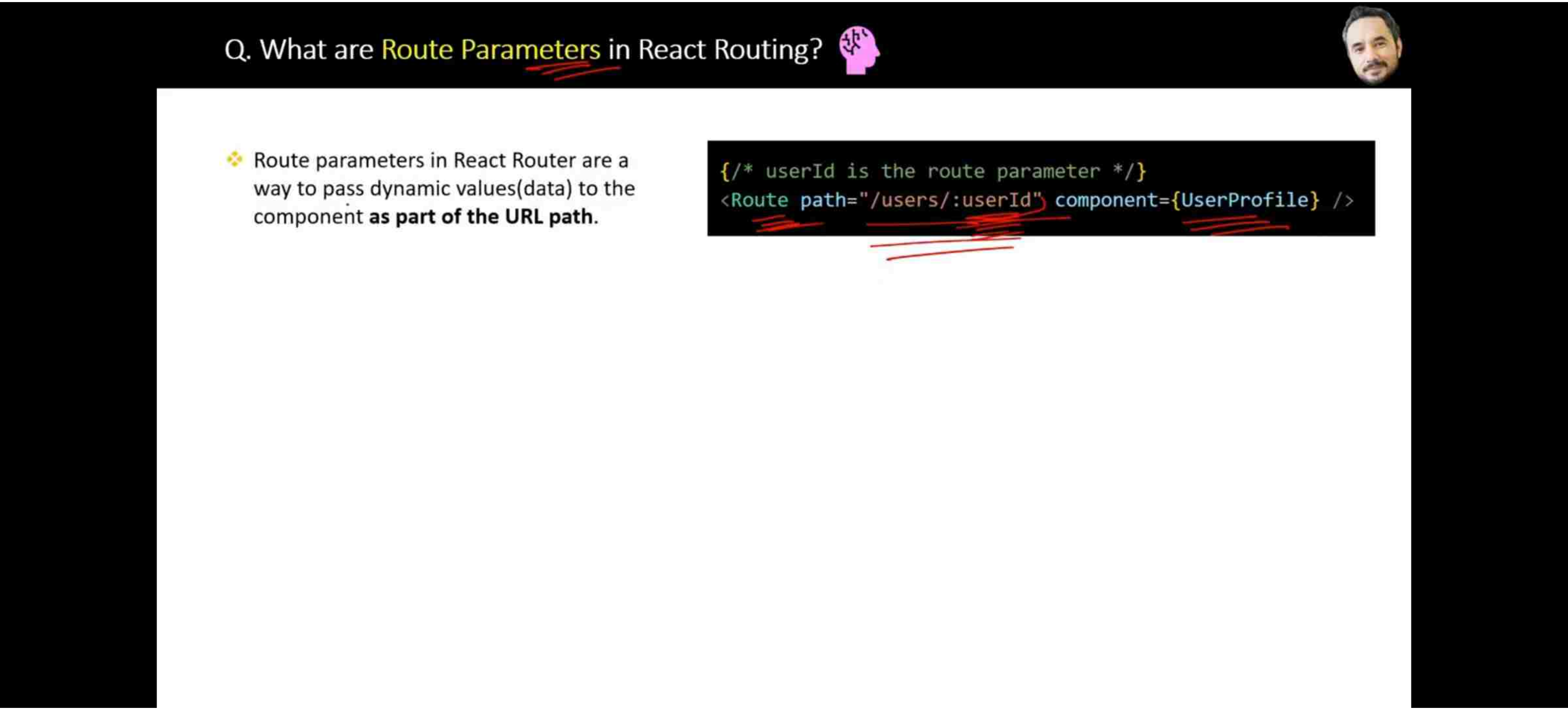




Q. What is the role of Switch Component in React Routing?

- Switch component ensures that only the **first matching <Route> is rendered** and rest are ignored.
  - For example, Switch is commonly used to handle 404 or "not found" routes.

```
import { Switch, Route } from 'react-router-dom';

<Switch>
  <Route path="/users" element={<UsersList />} />
  <Route path="/users/:id" element={<UserProfile />} />
</Switch>
```



Q. What are Route Parameters in React Routing?

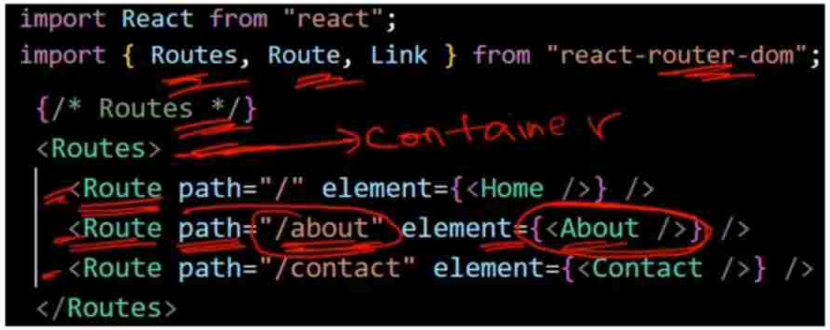
- Route parameters in React Router are a way to pass dynamic values(data) to the component **as part of the URL path.**

```
{/* userId is the route parameter */}
<Route path="/users/:userId" component={UserProfile} />
```

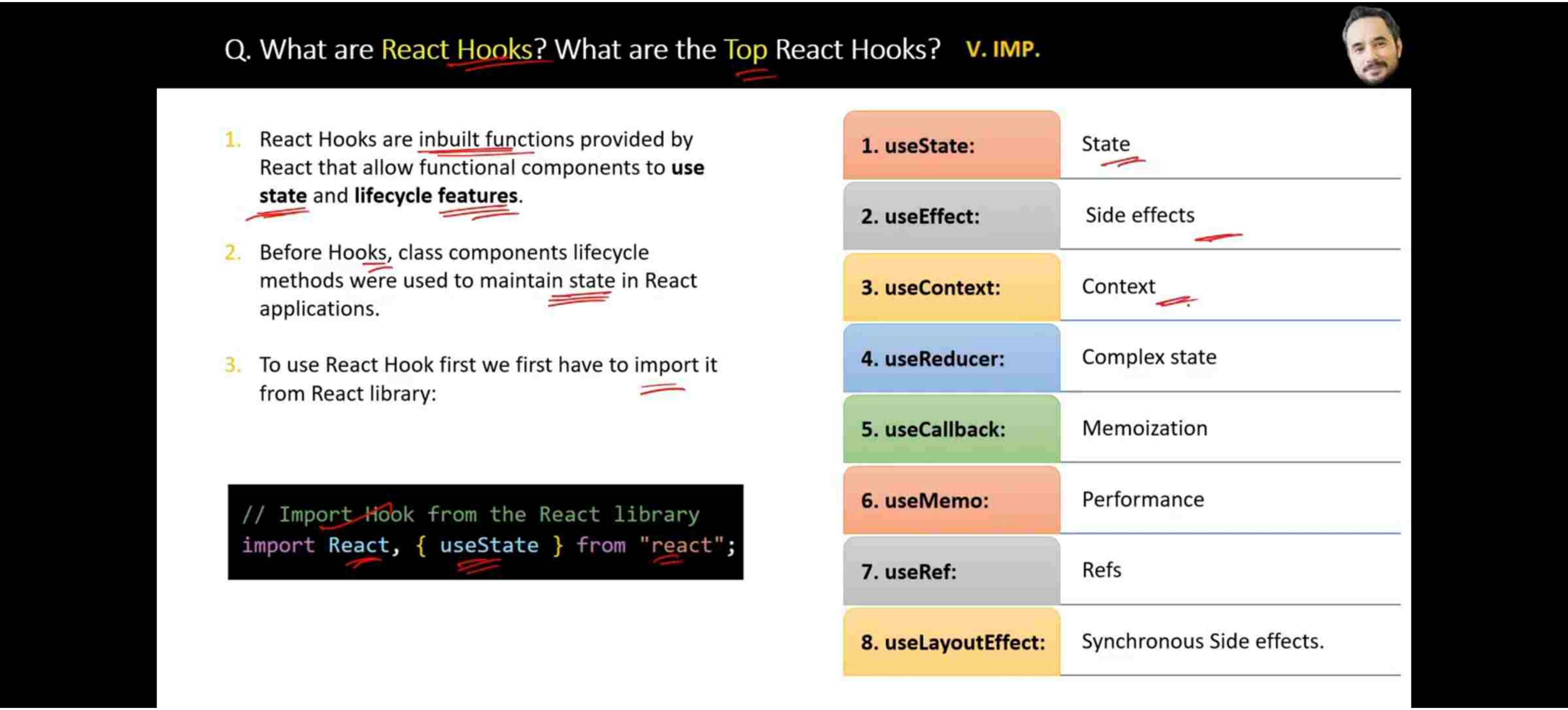
Q. What are the roles of <Routes> & <Route> component in React Routing? V. IMP.

- The <Routes> component is used as the root container for declaring your **collection of routes**.

- The <Route> component is used to define a route and specify the component that should render when the **route matches**.
  - For example, in this code if user enter "websitename.com/about" in url, then matching "About" component will be rendered.



```
import React from "react";
import { Routes, Route, Link } from "react-router-dom";

 {/* Routes */}
 <Routes>
  <Route path="/" element={<Home />} />
  <Route path="/about" element={<About />} />
  <Route path="/contact" element={<Contact />} />
 </Routes>
```



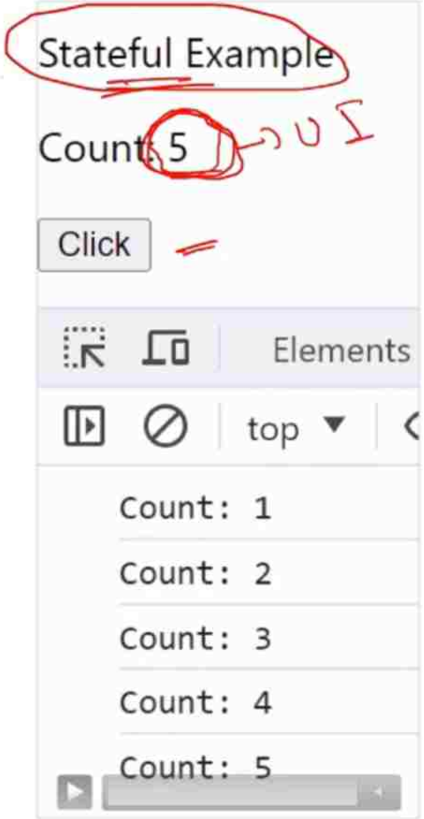
Q. What are React Hooks? What are the Top React Hooks? V. IMP.

1. React Hooks are inbuilt functions provided by React that allow functional components to **use state** and **lifecycle features**.
2. Before Hooks, class components lifecycle methods were used to maintain state in React applications.
3. To use React Hook first we first have to import it from React library:

```
// Import Hook from the React library
import React, { useState } from "react";
```

1. useState:	State
2. useEffect:	Side effects
3. useContext:	Context
4. useReducer:	Complex state
5. useCallback:	Memoization
6. useMemo:	Performance
7. useRef:	Refs
8. useLayoutEffect:	Synchronous Side effects.

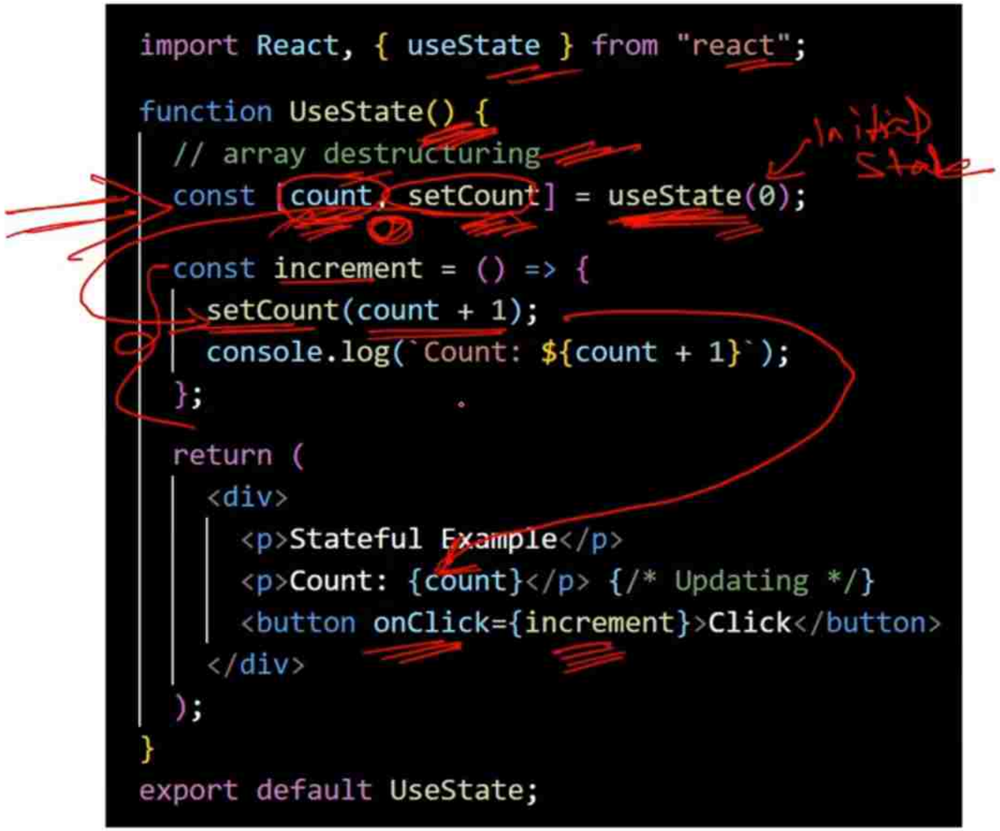
Q. What is the role of useState() hook and how it works? **V. IMP.**



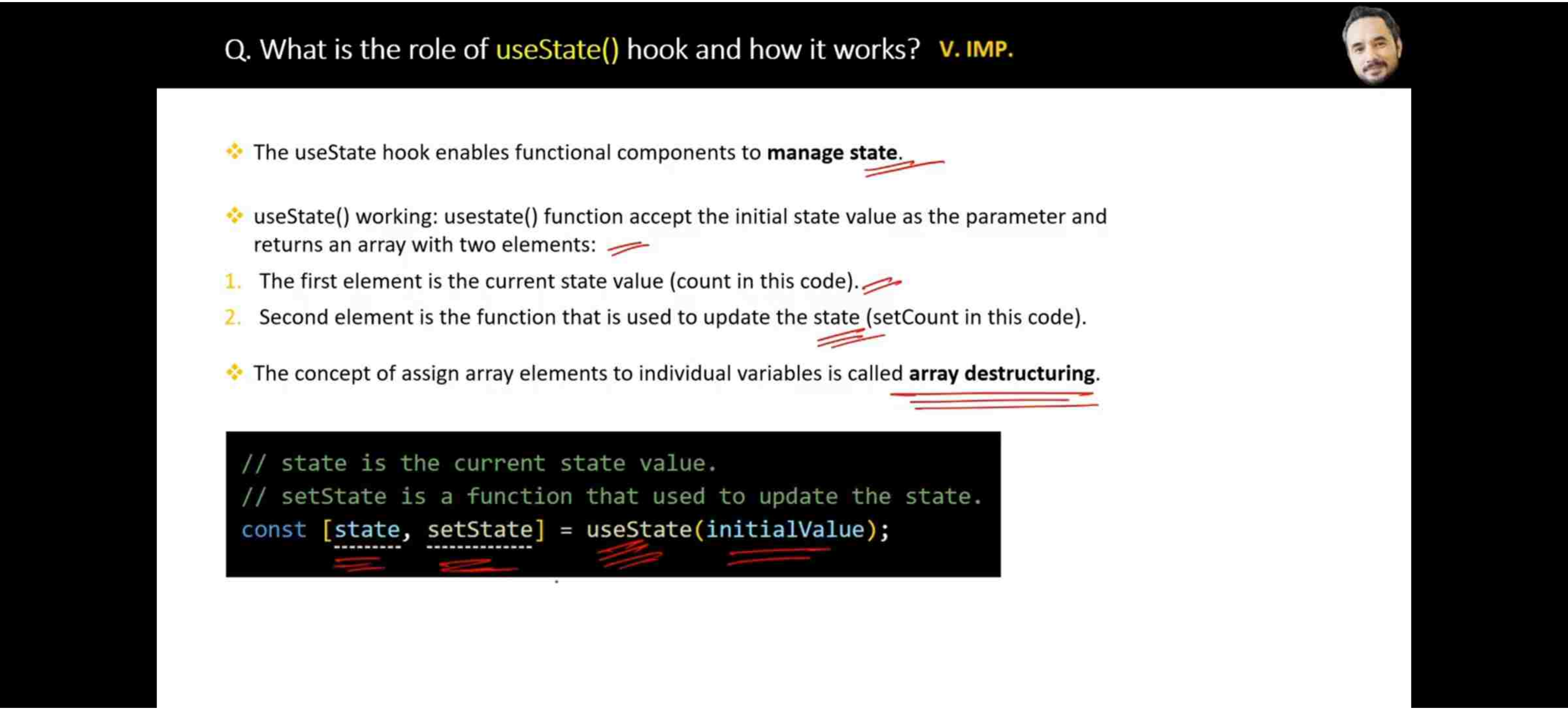
Stateful Example

Count: 5

```
Count: 1
Count: 2
Count: 3
Count: 4
Count: 5
```



```
import React, { useState } from "react";

function UseState() {
  // array destructuring
  const [count, setCount] = useState(0);

  const increment = () => {
    setCount(count + 1);
    console.log(`Count: ${count + 1}`);
  };

  return (
    <div>
      <p>Stateful Example</p>
      <p>Count: {count}</p> {/* Updating */}
      <button onClick={increment}>Click</button>
    </div>
  );
}
export default UseState;
```



Q. What is the role of useState() hook and how it works? V. IMP.

- The useState hook enables functional components to **manage state**.

- useState() working: usestate() function accept the initial state value as the parameter and returns an array with two elements:

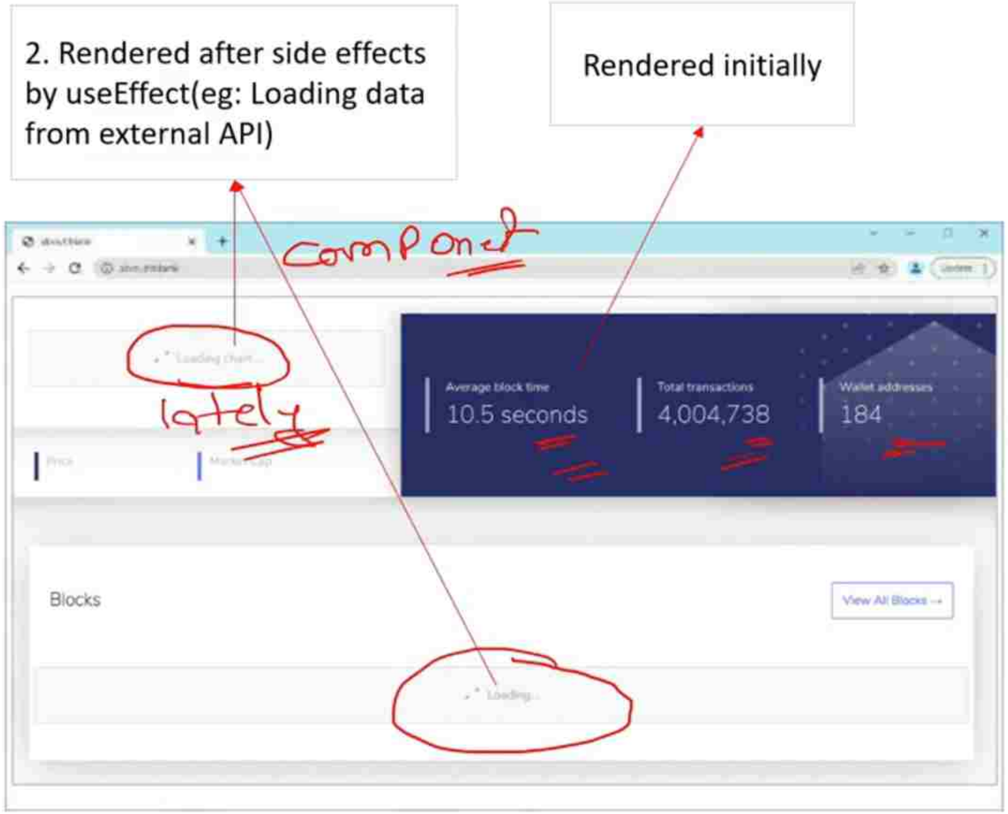
1. The first element is the current state value (count in this code).
2. Second element is the function that is used to update the state (setCount in this code).

- The concept of assign array elements to individual variables is called **array destructuring**.

```
// state is the current state value.
// setState is a function that used to update the state.
const [state, setState] = useState(initialValue);
```

Q. What is the role of useEffect(). How it works and what is its use? V. IMP.

- The useEffect Hook in React is used to perform **side effects** in functional components.
- For example, data fetching from API, subscriptions or any other operation that needs to be performed after the component has been rendered.



2. Rendered after side effects by useEffect(eg: Loading data from external API)

Rendered initially

component

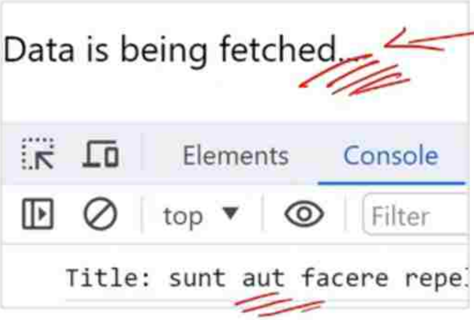
lately

Q. What is useEffect() hook and when to use it? V. IMP.

2 points to remember about useEffect():

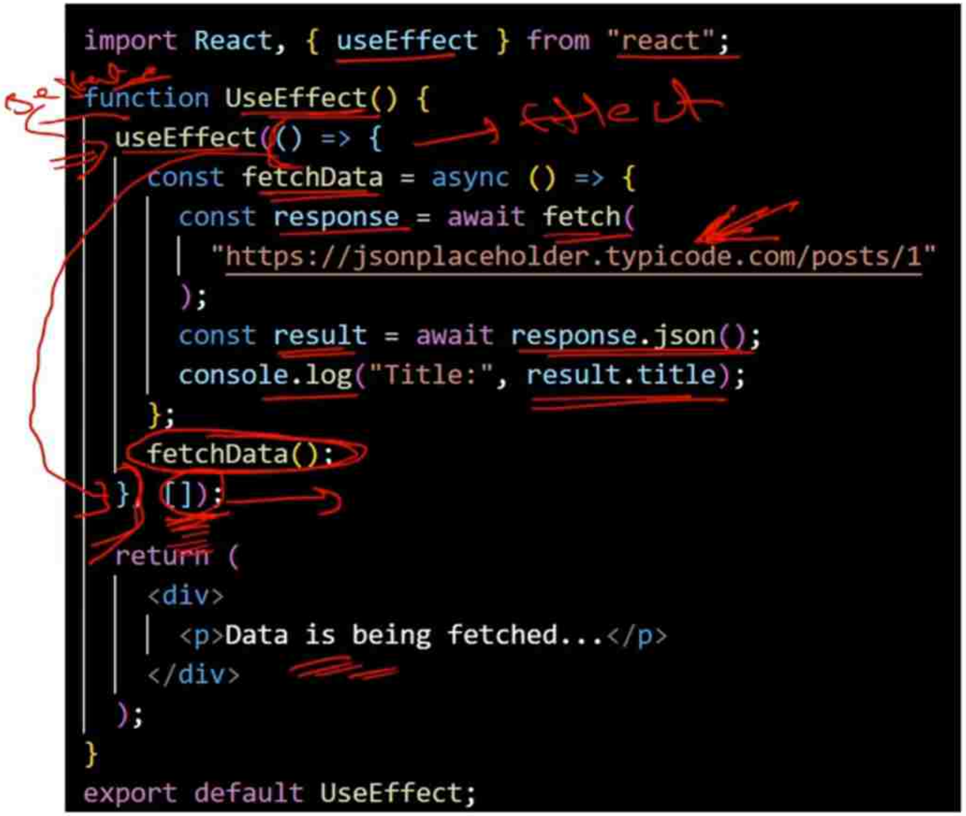
1. useEffect() is called after the component renders. Example, side effects.
2. useEffect() function will accept two parameter: (Effect function, dependency array).

Browser Output



Data is being fetched...

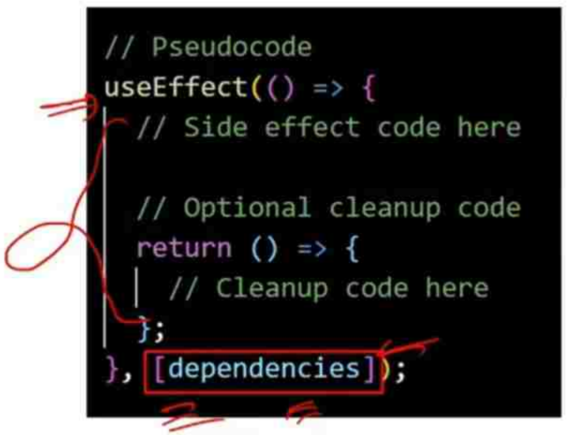
```
Title: sunt aut facere repe
```



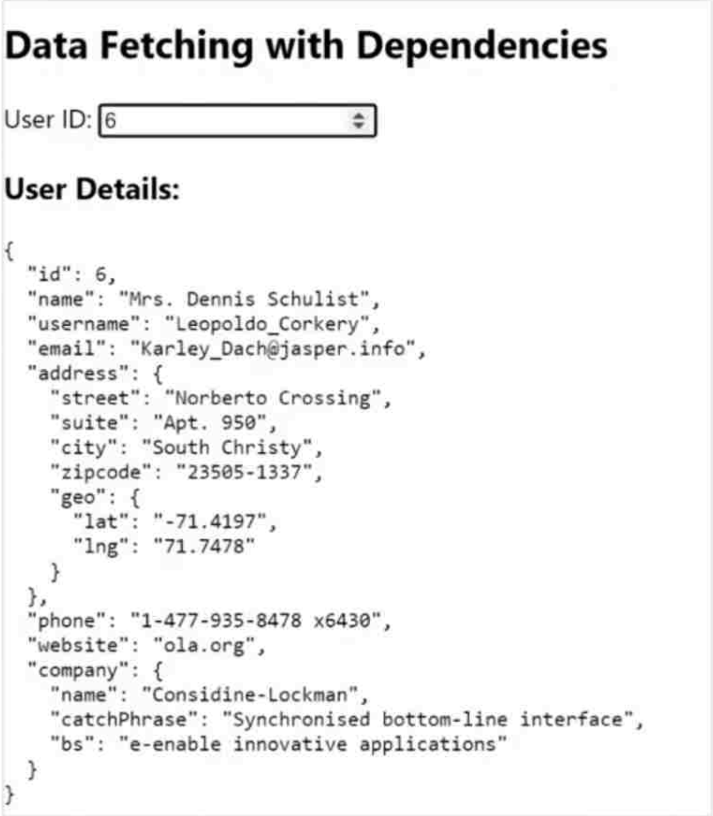
```
import React, { useEffect } from "react";

function UseEffect() {
  useEffect(() => {
    const fetchData = async () => {
      const response = await fetch(
        "https://jsonplaceholder.typicode.com/posts/1"
      );
      const result = await response.json();
      console.log("Title:", result.title);
    };
    fetchData();
  }, []);

  return (
    <div>
      <p>Data is being fetched...</p>
    </div>
  );
}
export default UseEffect;
```

Q. What is Dependency Array in useEffect() hook? V. IMP.

- Dependencies arrays(optional) act as triggers for useEffect to rerun; meaning if any of dependencies values change, the code inside useEffect() will be executed again.



```
// Pseudocode
useEffect(() => {
  // Side effect code here

  // Optional cleanup code
  return () => {
    // Cleanup code here
  };
}, [dependencies]);
```



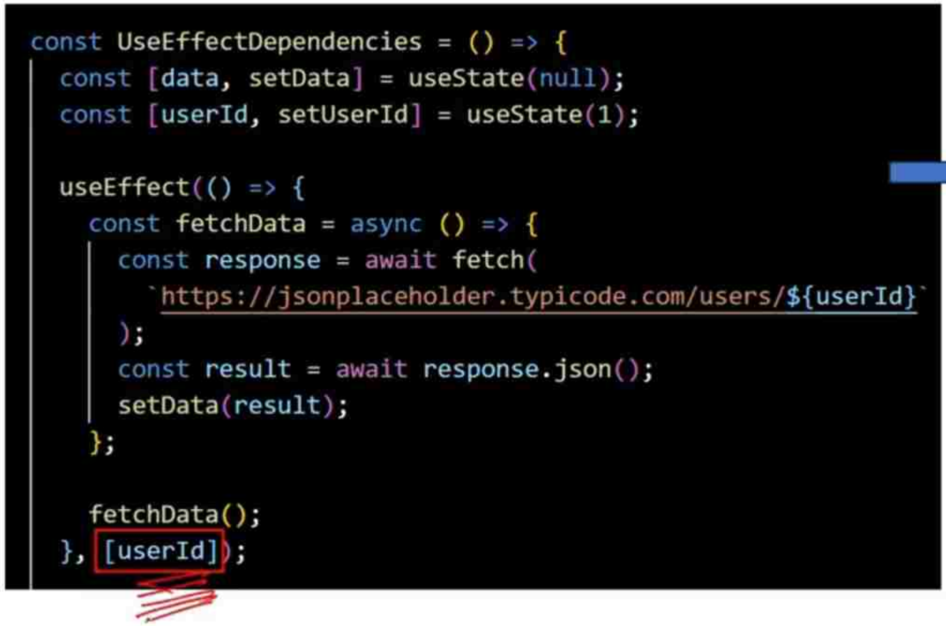
Data Fetching with Dependencies

User ID: 6

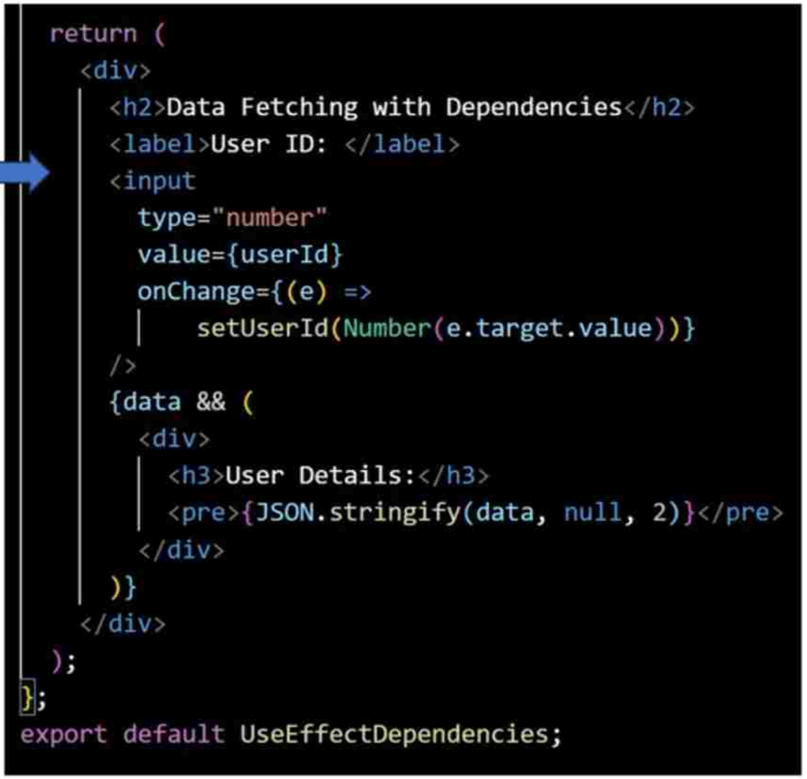
User Details:

```
{
  "id": 6,
  "name": "Mrs. Dennis Schulist",
  "username": "Leopoldo_Corkery",
  "email": "Karley_Dach@jasper.info",
  "address": {
    "street": "Norberto Crossing",
    "suite": "Apt. 950",
    "city": "South Christy",
    "zipcode": "23505-1337",
    "geo": {
      "lat": "-71.4197",
      "lng": "71.7478"
    }
  },
  "phone": "1-477-935-8478 x6430",
  "website": "ola.org",
  "company": {
    "name": "Considine-Lockman",
    "catchPhrase": "Synchronised bottom-line interface",
    "bs": "e-enable innovative applications"
  }
}
```

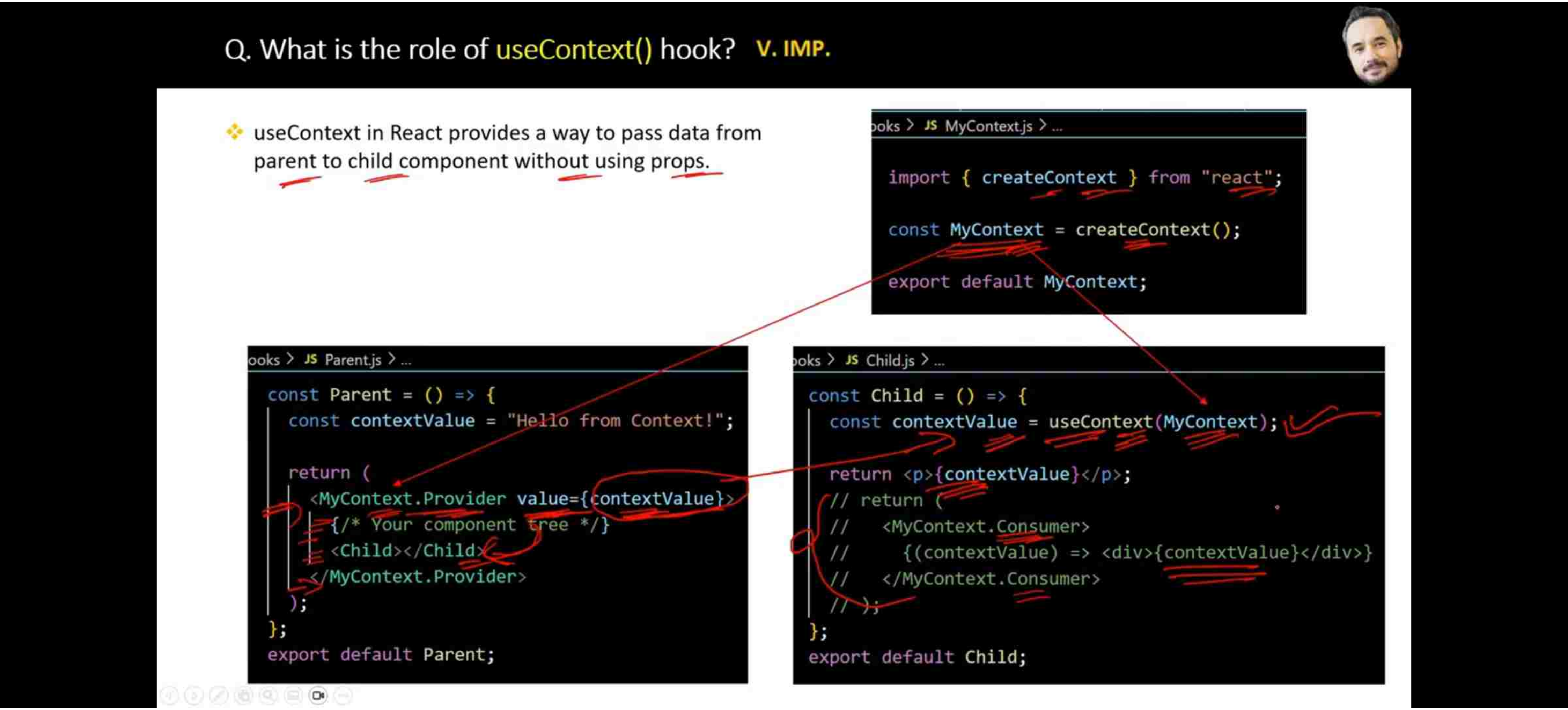
Q. What is Dependency Array in useEffect() hook?



```
const UseEffectDependencies = () => {
  const [data, setData] = useState(null);
  const [userId, setUserId] = useState(1);

  useEffect(() => {
    const fetchData = async () => {
      const response = await fetch(
        `https://jsonplaceholder.typicode.com/users/${userId}`
      );
      const result = await response.json();
      setData(result);
    };

    fetchData();
  }, [userId]);
```



```
  return (
    <div>
      <h2>Data Fetching with Dependencies</h2>
      <label>User ID: </label>
      <input
        type="number"
        value={userId}
        onChange={(e) =>
          setUserId(Number(e.target.value))}
      />
      {data && (
        <div>
          <h3>User Details:</h3>
          <pre>{JSON.stringify(data, null, 2)}</pre>
        </div>
      )}
    </div>
  );
};
export default UseEffectDependencies;
```



Q. What is the role of useContext() hook? V. IMP.

- useContext in React provides a way to pass data from parent to child component without using props.

ooks > JS MyContext.js > ...

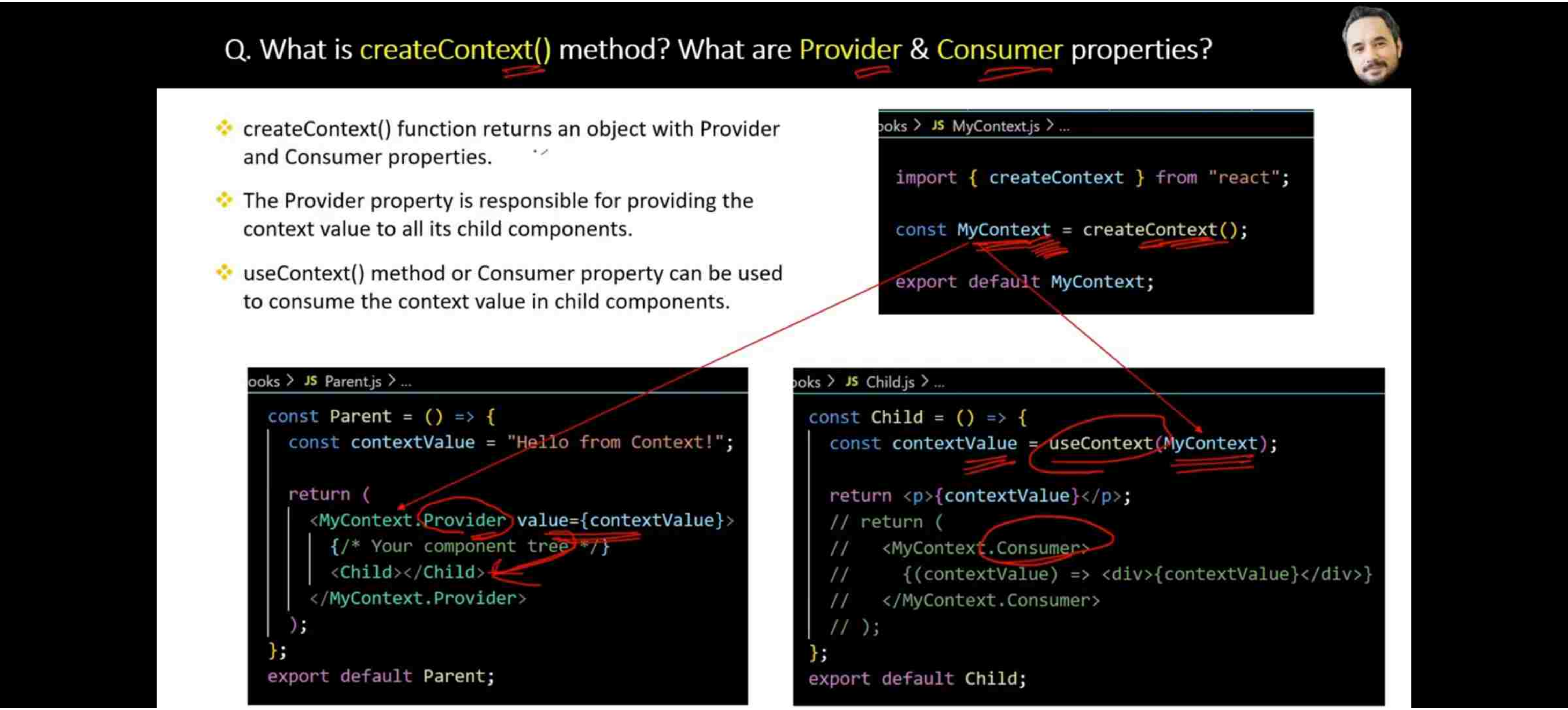
```
import { createContext } from "react";

const MyContext = createContext();

export default MyContext;
```

ooks > JS Parent.js > ...

```
const Parent = () => {
  const contextValue = "Hello from Context!";

  return (
    <MyContext.Provider value={contextValue}>
      {/* Your component tree */}
      <Child></Child>
    </MyContext.Provider>
  );
};
export default Parent;
```

ooks > JS Child.js > ...

```
const Child = () => {
  const contextValue = useContext(MyContext);

  return <p>{contextValue}</p>;
  // return (
  //   <MyContext.Consumer>
  //     {(contextValue) => <div>{contextValue}</div>}
  //   </MyContext.Consumer>
  // );
};
export default Child;
```



Q. What is createContext() method? What are Provider & Consumer properties?

- createContext() function returns an object with Provider and Consumer properties.
- The Provider property is responsible for providing the context value to all its child components.
- useContext() method or Consumer property can be used to consume the context value in child components.

oks > JS MyContext.js > ...

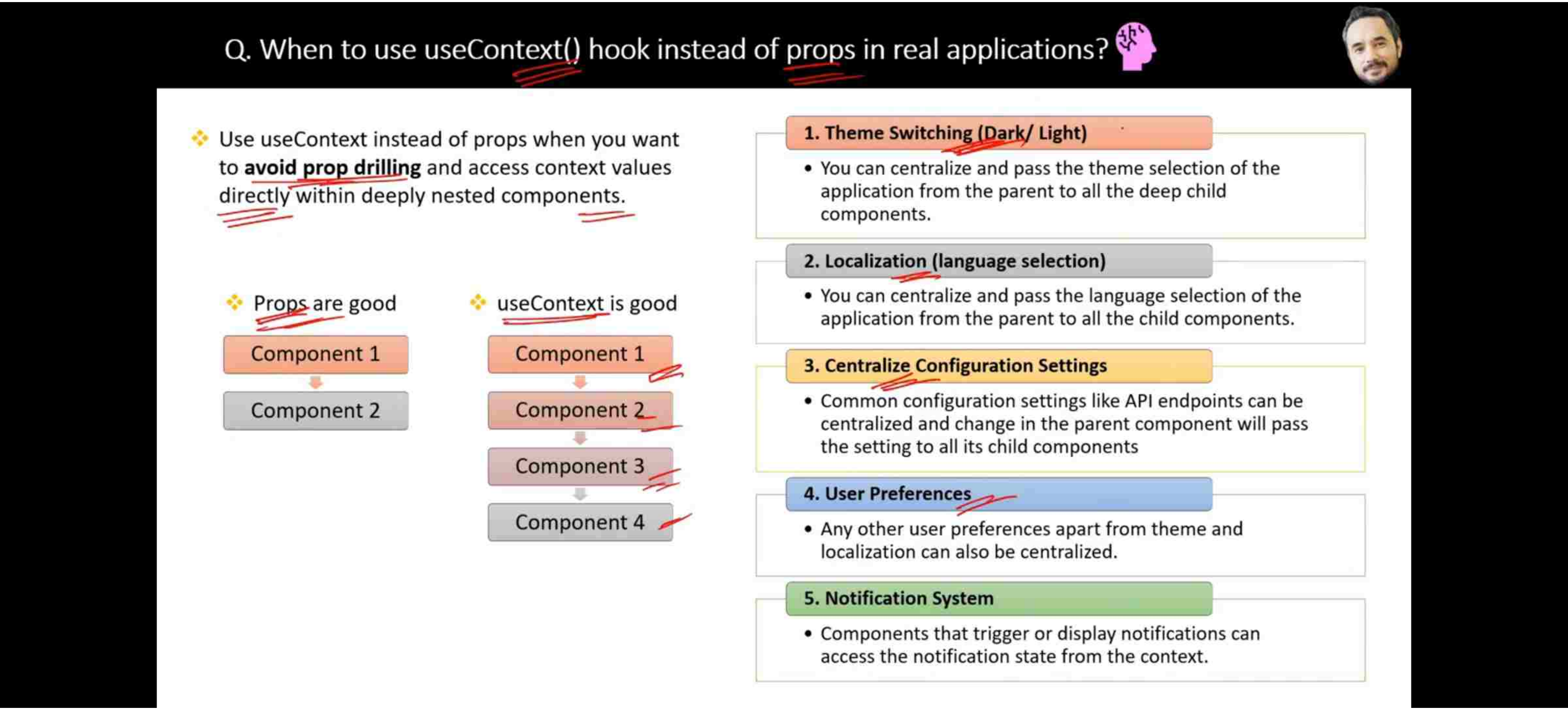
```
import { createContext } from "react";

const MyContext = createContext();

export default MyContext;
```

ooks > JS Parent.js > ...

```
const Parent = () => {
  const contextValue = "Hello from Context!";

  return (
    <MyContext.Provider value={contextValue}>
      {/* Your component tree */}
      <Child></Child>
    </MyContext.Provider>
  );
};
export default Parent;
```

oks > JS Child.js > ...

```
const Child = () => {
  const contextValue = useContext(MyContext);

  return <p>{contextValue}</p>;
  // return (
  //   <MyContext.Consumer>
  //     {(contextValue) => <div>{contextValue}</div>}
  //   </MyContext.Consumer>
  // );
};
export default Child;
```



Q. When to use useContext() hook instead of props in real applications?

- Use useContext instead of props when you want to **avoid prop drilling** and access context values directly within deeply nested components.

- Props are good
  - Component 1
  - Component 2

- useContext is good
  - Component 1
  - Component 2
  - Component 3
  - Component 4

1. Theme Switching (Dark/ Light)

- You can centralize and pass the theme selection of the application from the parent to all the deep child components.

2. Localization (language selection)

- You can centralize and pass the language selection of the application from the parent to all the child components.

3. Centralize Configuration Settings

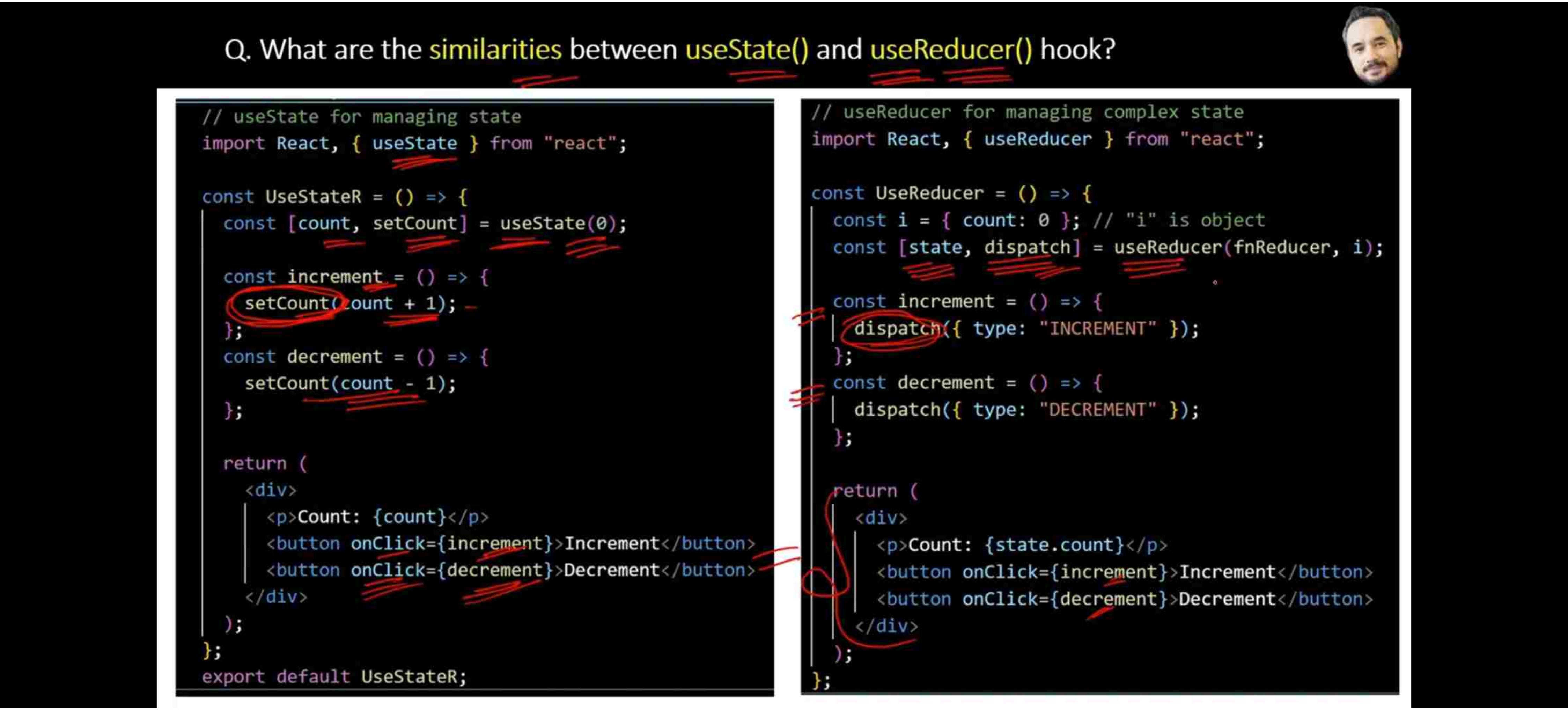
- Common configuration settings like API endpoints can be centralized and change in the parent component will pass the setting to all its child components

4. User Preferences

- Any other user preferences apart from theme and localization can also be centralized.

5. Notification System

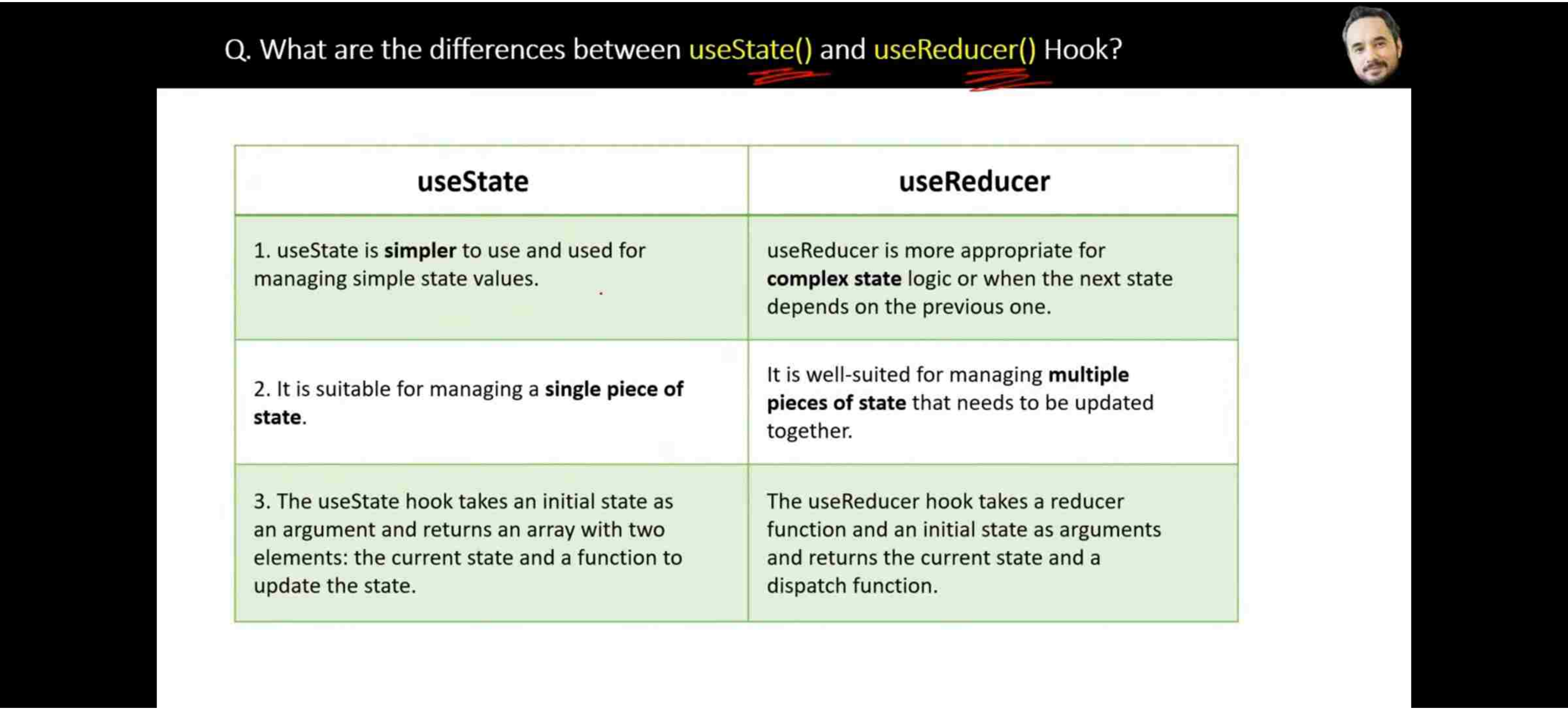
- Components that trigger or display notifications can access the notification state from the context.



Q. What are the similarities between useState() and useReducer() hook?

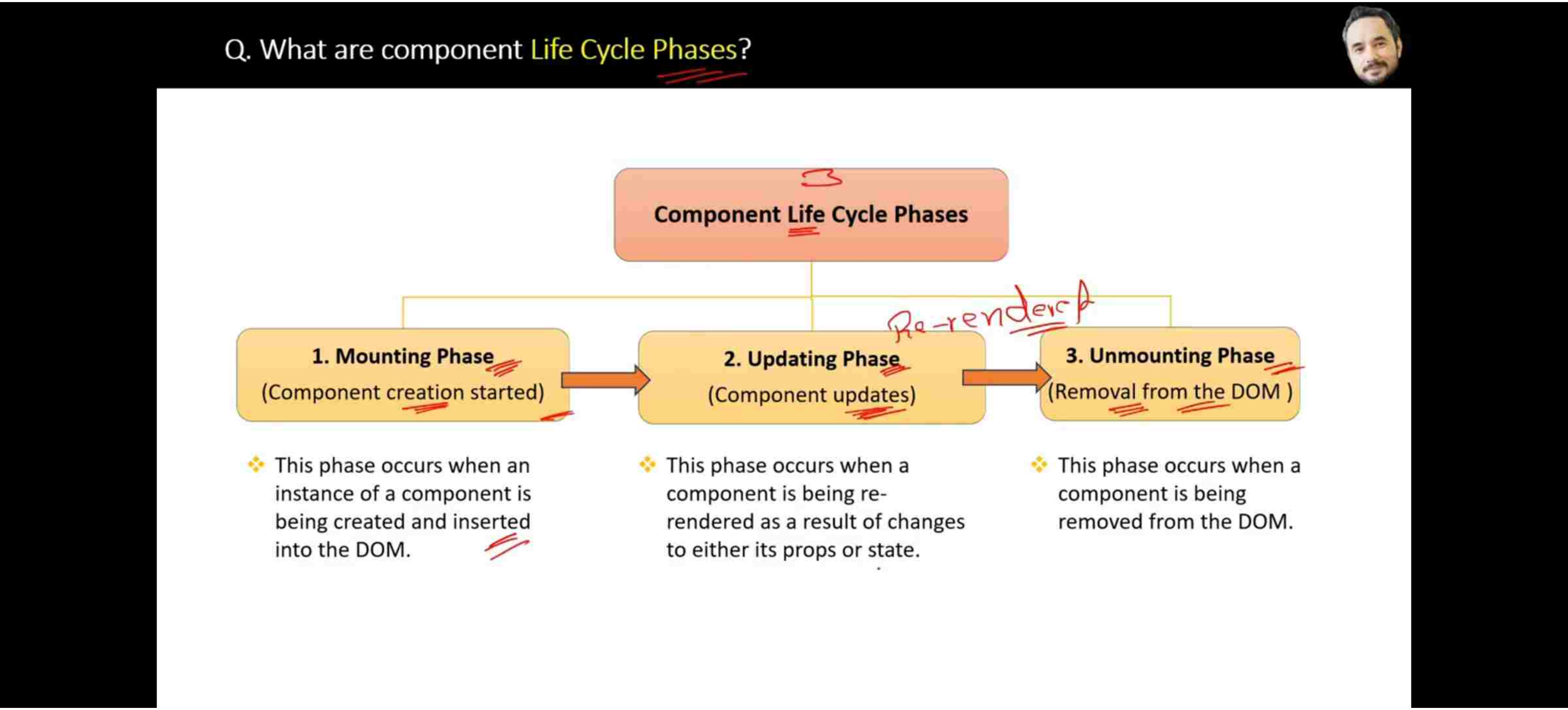
```
// useState for managing state
import React, { useState } from "react";

const UseStateR = () => {
  const [count, setCount] = useState(0);

  const increment = () => {
    setCount(count + 1);
  };
  const decrement = () => {
    setCount(count - 1);
  };

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={increment}>Increment</button>
      <button onClick={decrement}>Decrement</button>
    </div>
  );
};
export default UseStateR;
```

```
// useReducer for managing complex state
import React, { useReducer } from "react";

const UseReducer = () => {
  const i = { count: 0 }; // "i" is object
  const [state, dispatch] = useReducer(fnReducer, i);

  const increment = () => {
    dispatch({ type: "INCREMENT" });
  };
  const decrement = () => {
    dispatch({ type: "DECREMENT" });
  };

  return (
    <div>
      <p>Count: {state.count}</p>
      <button onClick={increment}>Increment</button>
      <button onClick={decrement}>Decrement</button>
    </div>
  );
};
```



Q. What are the differences between useState() and useReducer() Hook?

useState	useReducer
1. useState is **simpler** to use and used for managing simple state values.	useReducer is more appropriate for **complex state** logic or when the next state depends on the previous one.
2. It is suitable for managing a **single piece of state**.	It is well-suited for managing **multiple pieces of state** that needs to be updated together.
3. The useState hook takes an initial state as an argument and returns an array with two elements: the current state and a function to update the state.	The useReducer hook takes a reducer function and an initial state as arguments and returns the current state and a dispatch function.



Q. What are component Life Cycle Phases?

3

Component Life Cycle Phases

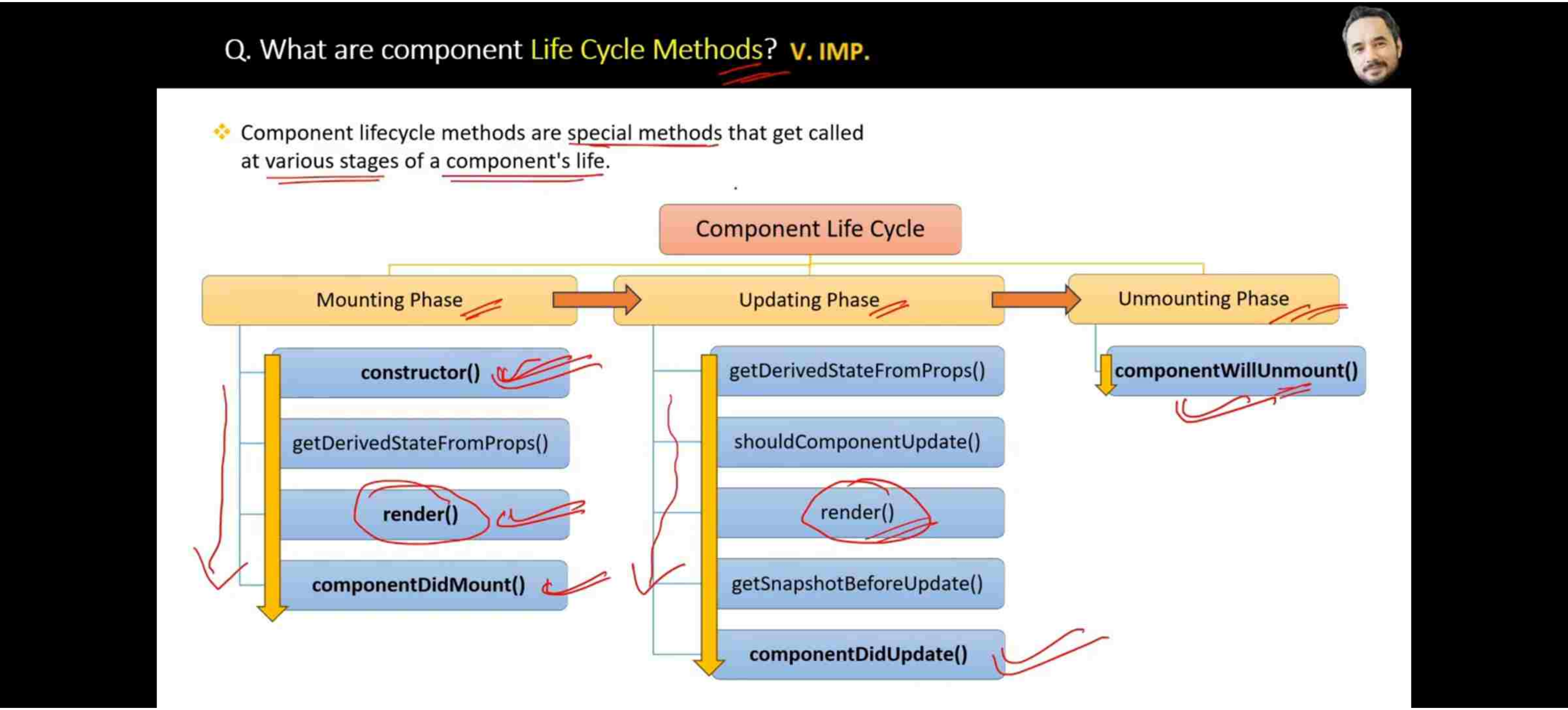
Re-rendered

1. Mounting Phase
(Component creation started)

2. Updating Phase
(Component updates)

3. Unmounting Phase
(Removal from the DOM)

- This phase occurs when an instance of a component is being created and inserted into the DOM.
- This phase occurs when a component is being re-rendered as a result of changes to either its props or state.
- This phase occurs when a component is being removed from the DOM.



Q. What are component Life Cycle Methods? V. IMP.

- Component lifecycle methods are special methods that get called at various stages of a component's life.

Component Life Cycle

Mounting Phase

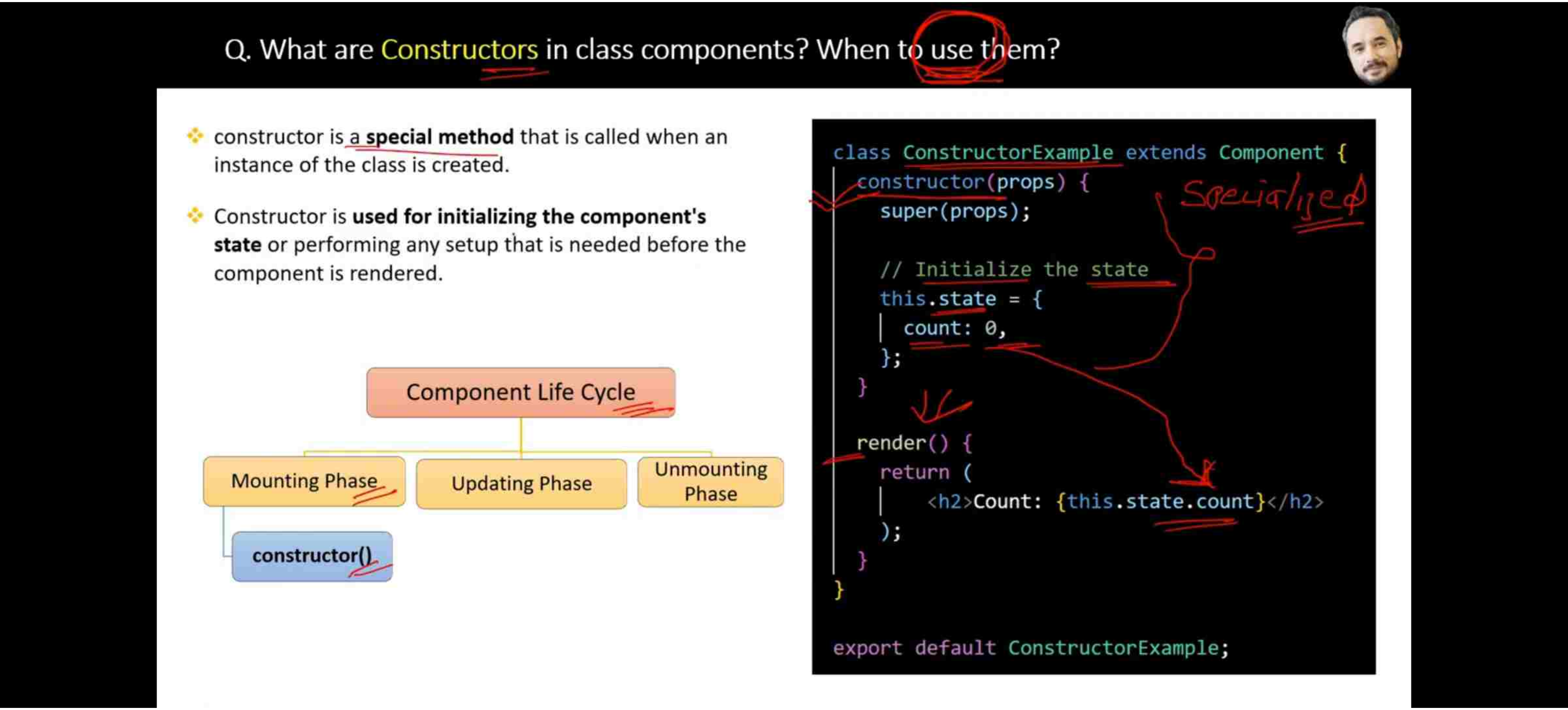
- **constructor()**
- getDerivedStateFromProps()
- **render()**
- **componentDidMount()**

Updating Phase

- getDerivedStateFromProps()
- shouldComponentUpdate()
- render()
- getSnapshotBeforeUpdate()
- **componentDidUpdate()**

Unmounting Phase

- **componentWillUnmount()**



Q. What are Constructors in class components? When to use them?

- constructor is a **special method** that is called when an instance of the class is created.
- Constructor is **used for initializing the component's state** or performing any setup that is needed before the component is rendered.

Component Life Cycle

Mounting Phase

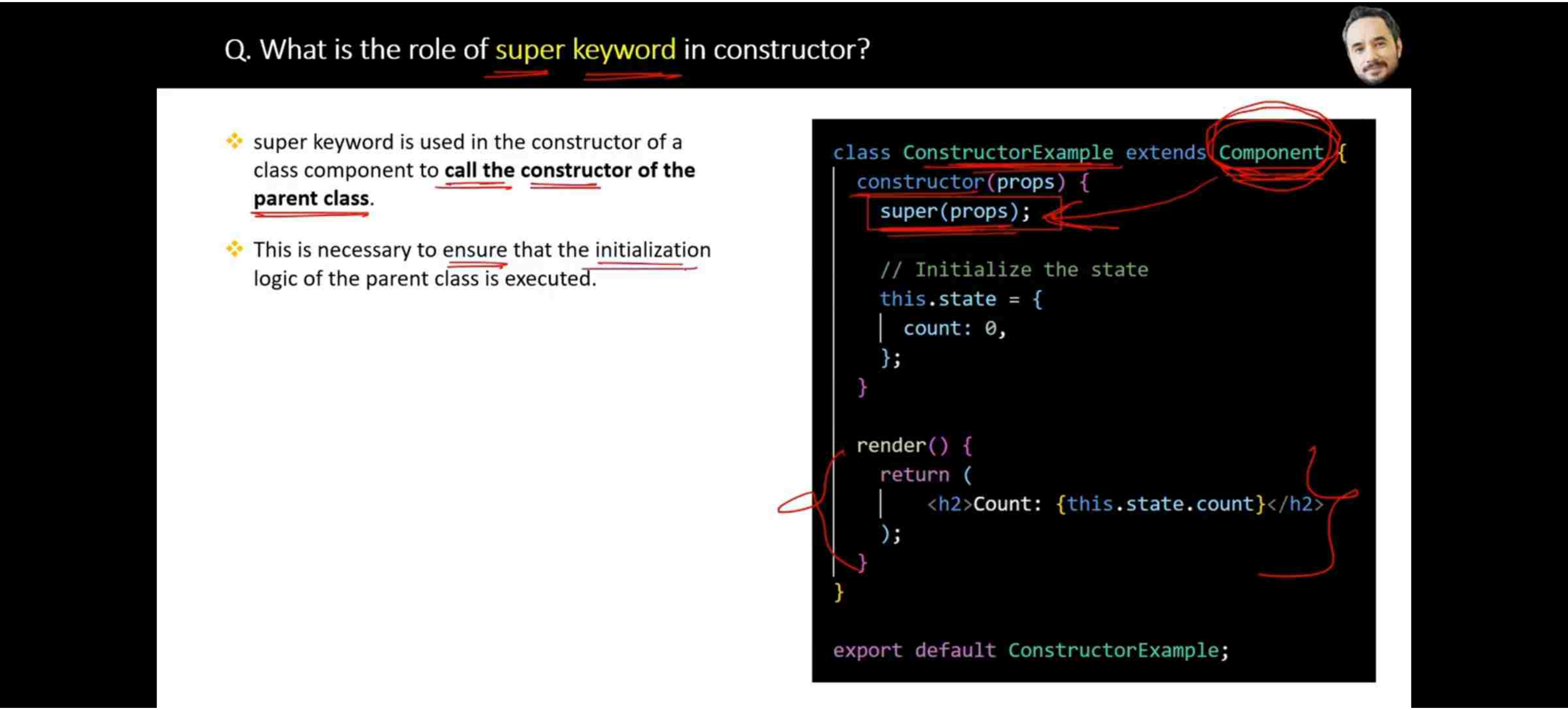
Updating Phase

Unmounting Phase

constructor()

```
class ConstructorExample extends Component {
  constructor(props) {
    super(props);

    // Initialize the state
    this.state = {
      count: 0,
    };
  }

  render() {
    return (
        <h2>Count: {this.state.count}</h2>
    );
  }
}

export default ConstructorExample;
```

Specialized



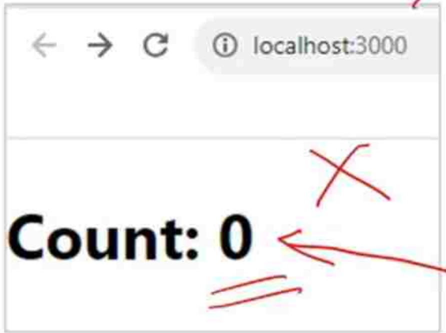
Q. What is the role of super keyword in constructor?

- super keyword is used in the constructor of a class component to **call the constructor of the parent class**.
- This is necessary to ensure that the initialization logic of the parent class is executed.

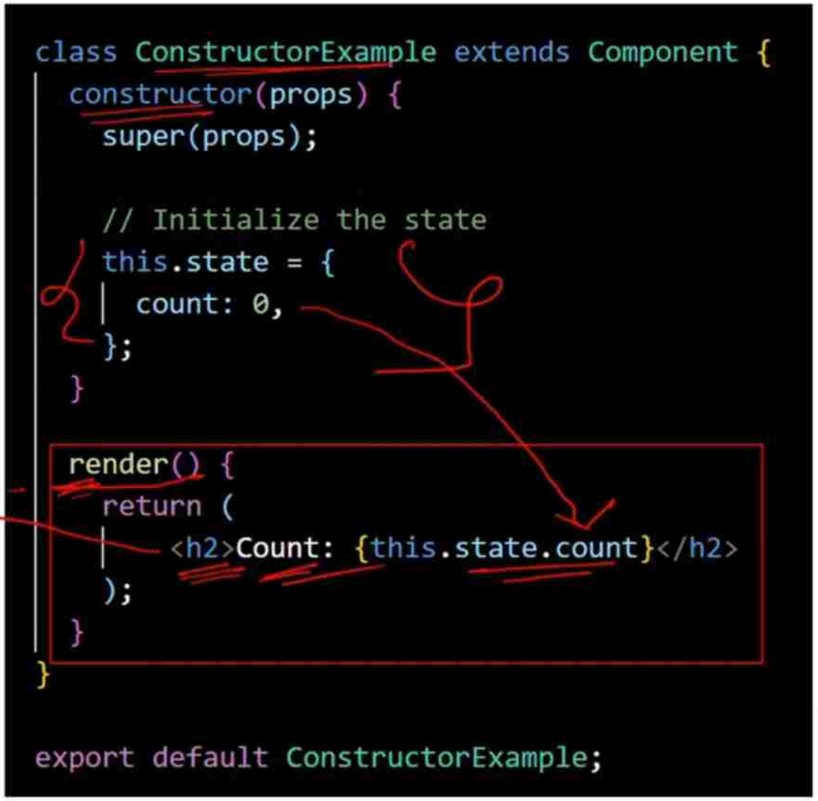
```
class ConstructorExample extends Component {
  constructor(props) {
    super(props);

    // Initialize the state
    this.state = {
      count: 0,
    };
  }

  render() {
    return (
        <h2>Count: {this.state.count}</h2>
    );
  }
}

export default ConstructorExample;
```

Q. What is the role of render() method in component life cycle?

- Render() method **returns the React elements** that will be rendered to the DOM.



Count: 0



```
class ConstructorExample extends Component {
  constructor(props) {
    super(props);

    // Initialize the state
    this.state = {
      count: 0,
    };
  }

  render() {
    return (
      <h2>Count: {this.state.count}</h2>
    );
  }
}

export default ConstructorExample;
```