JAVASCRIPT:

1.) <button onclick = { ( ) => { console.log("Welcome to Programiz!"); } } >

</button>

2.) let name ="Priya";

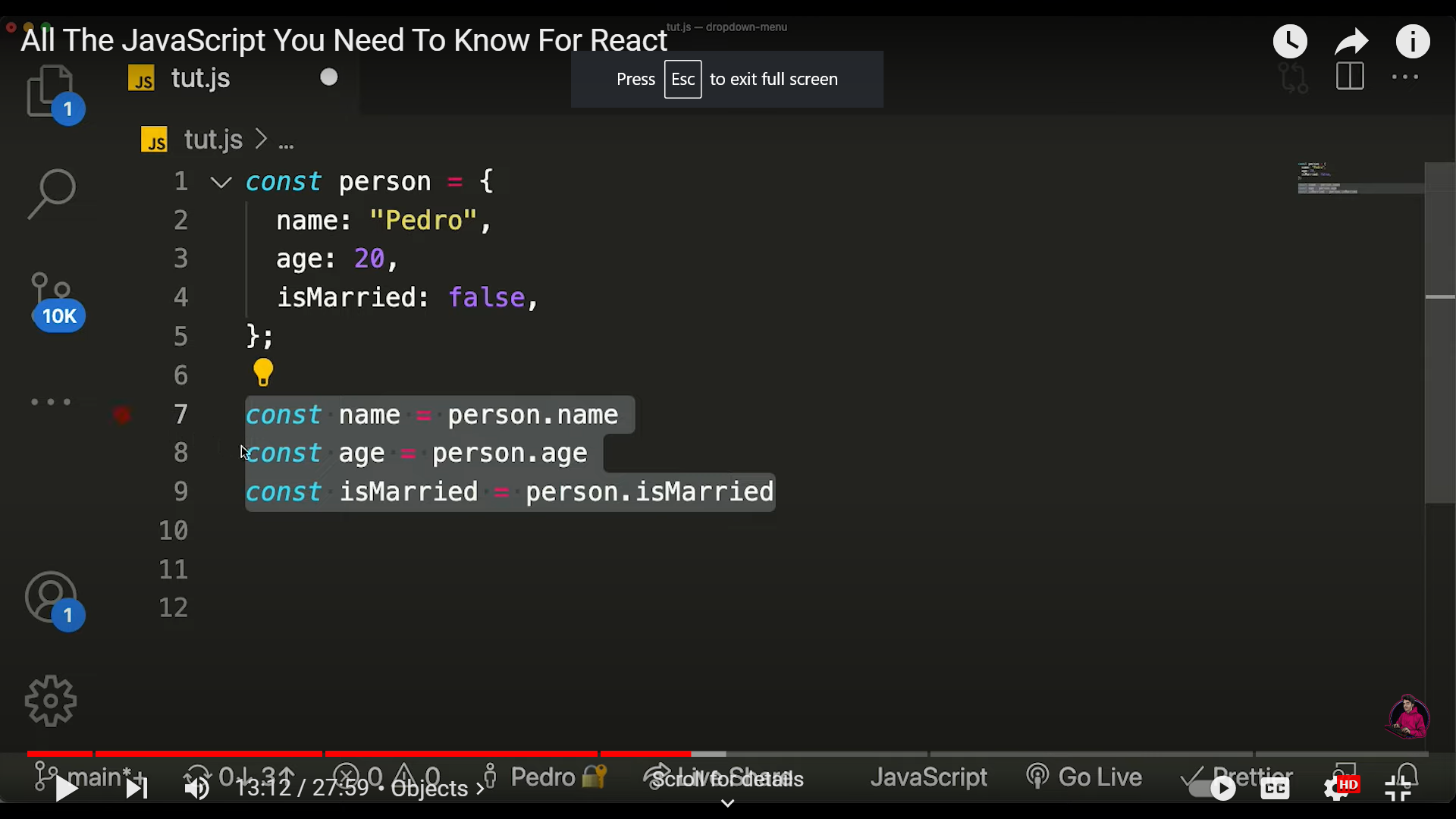
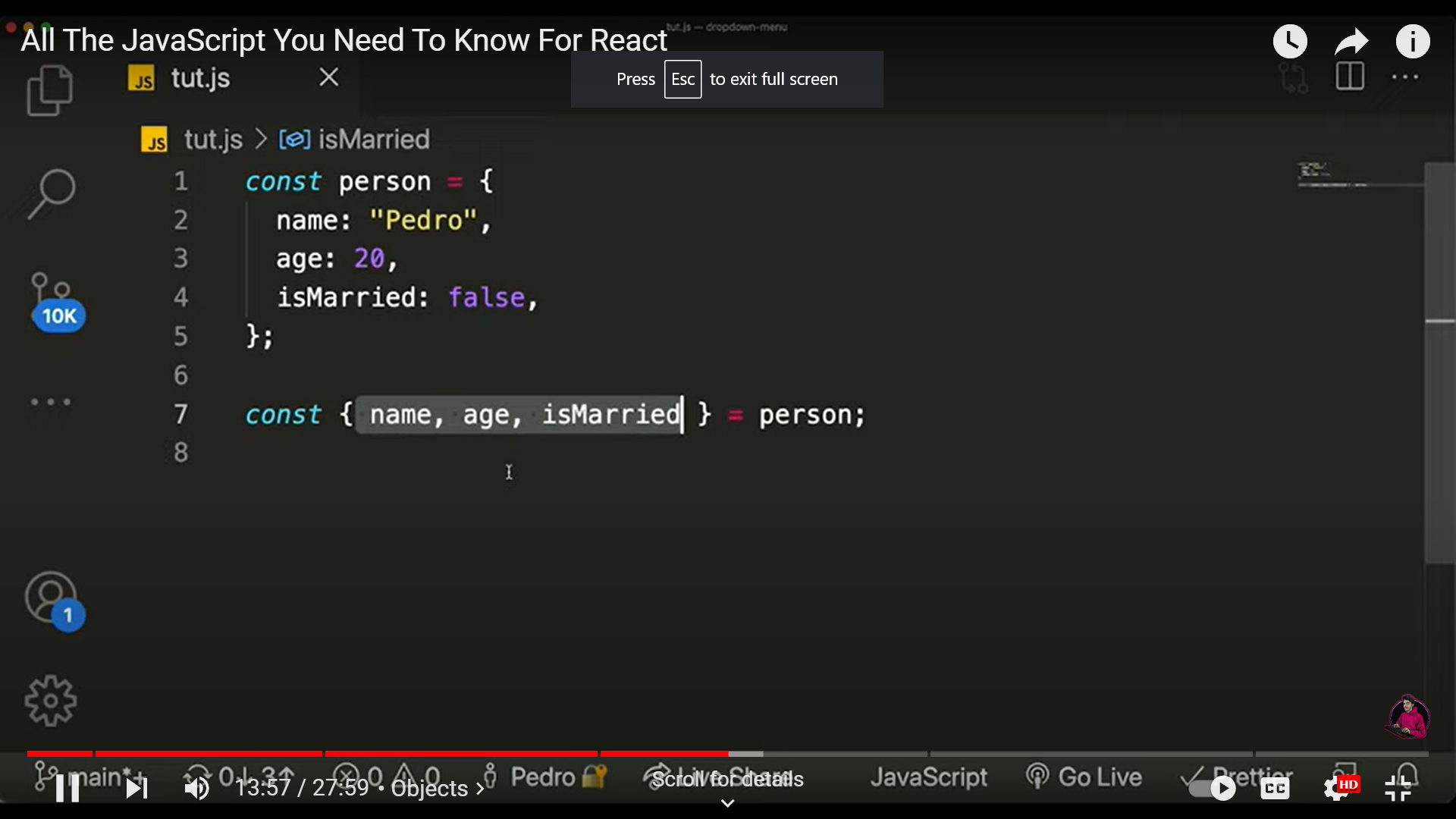
let num=16;

const Component = () => {

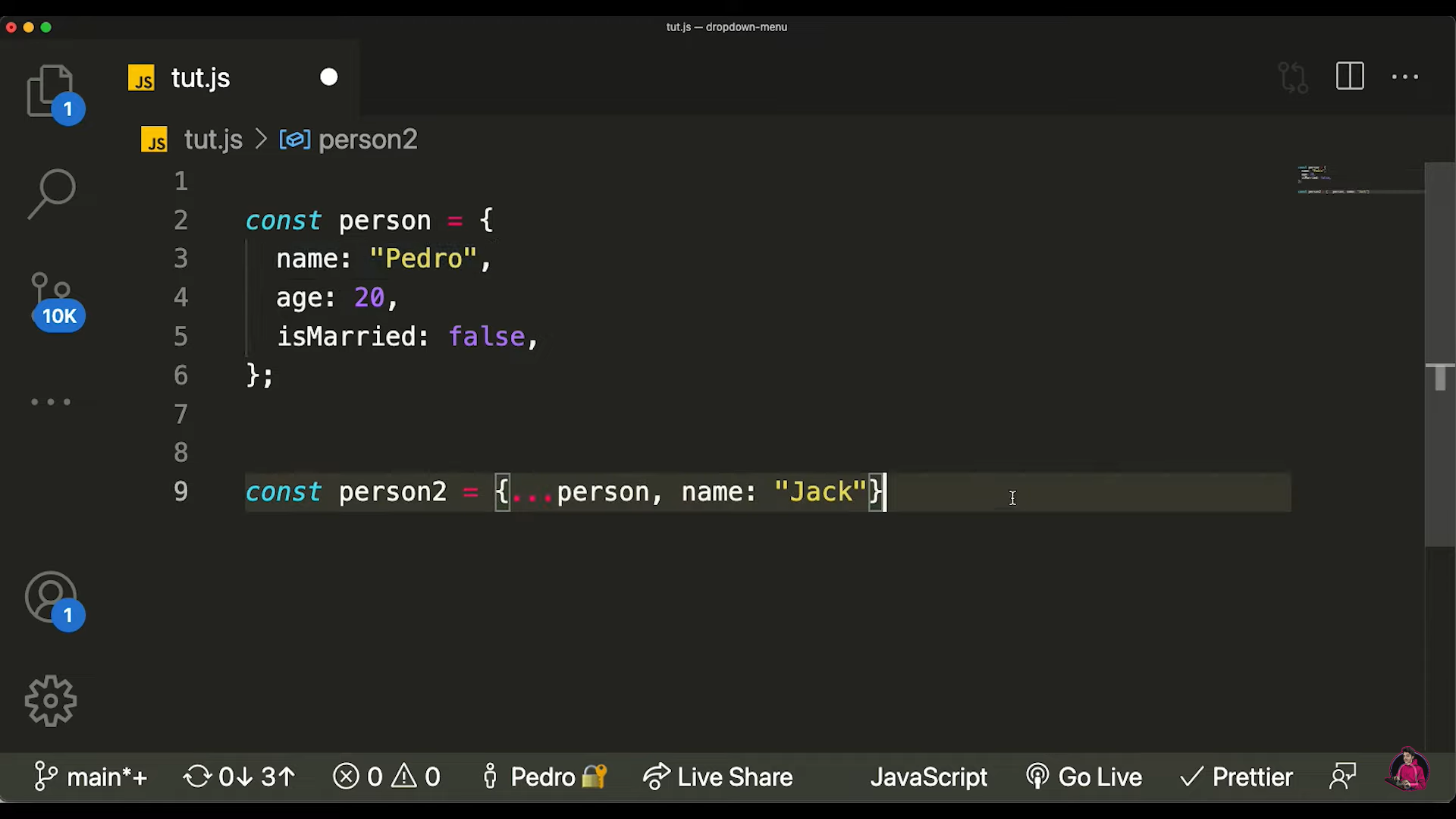
return num > 10 ? <div>"Priya"</div> : <div>"hari"</div>;

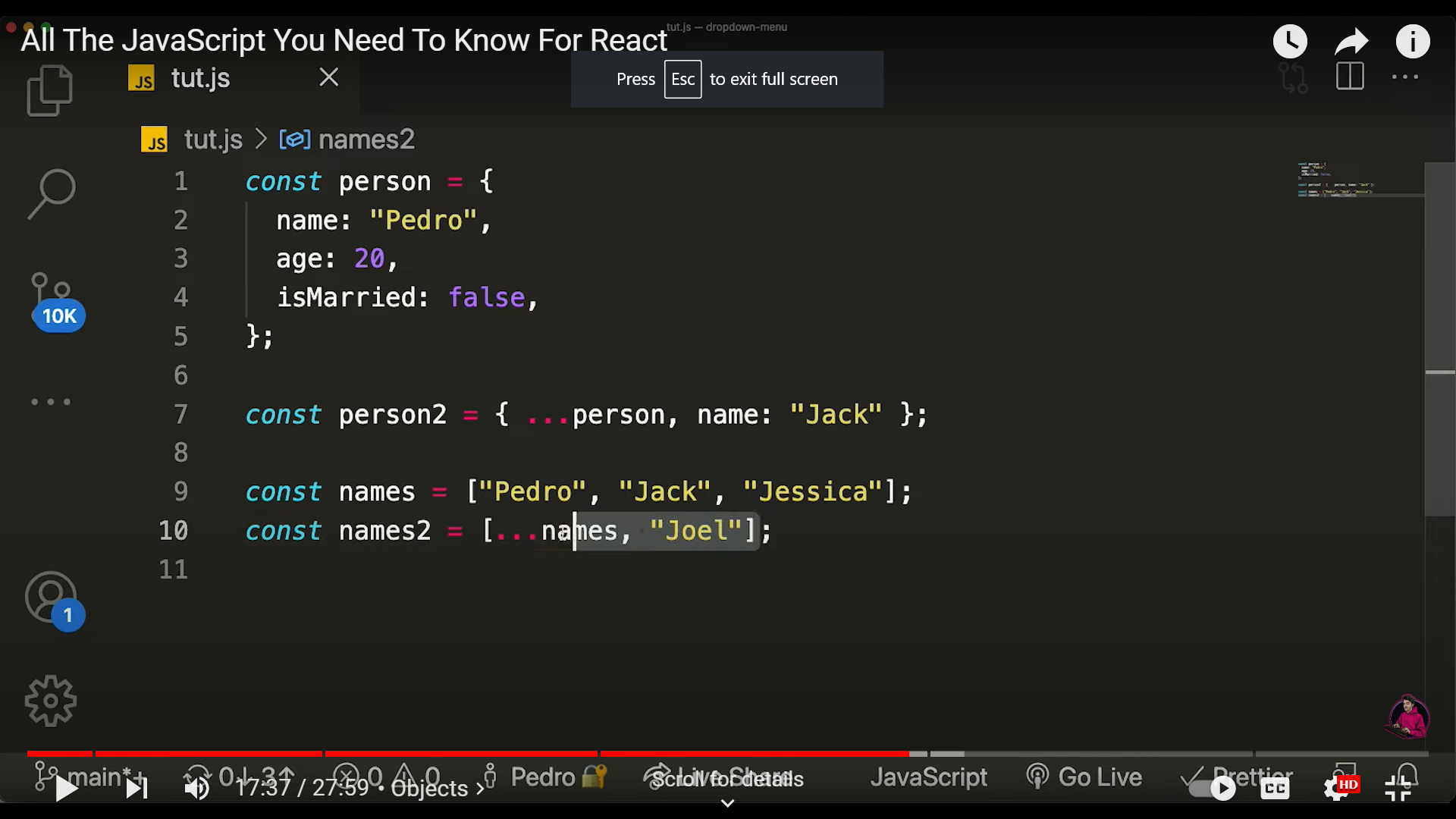
};

3.)

4.) When person2 is as same as person except the name





**REACT.JS**

Library – tool for specific functionality

Framework – set of tools for building apps eg : Bootstrap – open source framework for CSS

Dynamic and interactive user interfaces

Routing , HTTP , Managing app state, Internalization, Form validation , Animation

1.)node\_modules

->3rd party libraries like react & other tools are installed

2.)public

->images & videos

3.)assets

->app component

4.)filename.tsx(tsx for react components)

5.)**Class component :**

-> A class component must include the extends React.Component statement.

-> The component also requires a render() method, this method returns HTML.

class Car extends React.Component {

render() {

return <h2>Hi, I am a Car!</h2>;

}

}

6.)**Functional component :**

function Car() {

return <h2>Hi, I am a Car!</h2>;

}

export default Car;

->For React components use PascalCasing

App.tsx

import Message from './Message';

function App()

{

  return <div><Message /></div>;

}

export default App;

Message.tsx

function Message()

{

    const name='';

    if(name)

    return <h1>Hello {name}</h1>;

    else

    return <h1>Hello world</h1>;

}

 export default Message;

Virtual DOM:

App(Parent - div) ---🡪 Message (Child – h1)

**ELEMENTS IN REACT:**

Building blocks of React components

Smallest piece in react application

React elements – DOM nodes(HTML elements)

**COMPONENTS IN REACT**:

Building blocks of user interface

Components can be combined with other components = complex user interface

Reusable piece of code

Structure & behavior of web page / application

React Components – tree of DOM NODES

1. Functional Components: (Stateless)

* Defined using javascript functions
* Returns using react elements

1. Class Components: (Stateful)

* Defined using javascript class
* Returns react element
* Extends React.Component
* Implements render( ) method

import React from ‘react’;

function Employee()

{

return (<div>

<h1> Hello </h1>

<h1> Emp ID: </h1>

</div>)

}

export default Employee;

ReactDOM.render(<Employee/>,document.getElementById(‘root’)); // for calling the component

<div id=”root”></div>

1. React is going to take the user interface
2. This root component controls other components(header , footer, sidenav, main content) in app

**JSX: (Javascript XML)**

* When you use JSX, you can write HTML-like syntax in your JavaScript code, which makes it easier to visualize the UI and debug your code.
* const button = <button className="btn">Click me!</button>;

1. the **<button>** tag is actually a JSX element that creates a React component
2. The **className** attribute sets the class of the button element
3. the text inside the tags is the button's label.



**React Props:**

Props – properties

It allows us to pass information to a component

Props are used to pass the data from parent component to child component

**Function(Props)**

**Greet.js**

import React from 'react';

export const Greet = (props) =>

{

    return (

        <div>

        <h1> Hello {props.name}</h1>

        <h2> Emp Id: {props.empid} </h2>

        </div>

    )

}

export default Greet;

**App.js**

function App() {

  return (

    <div className="container">

      <Greet name='Raj' empid={1001}/>

</div>

  );

}

**Class (Props)= Welcome.js**

import React, {Component} from 'react'

class Welcome extends React.Component

{

    render()

    {

        return(

        <div>

            <h1> Welcome {this.props.name}</h1>

            <h2> Emp ID: {this.props.empid}</h2>

        </div>)

    }v

}

export default Welcome;

**App.js**

function App() {

  return (

    <div className="container">

      <Welcome name='Raj' empid={1001}/>

</div>

  );

}

PROPS & STATE:

