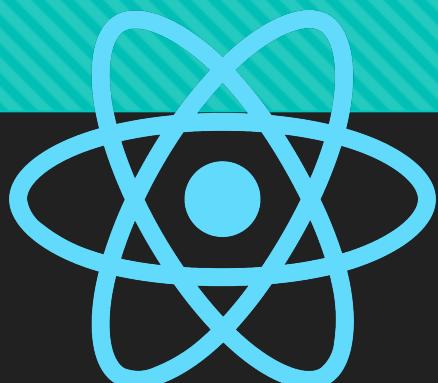


Modern Web Development Using React

Olgun DUTKAN – UI / UX Team in HAVELSAN
odutkan@havelsan.com.tr



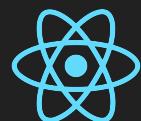
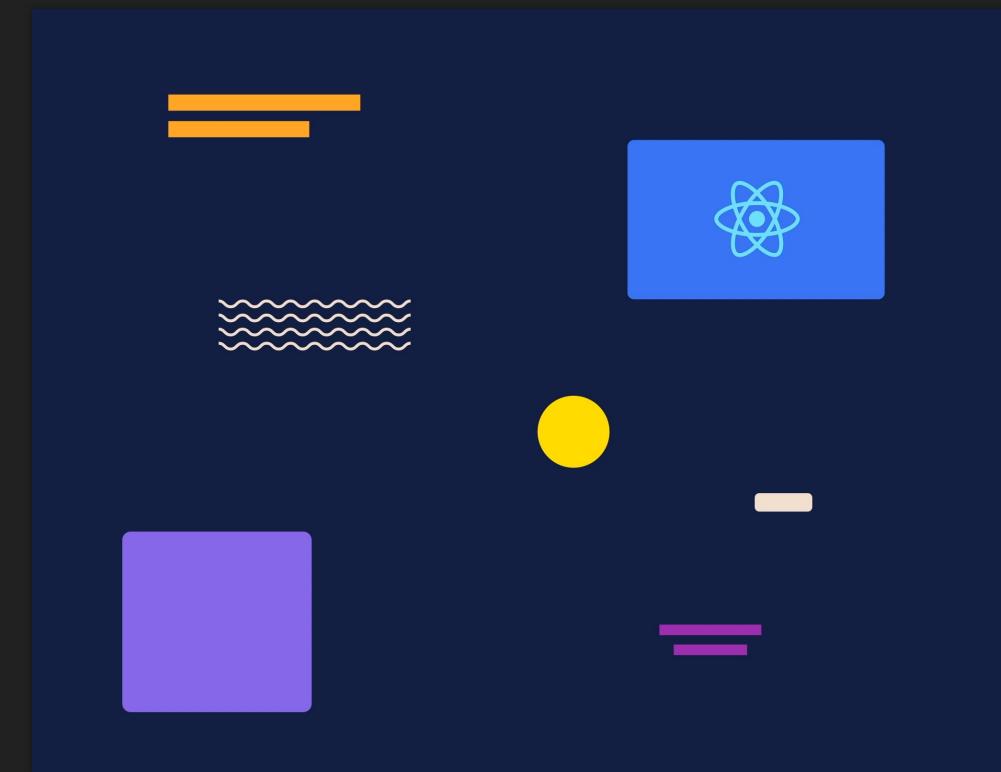
```
class HelloMessage extends React.Component {  
    render() {  
        return (  
            <div> Hello, I am {this.props.name} </div>  
        );  
    }  
}  
  
ReactDOM.render(  
    <HelloMessage name="Olgun DUTKAN" />,  
    document.getElementById('root')  
);
```

Agenda

1. Introduction to React
2. JSX
3. Components Interact
4. Lifecycle Methods
5. Hooks

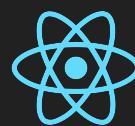
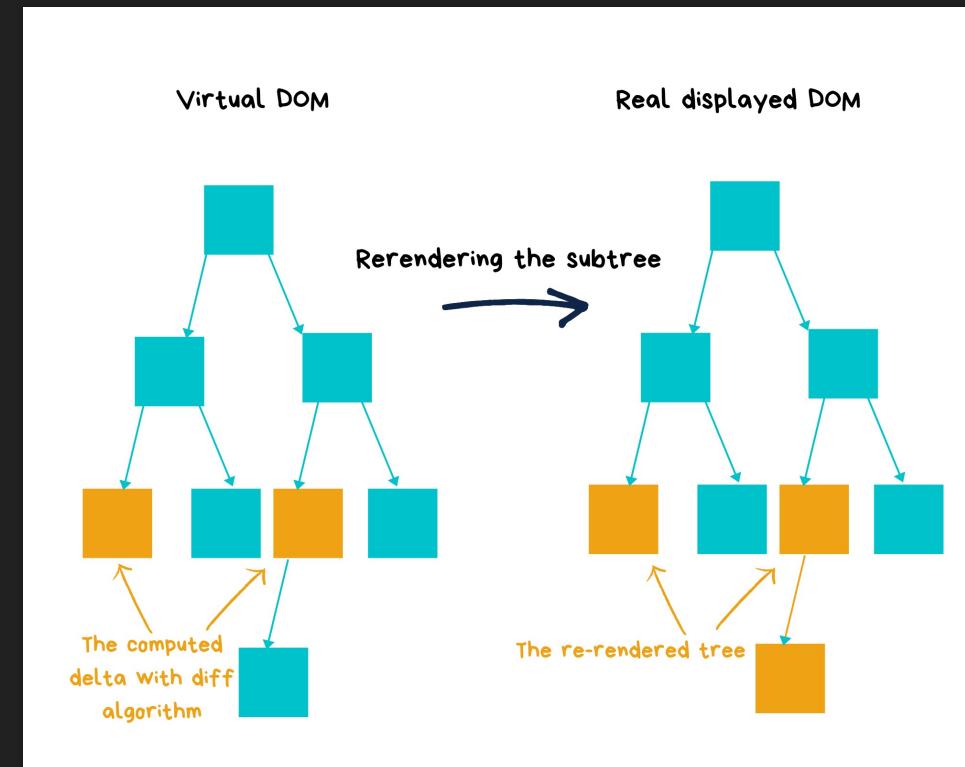
What is React?

- A JavaScript library for building user interfaces



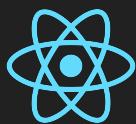
How does React Work?

- React creates a VIRTUAL DOM in memory.
- React only changes what needs to be changed!

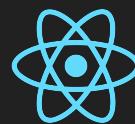


Why React?

- Simplicity
- Easy to learn
- Fast
- Modular
- Scalable
- Flexible



A brief history of React



Create React App

```
Day 2 — olgundutkan@Olguns-MacBook-Pro — ..lopment/Day 2 — zsh — 139x73
● ● ● Day 2 git:(main) ✘ node -v
v17.0.1
● ● ● Day 2 git:(main) ✘ npm -v
8.1.0
● ● ● Day 2 git:(main) ✘ npm install -g create-react-app
```



```
Day 2 — npx create-react-app my-todo-app --template typescript --use-npm — npx — npm install react react-dom react-scripts cra-template-typescript • npm exec create-react-app my...
● ● ● Day 2 git:(main) ✘ npx create-react-app my-todo-app --template typescript --use-npm
Creating a new React app in /Users/olgundutkan/Desktop/Modern Web Development/Day 2/my-todo-app.
Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template-typescript...
(( )) :: idealTree:yargs: sill placeDep ROOT yargs-parser@18.1.3 OK for: yargs@15.4.1 want: ^18.1.2
```



Run the React Application

```
my-todo-app — olgundutkan@Olguns-MacBook-Pro — ..2/my-todo-app — zsh — 139x73
→ Day 2 git:(main) ✘ cd my-todo-app
→ my-todo-app git:(main) ✘ npm run start
```

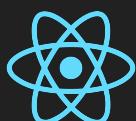


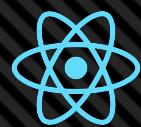
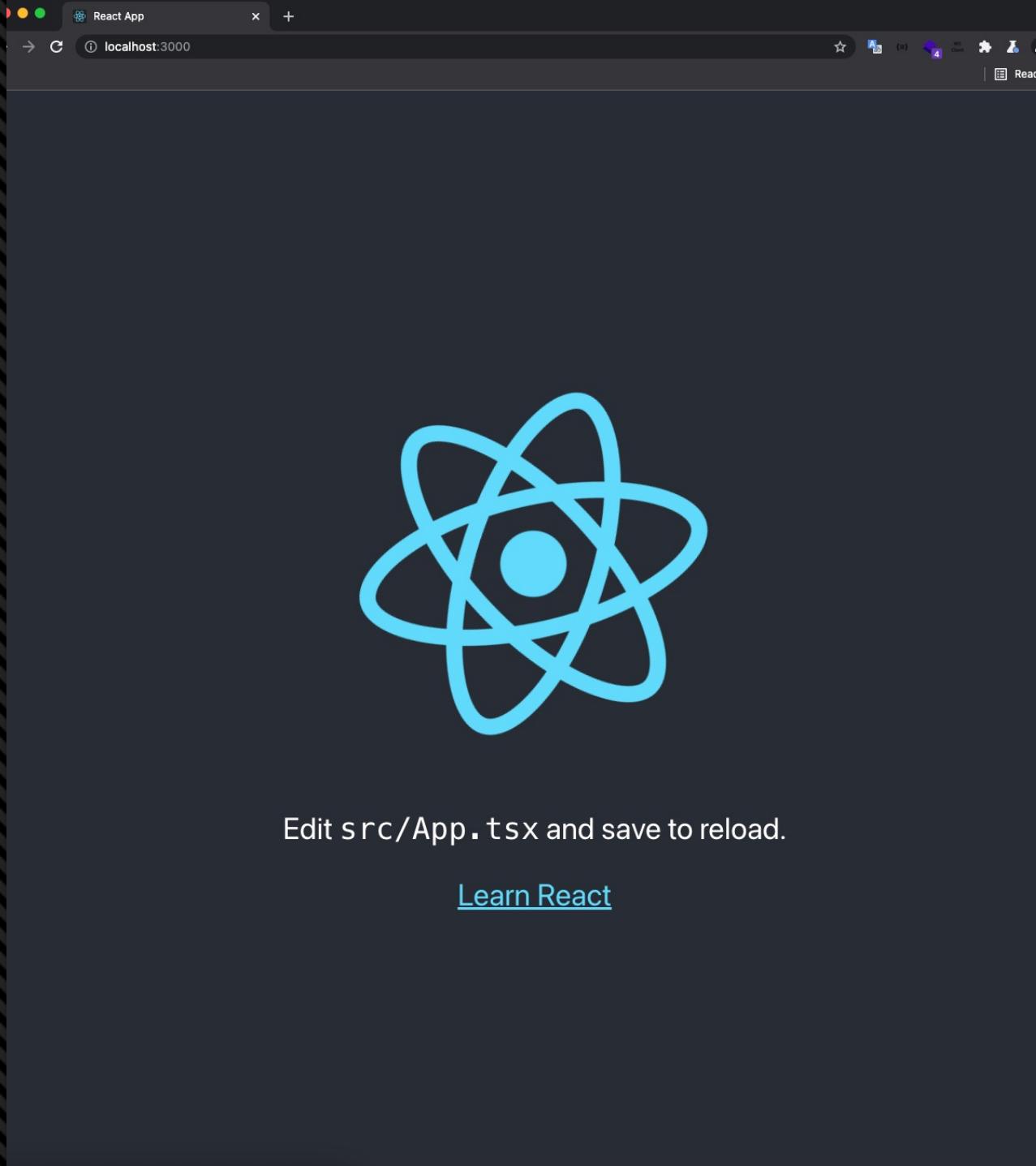
```
my-todo-app — npm run start — npm — node < npm run start TMPDIR=/var/folders/6c/xzqddvts1571bw05rgpkzdzh0000gn/T/ __CFBundleIdentifier=com.apple.Terminal XPC_FLAGS=0x0...
Compiled successfully!

You can now view my-todo-app in the browser.

Local:          http://localhost:3000
On Your Network: http://192.168.1.102:3000

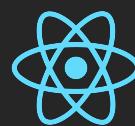
Note that the development build is not optimized.
To create a production build, use npm run build.
```





React Render HTML

1.8



The image shows a screenshot of the Visual Studio Code (VS Code) interface. It has two main panes. The left pane displays the file 'index.html' with its code content. The right pane displays the file 'index.tsx' with its code content. Both files are part of a project named 'my-todo-app'. The 'index.html' file contains standard HTML code, including meta tags, a title, and a script tag. The 'index.tsx' file is a React component that imports 'React' and 'ReactDOM' from 'react' and 'react-dom' respectively, and renders a 'App' component. The code is annotated with comments explaining the build process and deployment.

```
index.html (left pane)
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/.� favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta name="description" content="Web site created using create-react-app" />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <!--
      manifest.json provides metadata used when your web app is installed on a user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      It will be replaced with the URL of the 'public' folder during the build.
      Only files inside the 'public' folder can be referenced from the HTML.
    -->
    <!--
      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will work correctly both with client-side routing and a non-root public URL.
      Learn how to configure a non-root public URL by running 'npm run build'.
    -->
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty page.
    -->
    <!--
      You can add webfonts, meta tags, or analytics to this file.
      The build step will place the bundled scripts into the <body> tag.
    -->
    <!--
      To begin the development, run 'npm start' or 'yarn start'.
      To create a production bundle, use 'npm run build' or 'yarn build'.
    -->
  </body>
</html>
```

```
index.tsx (right pane)
```

```
src > index.tsx
1 import React from 'react';
2 import ReactDOM from 'react-dom';
3 import './index.css';
4 import App from './App';
5 import reportWebVitals from './reportWebVitals';
6
7 ReactDOM.render(
8   <React.StrictMode>
9     | <App />
10    </React.StrictMode>
11    document.getElementById('root')
12 );
13
14 // If you want to start measuring performance in your app, pass a function
15 // to log results (for example: reportWebVitals(console.log))
16 // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
17 reportWebVitals();
18
```

App.tsx — my-todo-app

EXPLORER

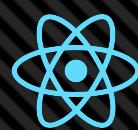
MY-TODO-APP

- > node_modules
- public
 - favicon.ico
 - index.html
 - logo192.png
 - logo512.png
 - manifest.json
 - robots.txt
- src
 - App.css
 - App.test.tsx
 - App.tsx
 - index.css
 - index.tsx
 - logo.svg
 - react-app-env.d.ts
 - reportWebVitals.ts
 - setupTests.ts
- .gitignore
- package-lock.json
- package.json
- README.md
- tsconfig.json

index.tsx U index.html U App.tsx U

src > App.tsx > ...

```
1 import React from 'react';
2 import logo from './logo.svg';
3 import './App.css';
4
5 function App() {
6   return (
7     <div className="App">
8       <header className="App-header">
9         <img src={logo} className="App-logo" alt="logo" />
10        <p>
11          | Edit <code>src/App.tsx</code> and save to reload.
12        </p>
13        <a
14          | className="App-link"
15          | href="https://reactjs.org"
16          | target="_blank"
17          | rel="noopener noreferrer"
18        >
19          | Learn React
20        </a>
21      </header>
22    </div>
23  );
24}
25
26 export default App;
```



```
console.info(`Review`)  
// Review
```

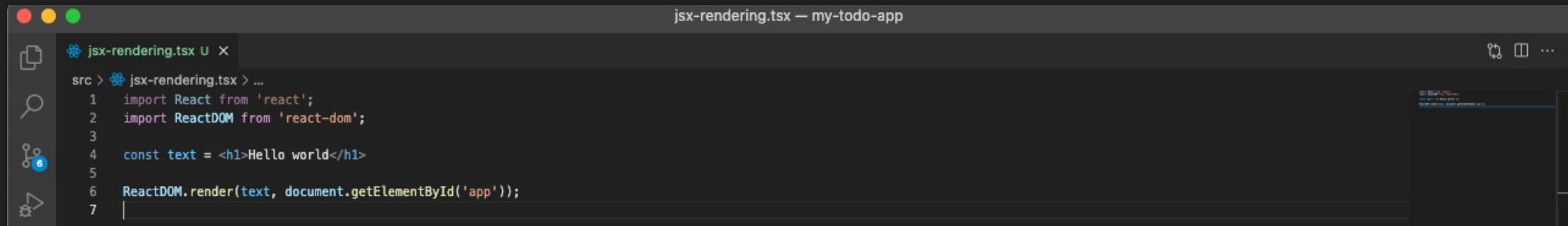
What is JSX?

- JSX stands for JavaScript XML.
- JSX allows us to write HTML in React.
- JSX makes it easier to write and add HTML in React.

```
const element = <h1>Hello, world!</h1>;
```



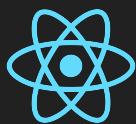
JSX Rendering



A screenshot of a code editor window titled "jsx-rendering.tsx — my-todo-app". The file contains the following code:

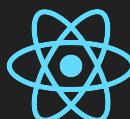
```
jsx-rendering.tsx
src > jsx-rendering.tsx > ...
1 import React from 'react';
2 import ReactDOM from 'react-dom';
3
4 const text = <h1>Hello world</h1>
5
6 ReactDOM.render(text, document.getElementById('app'));
7 |
```

The code imports React and ReactDOM, defines a variable text containing an h1 element with the text "Hello world", and then renders it into the element with id "app". The code editor interface includes a sidebar with icons for file operations like copy, search, and refresh.

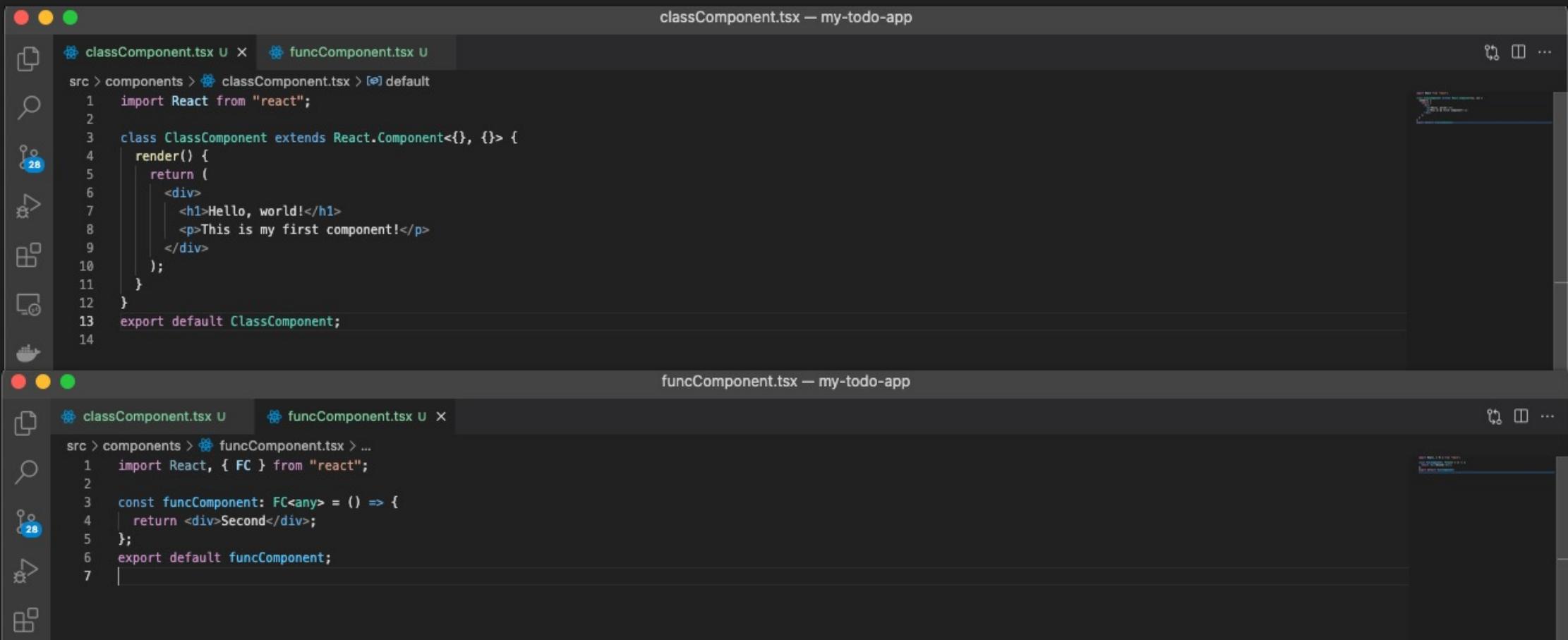


React Components

- Components are independent and reusable bits of code. They serve the same purpose as JavaScript functions, but work in isolation and return HTML.
- Components come in two types, Class components and Function components, in this tutorial we will concentrate on Function components.



Create Your First Component



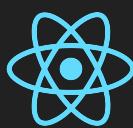
The image shows a dark-themed code editor with two tabs open: `classComponent.tsx` and `funcComponent.tsx`, both under the `src > components` directory of a project named `my-todo-app`.

classComponent.tsx — my-todo-app

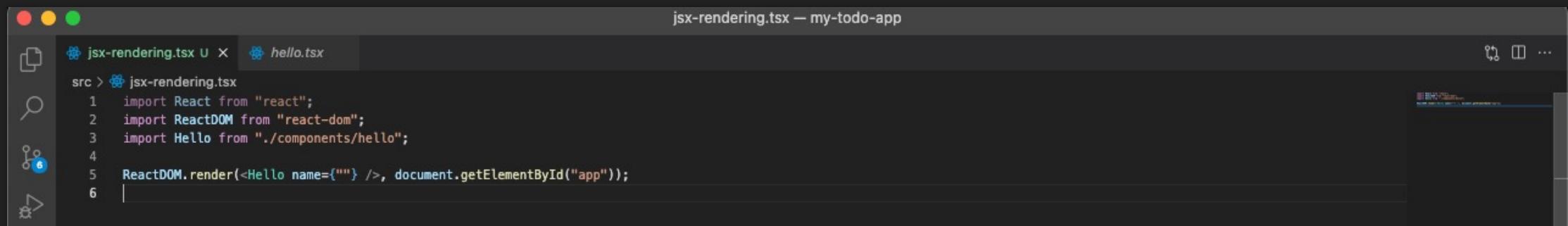
```
src > components > classComponent.tsx > [e] default
1 import React from "react";
2
3 class ClassComponent extends React.Component<{}, {}> {
4   render() {
5     return (
6       <div>
7         <h1>Hello, world!</h1>
8         <p>This is my first component!</p>
9       </div>
10    );
11  }
12}
13 export default ClassComponent;
14
```

funcComponent.tsx — my-todo-app

```
src > components > funcComponent.tsx > ...
1 import React, { FC } from "react";
2
3 const funcComponent: FC<any> = () => {
4   return <div>Second</div>;
5 };
6 export default funcComponent;
7
```



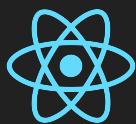
Component Rendering



A screenshot of a code editor window titled "jsx-rendering.tsx — my-todo-app". The window shows two tabs: "jsx-rendering.tsx" and "hello.tsx". The "jsx-rendering.tsx" tab is active and contains the following code:

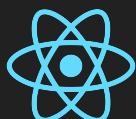
```
src > jsx-rendering.tsx
1 import React from "react";
2 import ReactDOM from "react-dom";
3 import Hello from "./components/hello";
4
5 ReactDOM.render(<Hello name="" />, document.getElementById("app"));
6 |
```

The code imports React and ReactDOM, and then renders a Hello component with an empty name prop into the DOM element with id "app". There are 6 code changes indicated in the sidebar.



React Props

- Props are arguments passed into React components.
- Props are passed to components via HTML attributes.



hello.tsx U X

```
src > components > hello.tsx > ...
1 import React, { FC } from "react";
2
3 interface IHello {
4   name: string;
5 }
6
7 const Hello: FC<IHello> = ({name}) =>
8   return <div>Hello, I am {name}</div>;
9
10 export default Hello;
```

App.tsx U

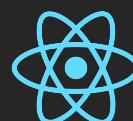
hello.tsx U

App.tsx U X

```
c > App.tsx > ...
1
```

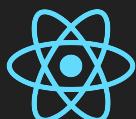
```
1 import Hello from './components/hello';
2
3 function App() {
4   return (
5     <Hello name="Olgun DUTKAN"/>
6   );
7 }
8
9 export default App;
10
```

React Props

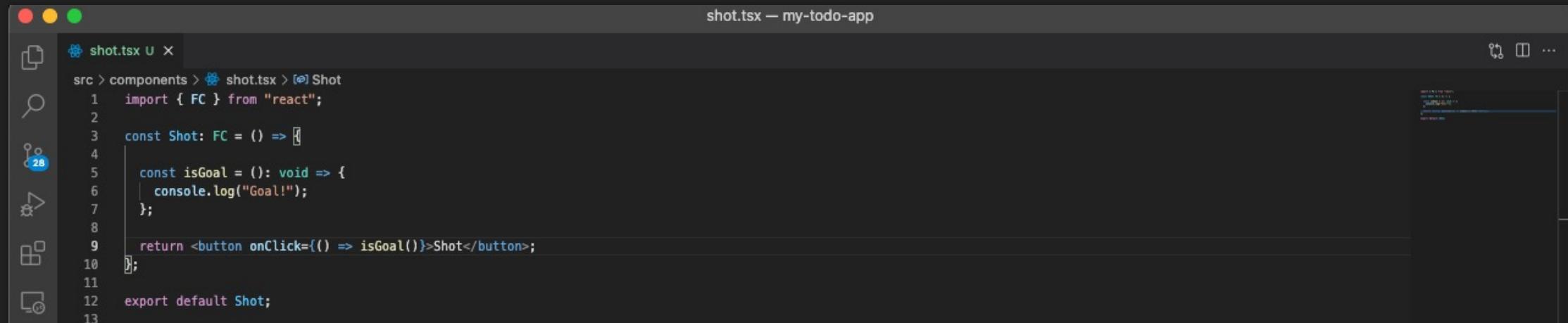


React Events

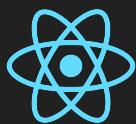
- Just like HTML DOM events, React can perform actions based on user events.
- React has the same events as HTML: click, change, mouseover etc.



React Events

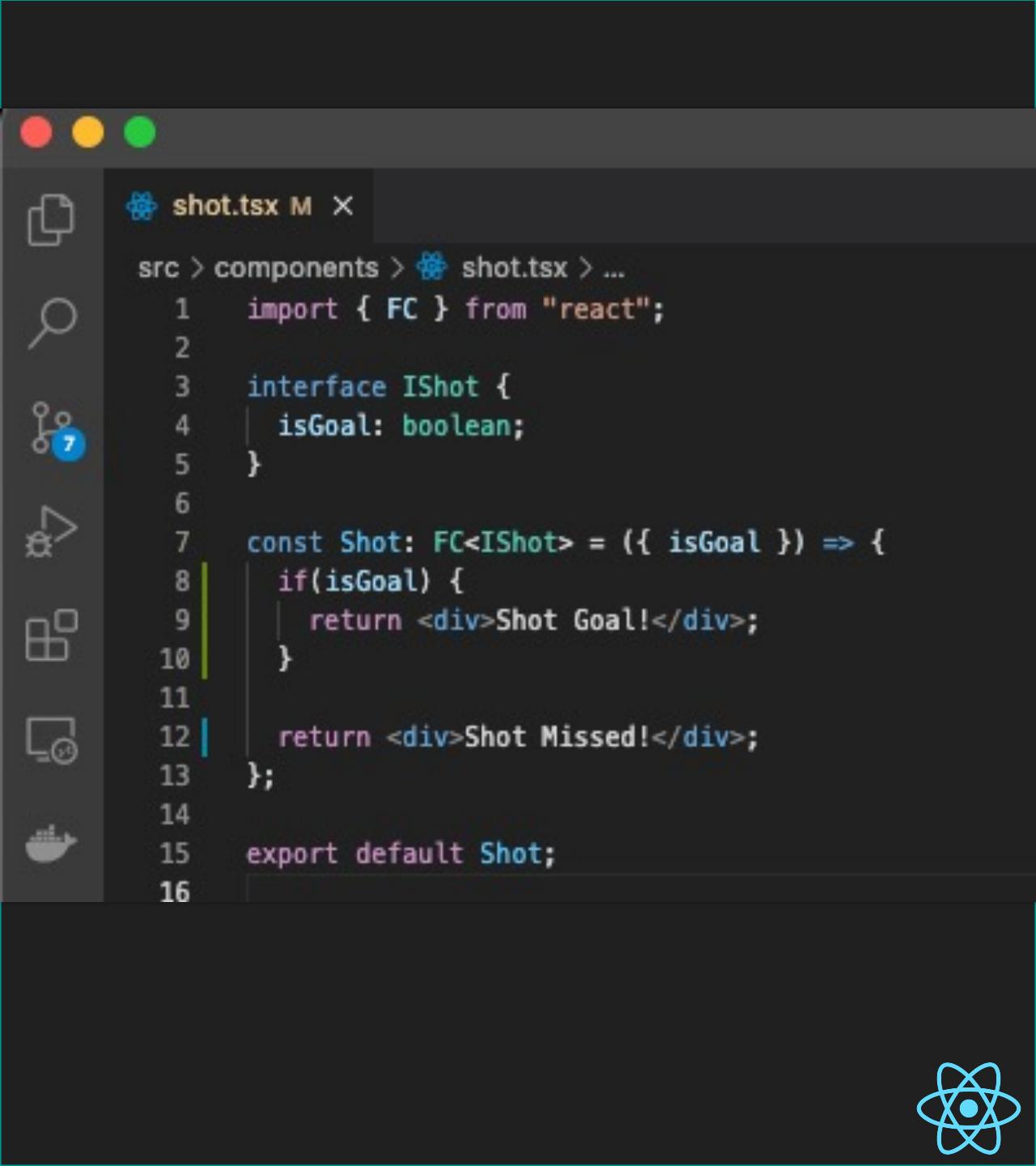


```
shot.tsx — my-todo-app
src > components > shot.tsx > Shot
1 import { FC } from "react";
2
3 const Shot: FC = () => [
4   const isGoal = (): void => {
5     console.log("Goal!");
6   };
7
8   return <button onClick={() => isGoal()}>Shot</button>;
9 ];
10
11
12 export default Shot;
13
```



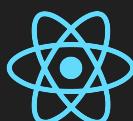
React Conditional Rendering

if Statement



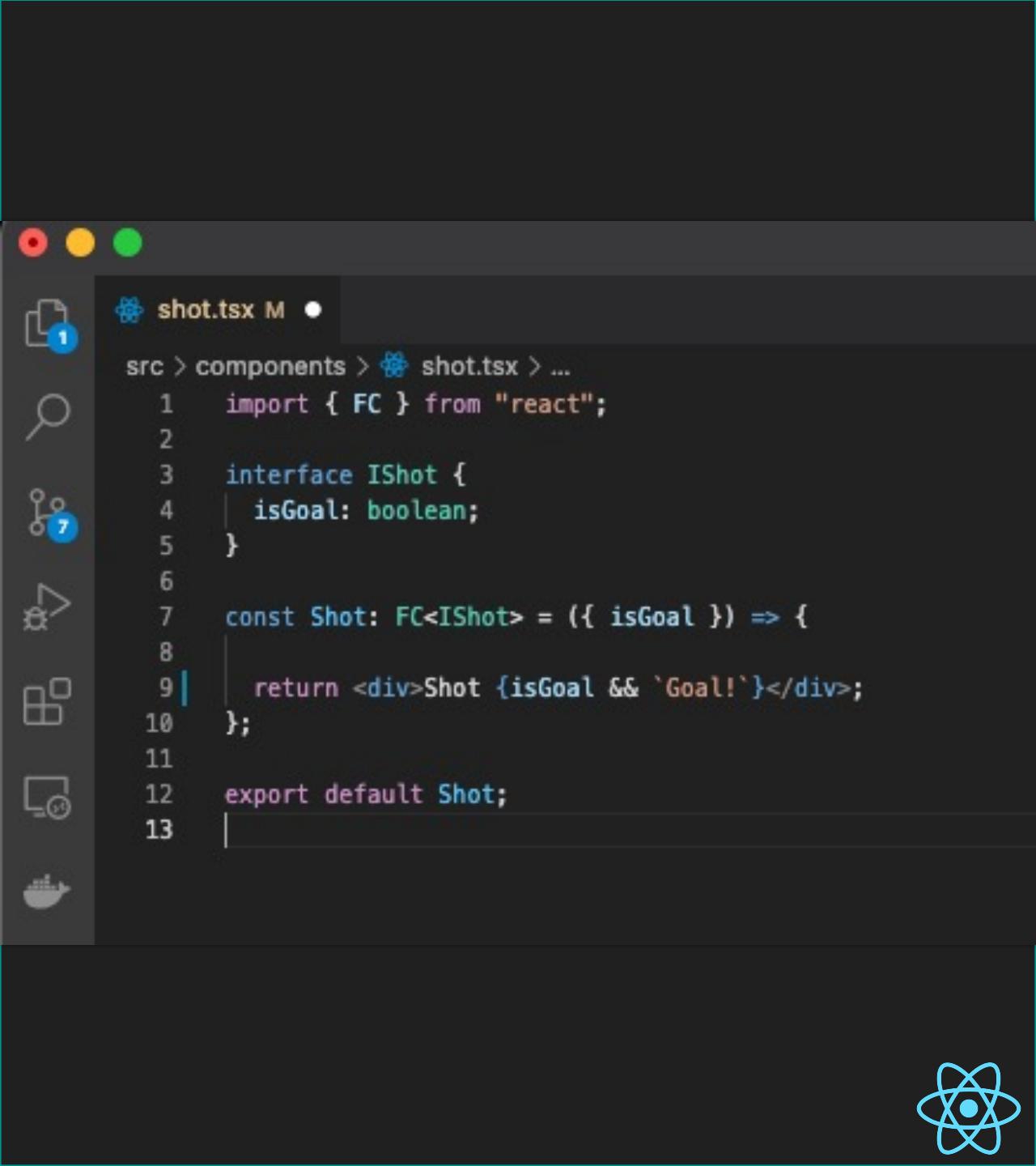
A screenshot of a code editor showing a file named `shot.tsx`. The code implements conditional rendering based on the `isGoal` prop. If `isGoal` is true, it returns `<div>Shot Goal!</div>`; otherwise, it returns `<div>Shot Missed!</div>`. The code editor interface includes a sidebar with icons for file operations like copy, search, and refresh.

```
src > components > shot.tsx > ...
1 import { FC } from "react";
2
3 interface IShot {
4   isGoal: boolean;
5 }
6
7 const Shot: FC<IShot> = ({ isGoal }) => {
8   if(isGoal) {
9     return <div>Shot Goal!</div>;
10  }
11
12  return <div>Shot Missed!</div>;
13}
14
15 export default Shot;
16
```

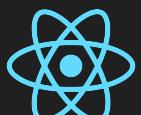


React Conditional Rendering

Logical && Operator



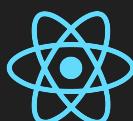
```
shot.tsx M ...
src > components > shot.tsx > ...
1 import { FC } from "react";
2
3 interface IShot {
4   isGoal: boolean;
5 }
6
7 const Shot: FC<IShot> = ({ isGoal }) => {
8
9   return <div>Shot {isGoal && `Goal!`}</div>;
10
11
12 export default Shot;
13 }
```



React Conditional Rendering

Logical && Operator

```
shot.tsx U X App.tsx U
src > components > shot.tsx > ...
1 import { FC } from "react";
2
3 interface IShot {
4   isGoal: boolean;
5 }
6
7 const Shot: FC<IShot> = ({ isGoal }) => {
8
9   return <div>Shot {isGoal ? 'Goal!' : 'Missed!'}</div>;
10};
11
12 export default Shot;
13
```



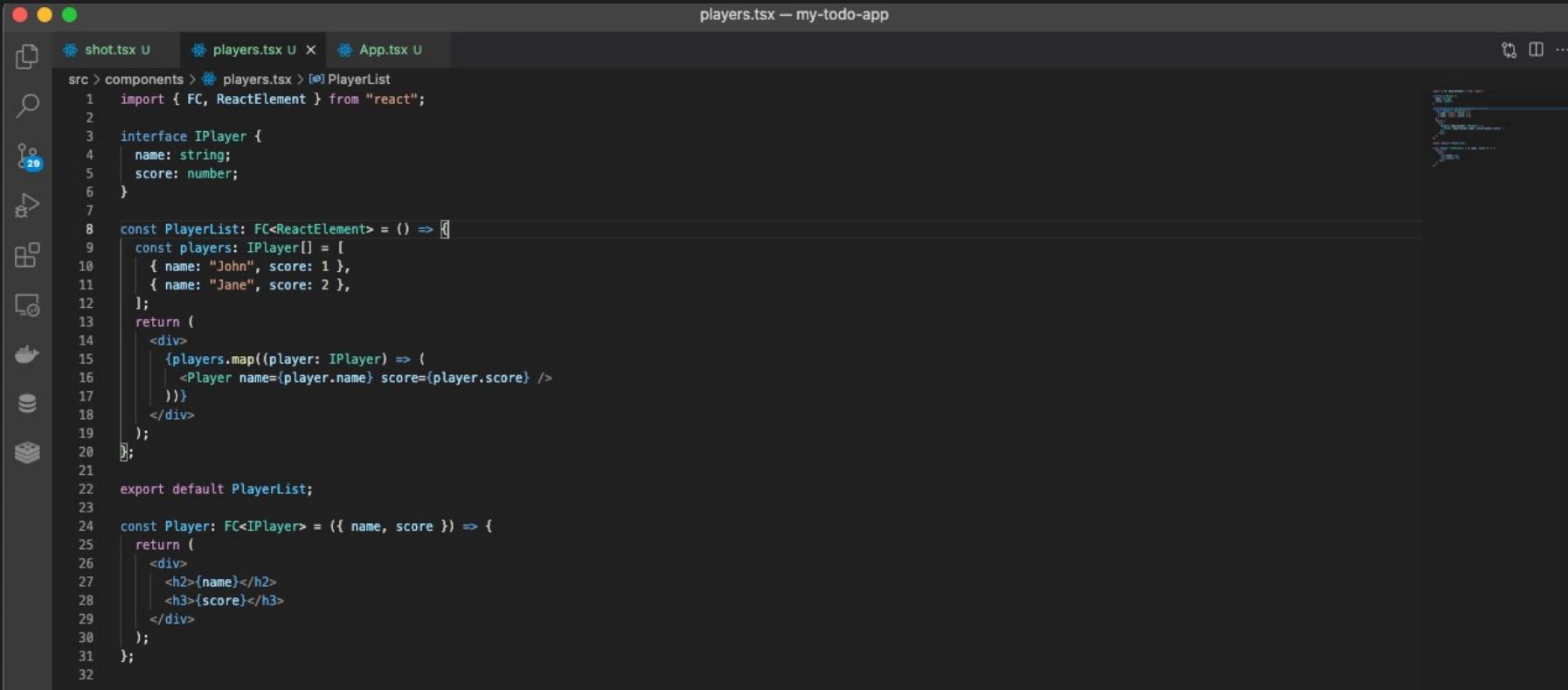
React Conditional Rendering

if Statement

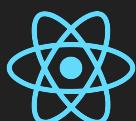
```
shot.tsx U      App.tsx U X
src > App.tsx > ...
1  import Hello from "./components/hello";
2  import Shot from "./components/shot";
3
4  function App() {
5    return (
6      <div className="App">
7        <Hello name="Olgun DUTKAN" />
8        <Shot isGoal={true} />
9      </div>
10 );
11 }
12
13 export default App;
14
```



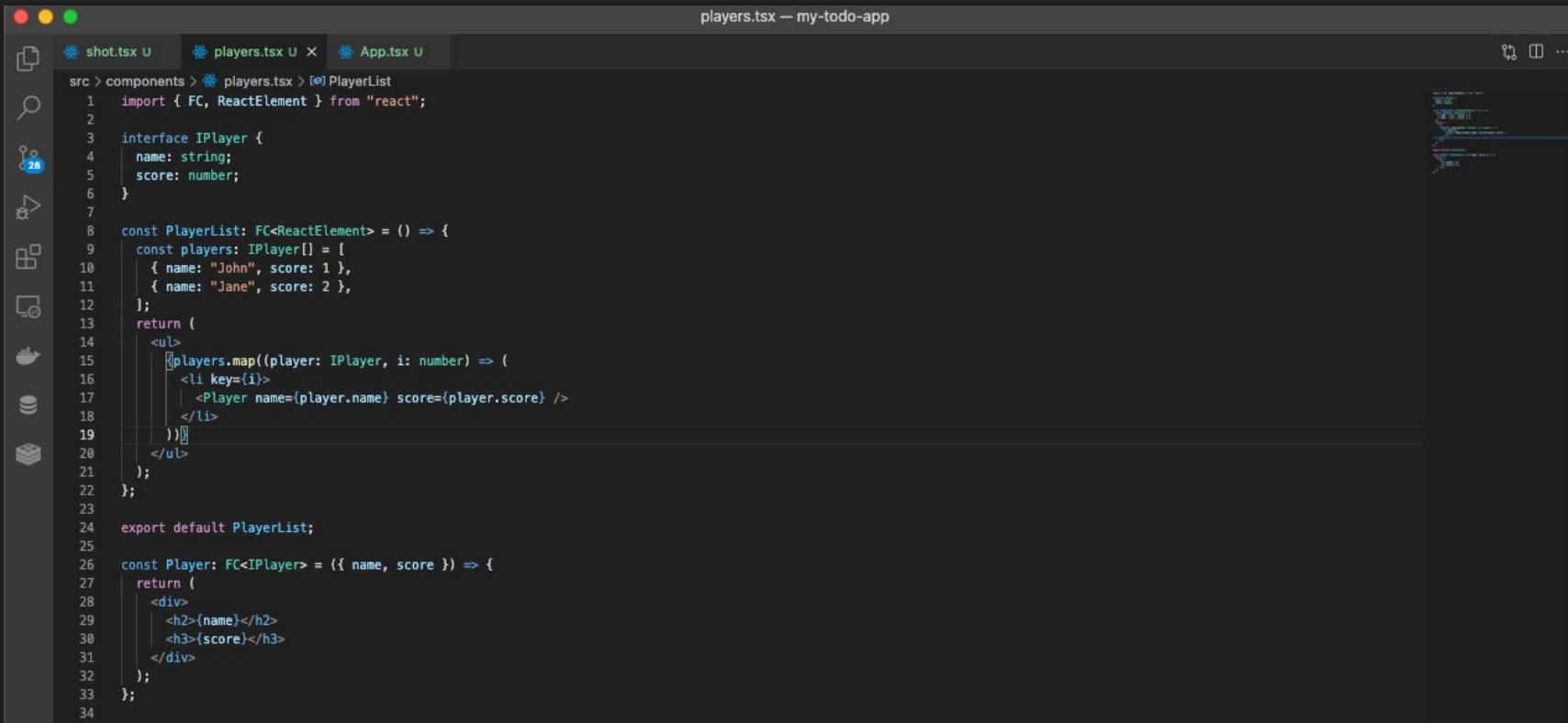
React Lists



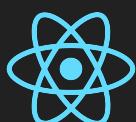
```
players.tsx — my-todo-app
src > components > players.tsx > [PlayerList]
  1  import { FC, ReactElement } from "react";
  2
  3  interface IPlayer {
  4    name: string;
  5    score: number;
  6  }
  7
  8  const PlayerList: FC<ReactElement> = () => [
  9    const players: IPlayer[] = [
 10      { name: "John", score: 1 },
 11      { name: "Jane", score: 2 },
 12    ];
 13    return (
 14      <div>
 15        {players.map((player: IPlayer) => (
 16          <Player name={player.name} score={player.score} />
 17        ))}
 18      </div>
 19    );
 20  ];
 21
 22  export default PlayerList;
 23
 24  const Player: FC<IPlayer> = ({ name, score }) => {
 25    return (
 26      <div>
 27        <h2>{name}</h2>
 28        <h3>{score}</h3>
 29      </div>
 30    );
 31  };
 32
```



Keys



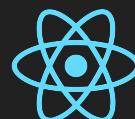
```
players.tsx — my-todo-app
src > components > players.tsx > [e] PlayerList
  1 import { FC, ReactElement } from "react";
  2
  3 interface IPlayer {
  4   name: string;
  5   score: number;
  6 }
  7
  8 const PlayerList: FC<ReactElement> = () => {
  9   const players: IPlayer[] = [
 10     { name: "John", score: 1 },
 11     { name: "Jane", score: 2 },
 12   ];
 13   return (
 14     <ul>
 15       <players.map((player: IPlayer, i: number) => (
 16         <li key={i}>
 17           <Player name={player.name} score={player.score} />
 18         </li>
 19       ))</ul>
 20     );
 21   );
 22 }
 23
 24 export default PlayerList;
 25
 26 const Player: FC<IPlayer> = ({ name, score }) => {
 27   return (
 28     <div>
 29       <h2>{name}</h2>
 30       <h3>{score}</h3>
 31     </div>
 32   );
 33 };
 34 }
```



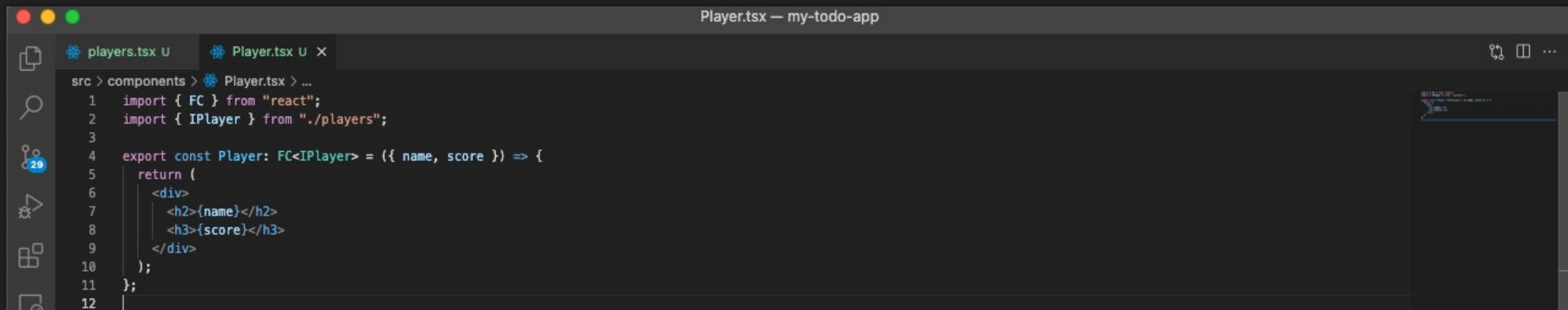
```
console.info(`Review`)  
// Review
```

Components Interact

- A React application can contain dozens, or even hundreds, of components.
- Each component might be small and relatively unremarkable on its own. When combined, however, they can form enormous, fantastically complex ecosystems of information.
- In other words, React apps are made out of components, but what makes React special isn't components themselves. What makes React special is the ways in which components interact.



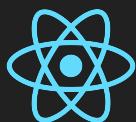
Component Render Other Component



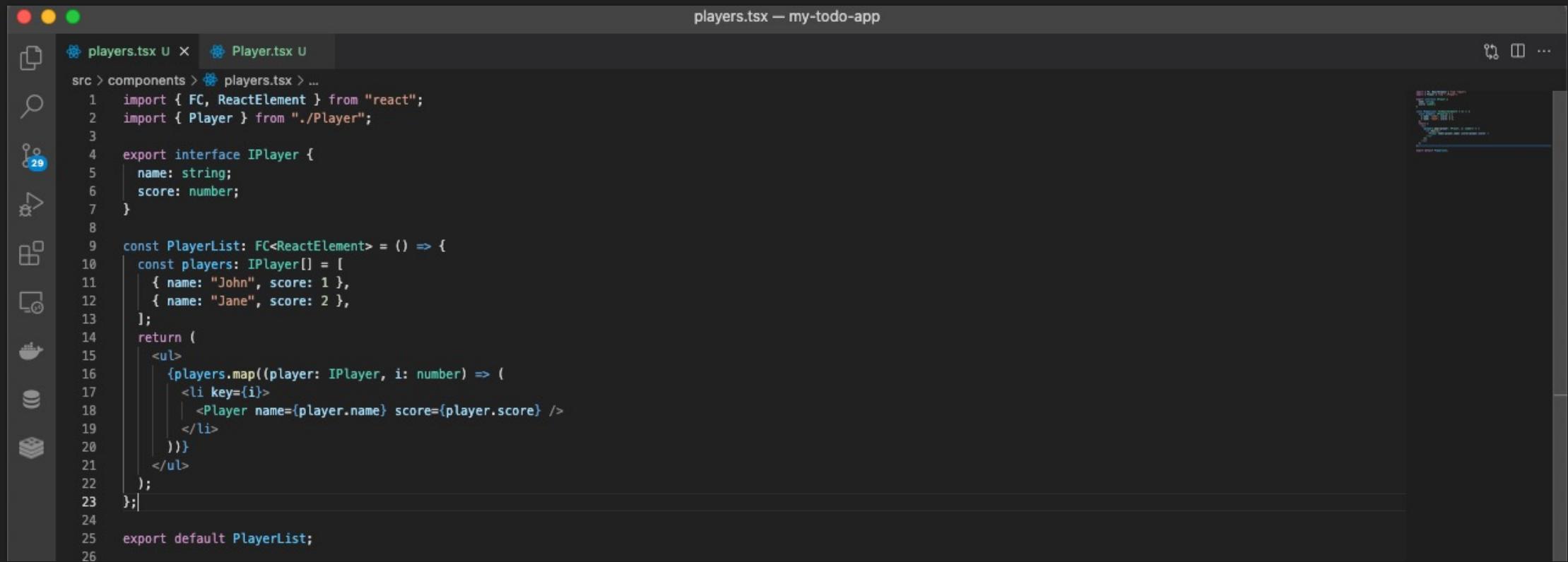
A screenshot of a dark-themed code editor window titled "Player.tsx — my-todo-app". The window shows two tabs: "players.tsx" and "Player.tsx". The "Player.tsx" tab is active, displaying the following code:

```
src > components > Player.tsx > ...
1 import { FC } from "react";
2 import { IPlayer } from "./players";
3
4 export const Player: FC<IPlayer> = ({ name, score }) => {
5   return (
6     <div>
7       <h2>{name}</h2>
8       <h3>{score}</h3>
9     </div>
10  );
11};
12|
```

The code defines a functional component named "Player" that takes "name" and "score" as props. It returns a `<div>` element containing an `<h2>` and an `<h3>`. The "name" prop is passed to the `<h2>` and the "score" prop is passed to the `<h3>`.



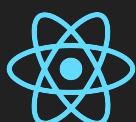
Component Render Other Component



The screenshot shows a dark-themed code editor with two tabs open: "players.tsx" and "Player.tsx". The "players.tsx" tab is active, displaying the following code:

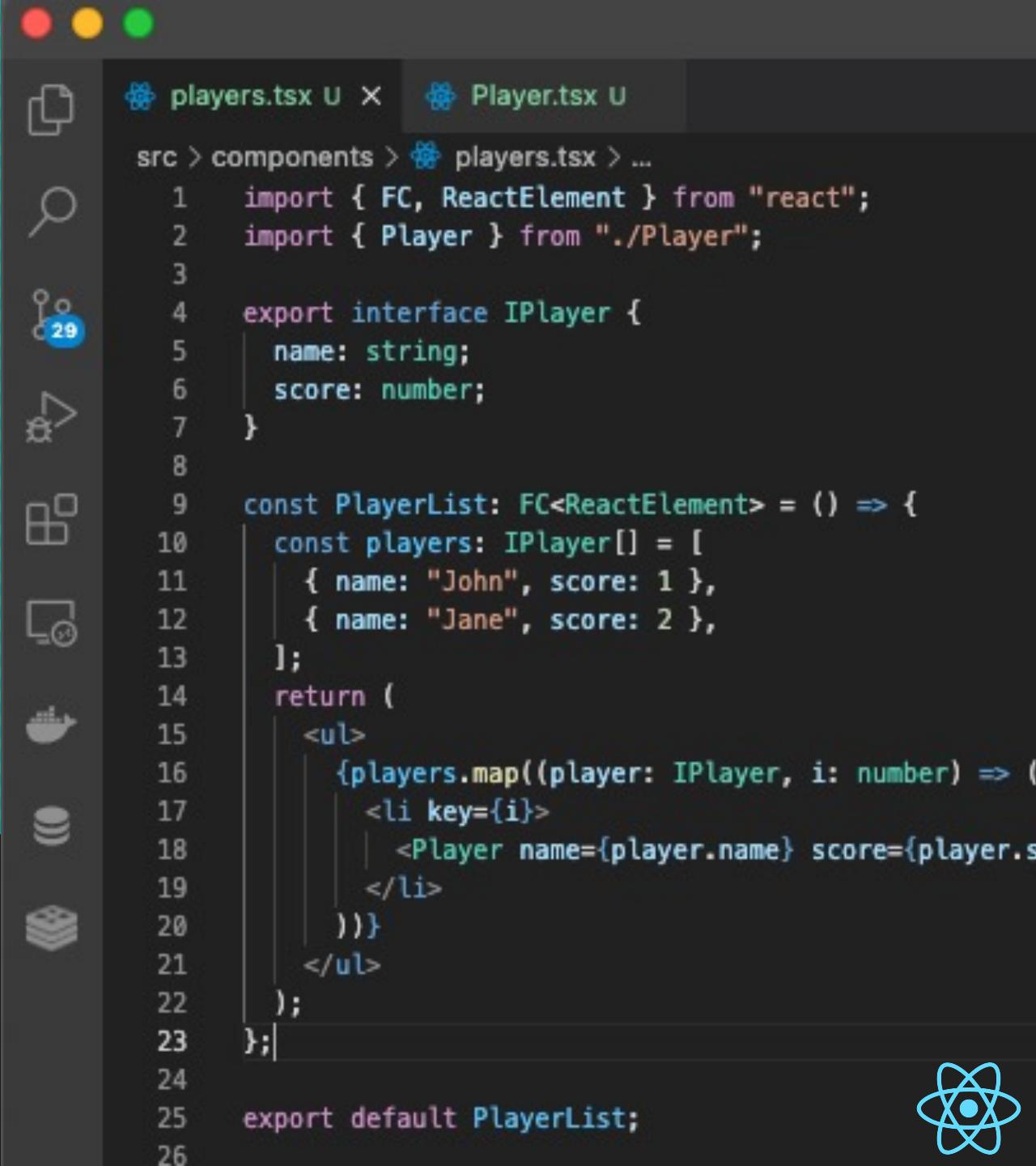
```
players.tsx — my-todo-app
src > components > players.tsx > ...
1 import { FC, ReactElement } from "react";
2 import { Player } from "./Player";
3
4 export interface IPlayer {
5   name: string;
6   score: number;
7 }
8
9 const PlayerList: FC<ReactElement> = () => {
10   const players: IPlayer[] = [
11     { name: "John", score: 1 },
12     { name: "Jane", score: 2 },
13   ];
14   return (
15     <ul>
16       {players.map((player: IPlayer, i: number) => (
17         <li key={i}>
18           <Player name={player.name} score={player.score} />
19         </li>
20       ))}
21     </ul>
22   );
23 };
24
25 export default PlayerList;
26
```

The "Player.tsx" tab is visible in the background, indicating it is another component file in the project.



this.props

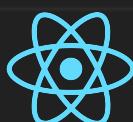
A component can pass *information* to another component.



The image shows a dark-themed code editor interface with two tabs open: "players.tsx U" and "Player.tsx U". The left sidebar contains icons for file operations like copy, search, and refresh, with a blue badge showing the number 29. The "players.tsx" file contains the following code:

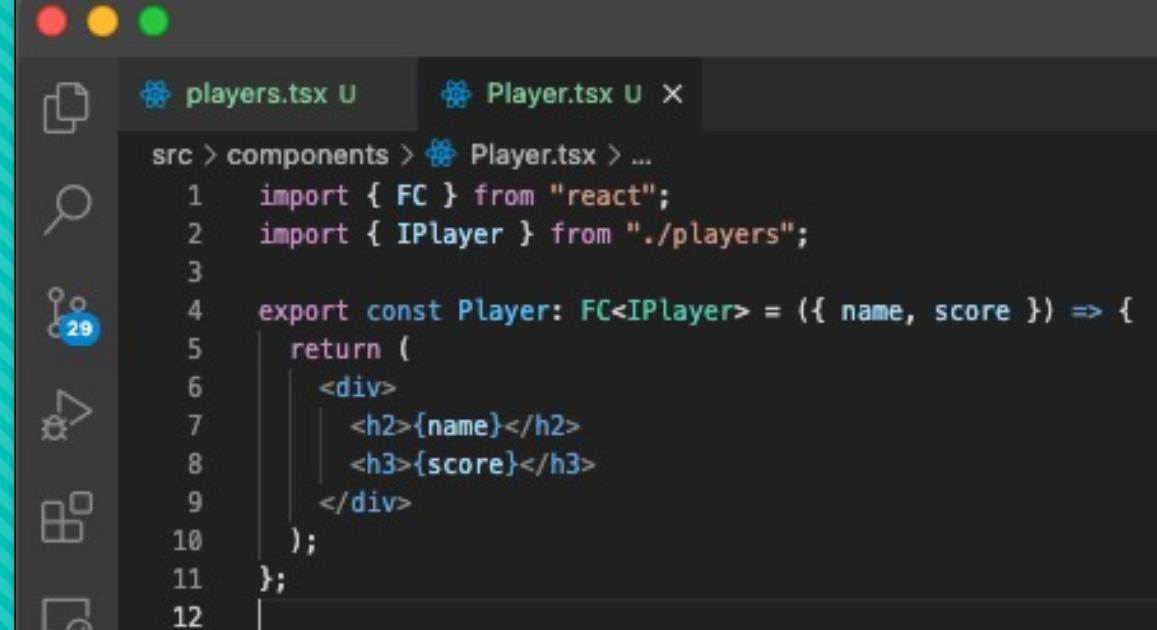
```
src > components > players.tsx > ...
1 import { FC, ReactElement } from "react";
2 import { Player } from "./Player";
3
4 export interface IPlayer {
5   name: string;
6   score: number;
7 }
8
9 const PlayerList: FC<ReactElement> = () => {
10   const players: IPlayer[] = [
11     { name: "John", score: 1 },
12     { name: "Jane", score: 2 },
13   ];
14   return (
15     <ul>
16       {players.map((player: IPlayer, i: number) => (
17         <li key={i}>
18           <Player name={player.name} score={player.score} />
19         </li>
20       ))}
21     </ul>
22   );
23 };
24
25 export default PlayerList;
```

The "Player.tsx" file is partially visible at the top right of the editor window.



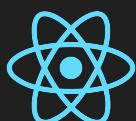
this.props

A component can pass *information* to another component.



The image shows a dark-themed code editor window with two tabs open: "players.tsx" and "Player.tsx". The "Player.tsx" tab is active, showing the following code:

```
src > components > Player.tsx > ...
1 import { FC } from "react";
2 import { IPlayer } from "./players";
3
4 export const Player: FC<IPlayer> = ({ name, score }) => {
5   return (
6     <div>
7       <h2>{name}</h2>
8       <h3>{score}</h3>
9     </div>
10  );
11}
12
```

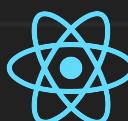


defaultProps

A component can pass *information* to another component.

players.tsx U Player.tsx U X

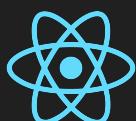
```
src > components > Player.tsx > ...
1 import { FC } from "react";
2 import { IPlayer } from "./players"
3
4 export const Player: FC<IPlayer> =
5   return (
6     <div>
7       <h2>{name}</h2>
8       <h3>{score}</h3>
9     </div>
10    );
11  };
12
13 Player.defaultProps = [
14   name: "Anonymous",
15   score: 0,
16 ] as IPlayer;
```



this.state

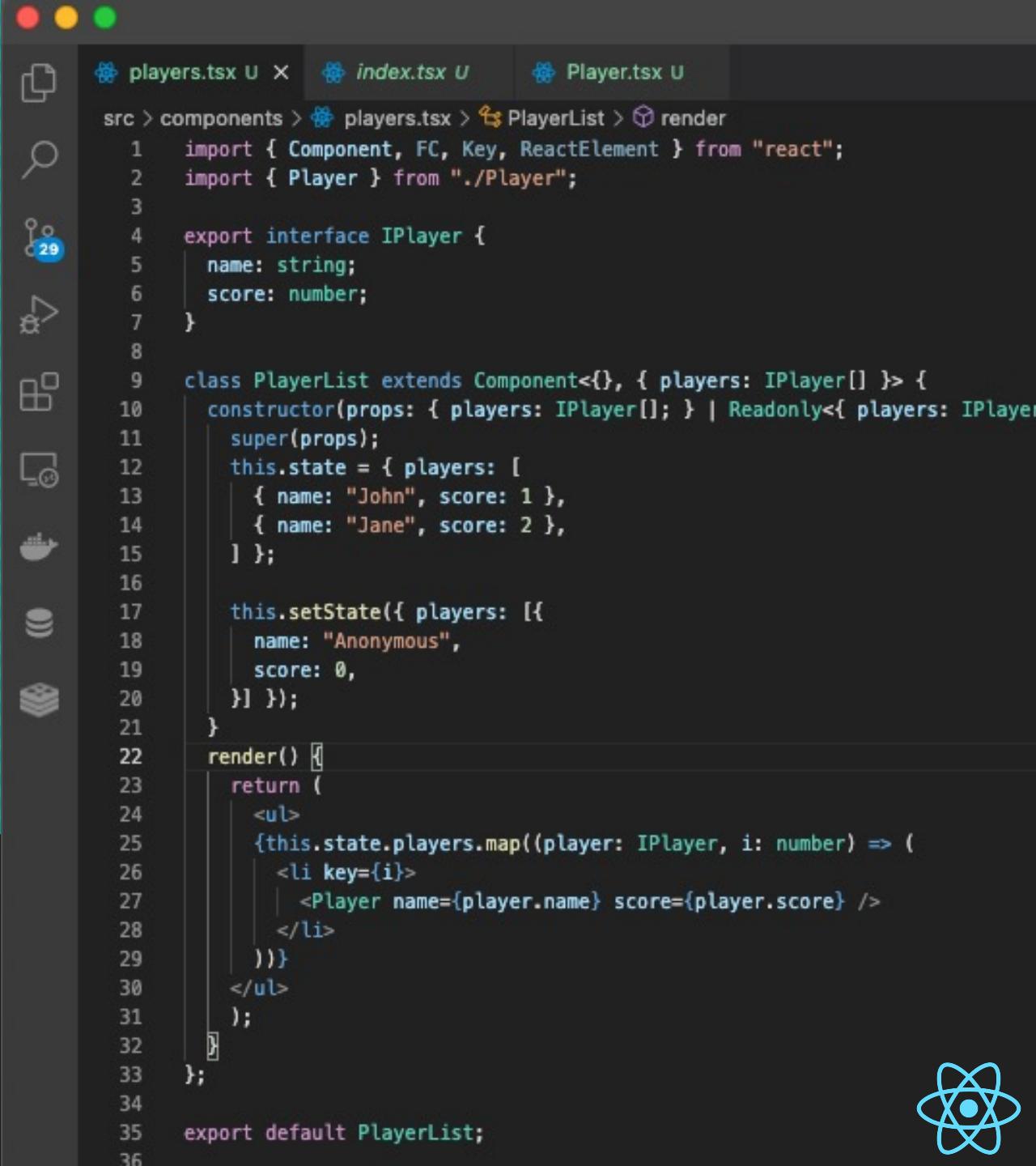
Dynamic information is information that can change.

```
src > components > players.tsx > PlayerList > constructor > players
  1 import { Component, FC, Key, ReactElement } from "react";
  2 import { Player } from "./Player";
  3
  4 export interface IPlayer {
  5   name: string;
  6   score: number;
  7 }
  8
  9 class PlayerList extends Component<{}, { players: IPlayer[] }> {
10   constructor(props: { players: IPlayer[]; } | Readonly<{ players: IPlayer[]; }>) {
11     super(props);
12     this.state = { players: [
13       { name: "John", score: 1 },
14       { name: "Jane", score: 2 },
15     ] };
16   }
17   render() {
18     return (
19       <ul>
20         {this.state.players.map((player: IPlayer, i: number) => (
21           <li key={i}>
22             <Player name={player.name} score={player.score} />
23           </li>
24         )));
25       </ul>
26     );
27   }
28 }
29
30 export default PlayerList;
```



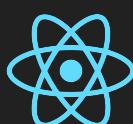
Update state with this.setState

A component changes its state by calling the function `this.setState()`.



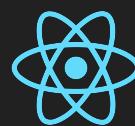
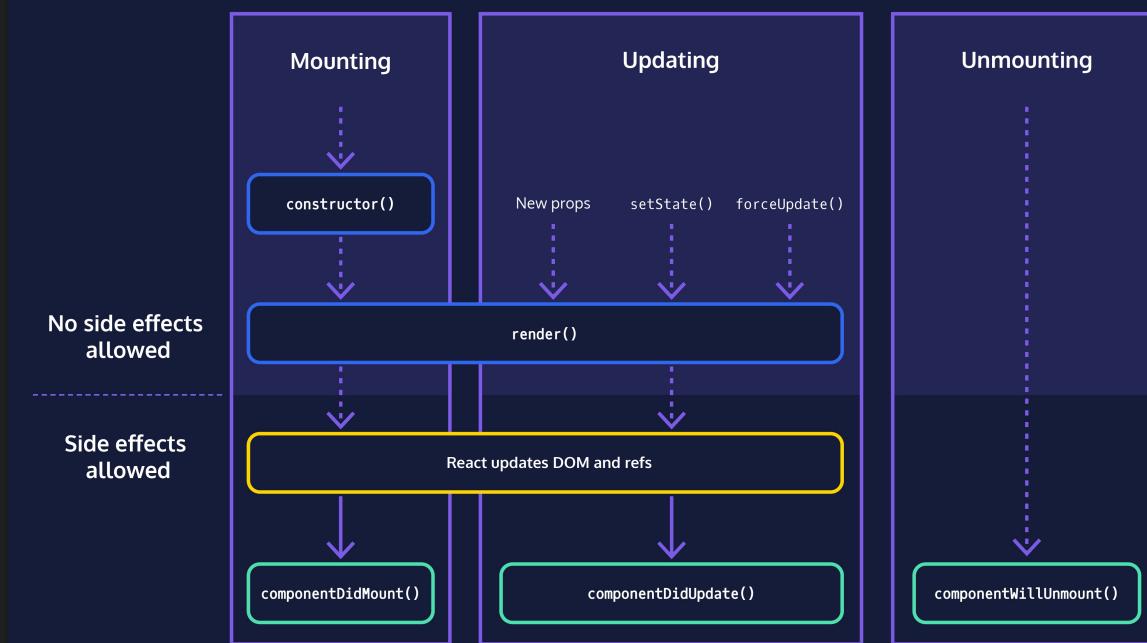
The screenshot shows a code editor window with three tabs at the top: "players.tsx U X", "index.tsx U", and "Player.tsx U". The "Player.tsx U" tab is active, showing the following code:

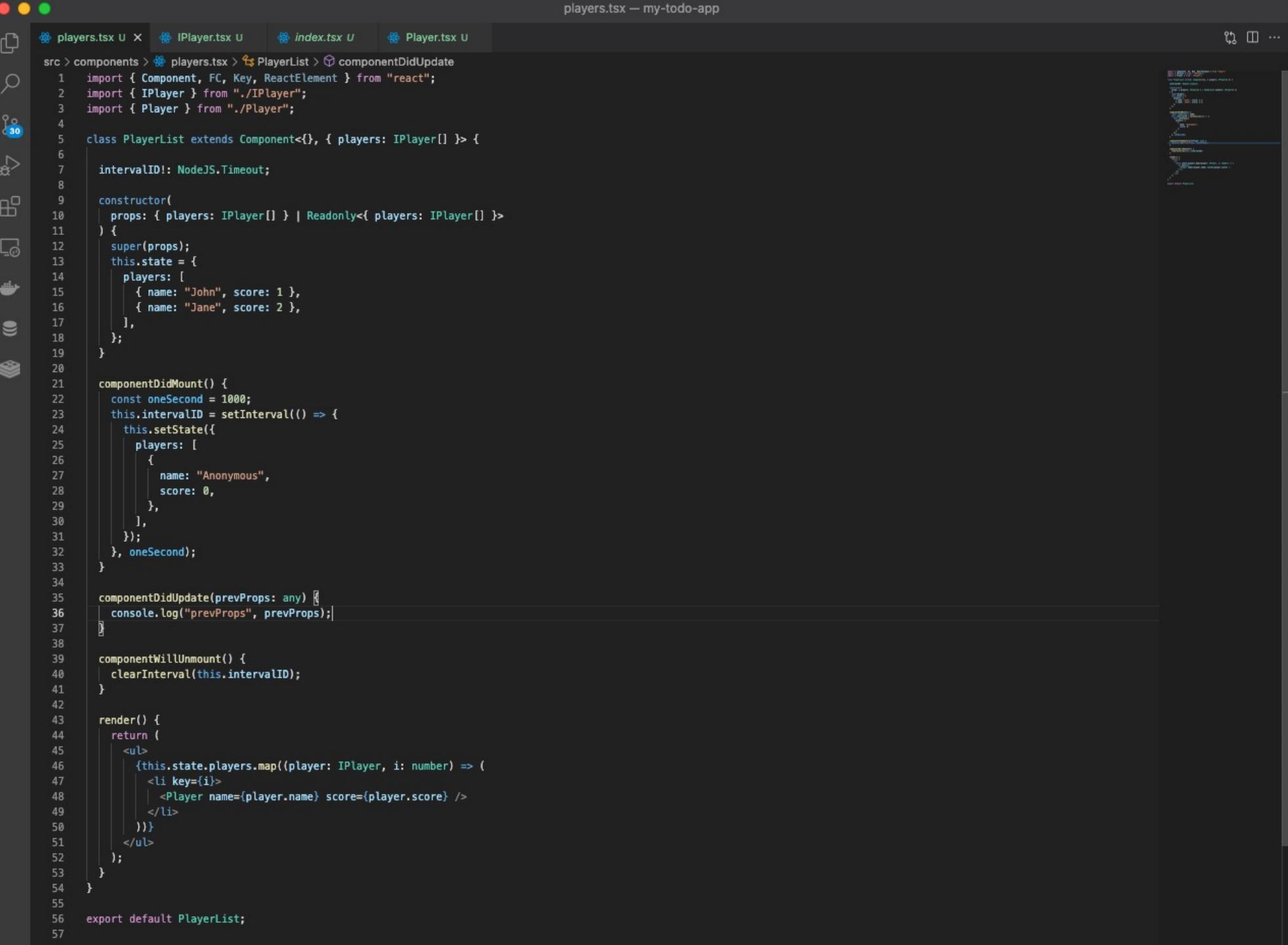
```
src > components > players.tsx > PlayerList > render
1 import { Component, FC, Key, ReactElement } from "react";
2 import { Player } from "./Player";
3
4 export interface IPlayer {
5   name: string;
6   score: number;
7 }
8
9 class PlayerList extends Component<{}, { players: IPlayer[] }> {
10   constructor(props: { players: IPlayer[]; } | Readonly<{ players: IPlayer[] }>) {
11     super(props);
12     this.state = { players: [
13       { name: "John", score: 1 },
14       { name: "Jane", score: 2 },
15     ] };
16
17     this.setState({ players: [
18       { name: "Anonymous",
19         score: 0,
20       } ]);
21   }
22   render() {
23     return (
24       <ul>
25         {this.state.players.map((player: IPlayer, i: number) => (
26           <li key={i}>
27             <Player name={player.name} score={player.score} />
28           </li>
29         )));
30       </ul>
31     );
32   }
33 }
34
35 export default PlayerList;
```



```
console.info(`Review`)  
// Review
```

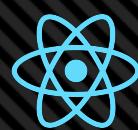
The Component Lifecycle





players.tsx U × IPlayer.tsx U index.tsx U Player.tsx U

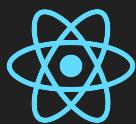
```
src > components > players.tsx > PlayerList > componentDidUpdate
1 import { Component, FC, Key, ReactElement } from "react";
2 import { IPlayer } from "./IPlayer";
3 import { Player } from "./Player";
4
5 class PlayerList extends Component<{}, { players: IPlayer[] }> {
6
7   intervalID!: NodeJS.Timeout;
8
9   constructor(
10     props: { players: IPlayer[] } | Readonly<{ players: IPlayer[] }>
11   ) {
12     super(props);
13     this.state = {
14       players: [
15         { name: "John", score: 1 },
16         { name: "Jane", score: 2 },
17       ],
18     };
19   }
20
21   componentDidMount() {
22     const oneSecond = 1000;
23     this.intervalID = setInterval(() => {
24       this.setState({
25         players: [
26           {
27             name: "Anonymous",
28             score: 0,
29           },
30         ],
31       });
32     }, oneSecond);
33   }
34
35   componentDidUpdate(prevProps: any) {
36     console.log("prevProps", prevProps);
37   }
38
39   componentWillUnmount() {
40     clearInterval(this.intervalID);
41   }
42
43   render() {
44     return (
45       <ul>
46         {this.state.players.map((player: IPlayer, i: number) => (
47           <li key={i}>
48             <Player name={player.name} score={player.score} />
49           </li>
50         )));
51       </ul>
52     );
53   }
54 }
55
56 export default PlayerList;
```



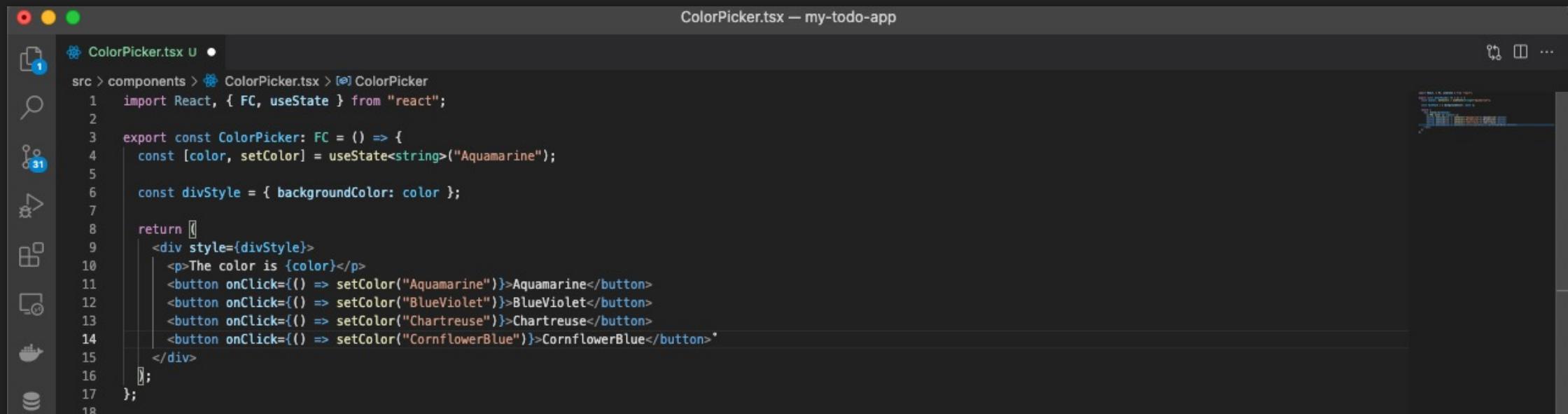
```
console.info(`Review`)  
// Review
```

Hooks

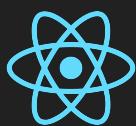
- useState
- useEffect
- useContext
- useReducer
- useRef



React.useState



```
ColorPicker.tsx — my-todo-app
src > components > ColorPicker.tsx > ColorPicker
1 import React, { FC, useState } from "react";
2
3 export const ColorPicker: FC = () => {
4   const [color, setColor] = useState<string>("Aquamarine");
5
6   const divStyle = { backgroundColor: color };
7
8   return [
9     <div style={divStyle}>
10       <p>The color is {color}</p>
11       <button onClick={() => setColor("Aquamarine")}>Aquamarine</button>
12       <button onClick={() => setColor("BlueViolet")}>BlueViolet</button>
13       <button onClick={() => setColor("Chartreuse")}>Chartreuse</button>
14       <button onClick={() => setColor("CornflowerBlue")}>CornflowerBlue</button>
15     </div>
16   ];
17 };
18
```



React.useEffect

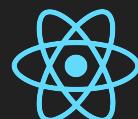
The image shows two side-by-side code editors, both titled "pageTitle.tsx — my-todo-app".

Left Editor (Original Class-Based Component):

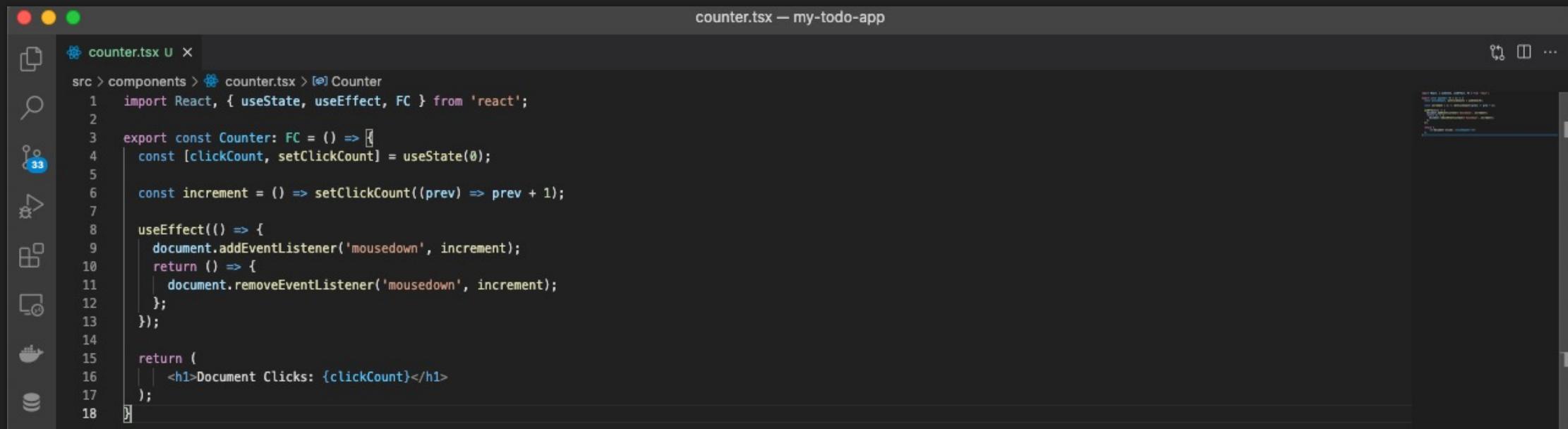
```
src > components > pageTitle.tsx > pageTitle > constructor
1 import React, { Component } from "react";
2
3 export default class PageTitle extends Component<{}, { name: string }> {
4   constructor(props: any) {
5     super(props);
6     this.state = [
7       name: "",
8     ];
9   }
10
11   componentDidMount() {
12     document.title = this.state.name;
13   }
14
15   componentDidUpdate() {
16     if (document.title == `Hi, ${this.state.name}`);
17   }
18
19   render() {
20     return (
21       <div>
22         <p>Use the input field below to rename this page!</p>
23         <input
24           onChange={({ target }) => this.setState({ name: target.value })}
25           value={this.state.name}
26           type="text"
27         />
28       </div>
29     );
30   }
31 }
32 }
```

Right Editor (Updated Functional Component Using useEffect):

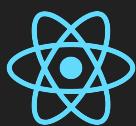
```
src > components > pageTitle.tsx > pageTitle
1 import React, { useState, useEffect, FC } from 'react';
2
3 export const PageTitle: FC = () => {
4   const [name, setName] = useState('');
5
6   useEffect(() => {
7     document.title = `Hi, ${name}`;
8   });
9
10  return [
11    <div>
12      <p>Use the input field below to rename this page!</p>
13      <input onChange={({ target }) => setName(target.value)} value={name} type='text' />
14    </div>
15  ];
16}
```



React.useEffect



```
counter.tsx — my-todo-app
src > components > counter.tsx > Counter
1 import React, { useState, useEffect, FC } from 'react';
2
3 export const Counter: FC = () => [
4   const [clickCount, setClickCount] = useState(0);
5
6   const increment = () => setClickCount((prev) => prev + 1);
7
8   useEffect(() => {
9     document.addEventListener('mousedown', increment);
10    return () => {
11      document.removeEventListener('mousedown', increment);
12    };
13  });
14
15  return (
16    <h1>Document Clicks: {clickCount}</h1>
17  );
18]
```



```
console.info(`Review`)  
// Review
```

References

- <https://reactjs.org>
- <https://www.codecademy.com/courses/react-101/lessons/your-first-react-component/exercises/import-react>

Thanks for listening...