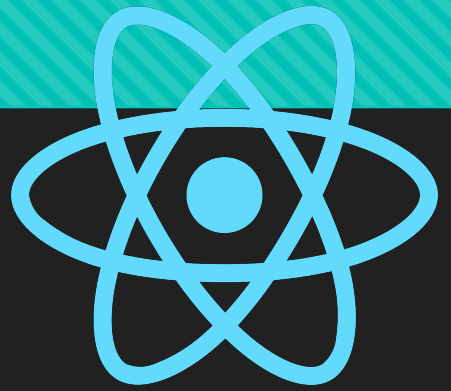
Modern Web Development Using React

Olgun DUTKAN – UI / UX Team in HAVELSAN

odutkan@havelsan.com.tr



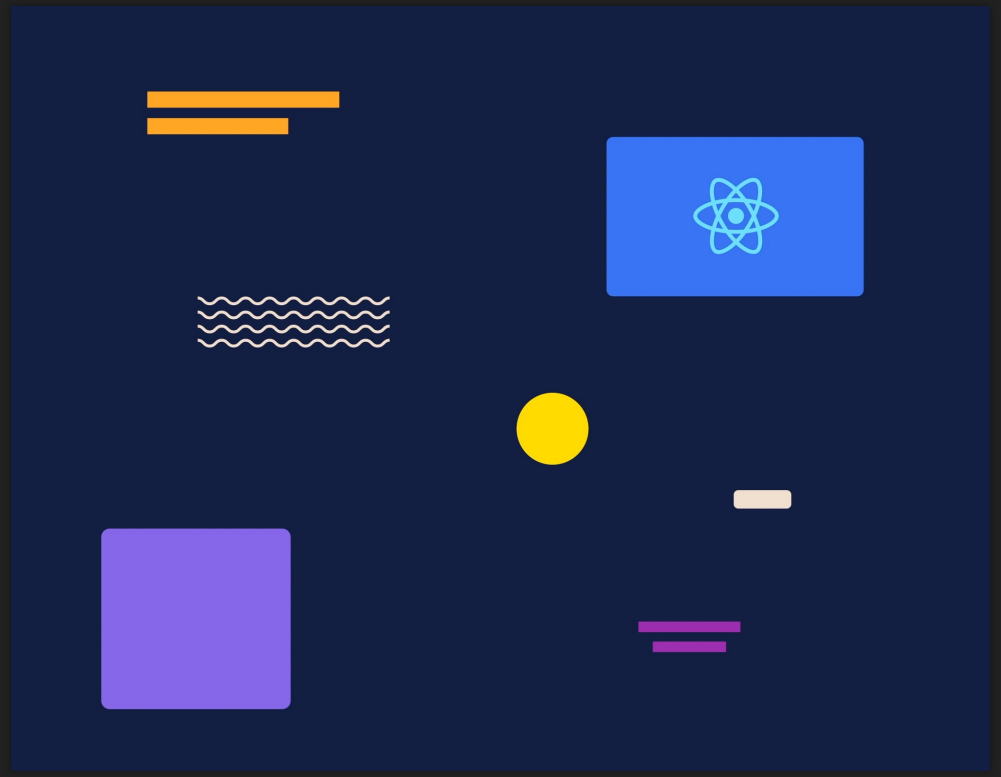
```
class HelloMessage extends React.Component {
    render() {
        return (
            <div> Hello, I am {this.props.name} </div>
        );
    }
}

ReactDOM.render(
    <HelloMessage name="Olgun DUTKAN" />,
    document.getElementById('root')
);
```

Agenda

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

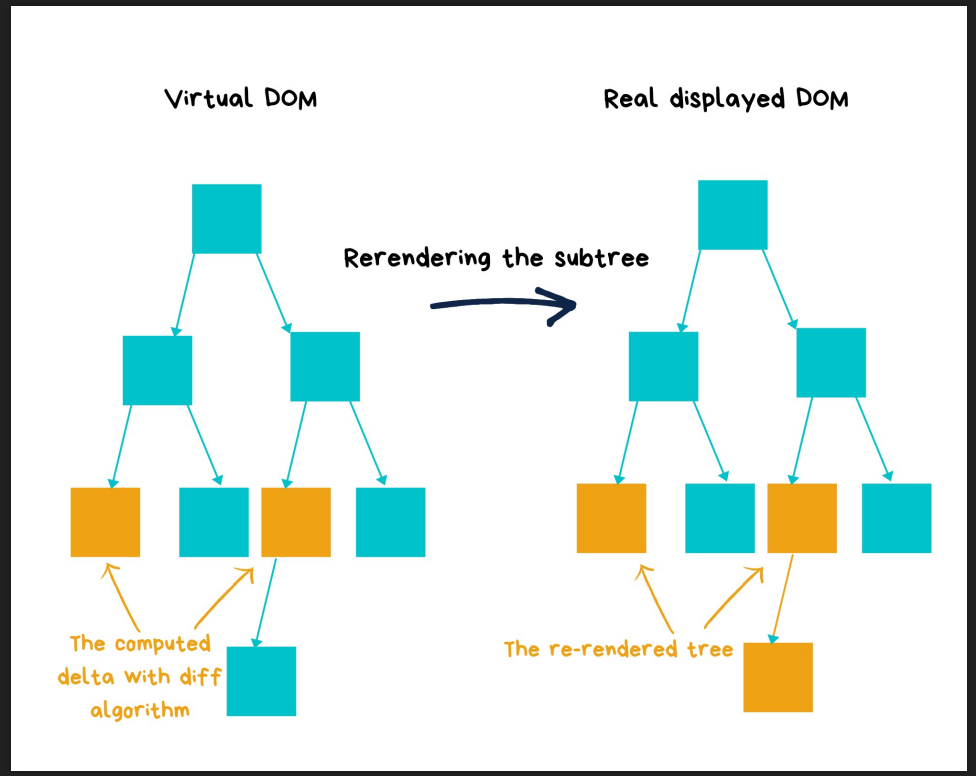
What is React?

- 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!

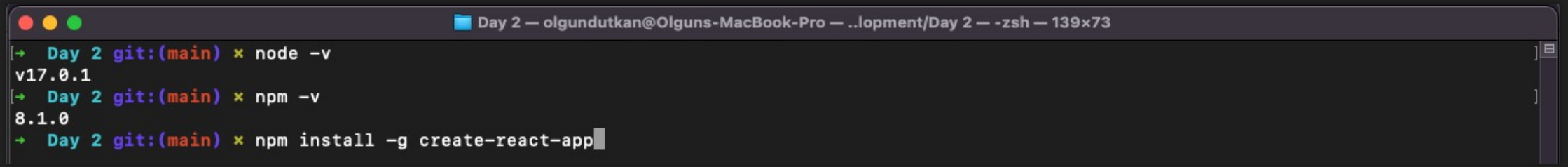


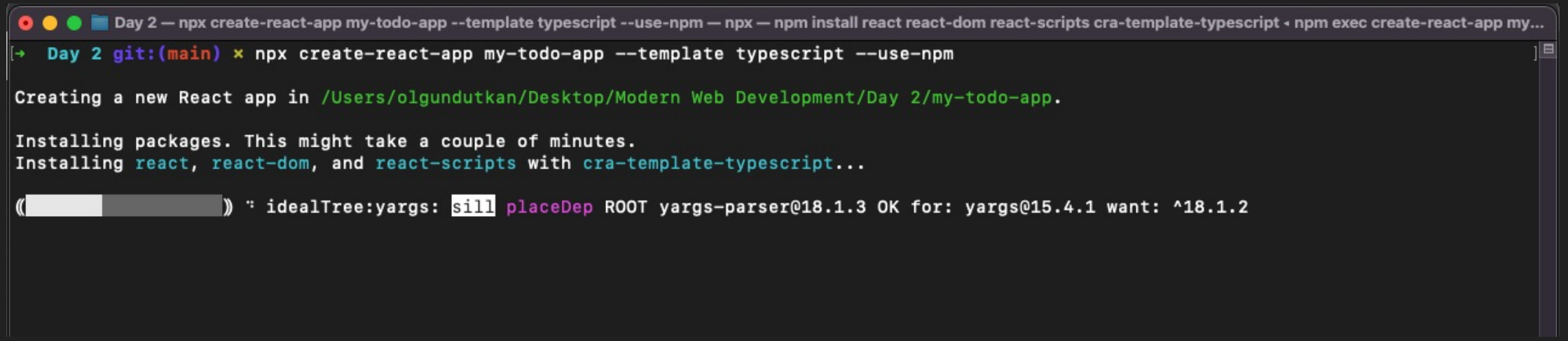
Why React?

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

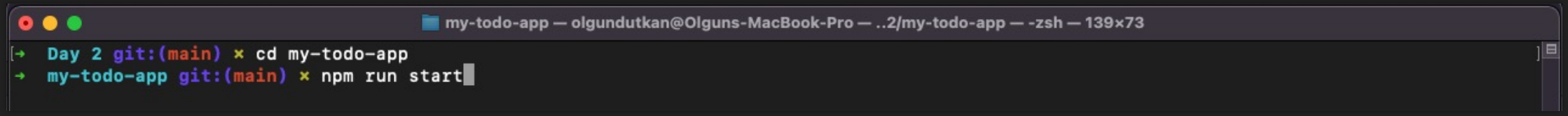
A brief history of React

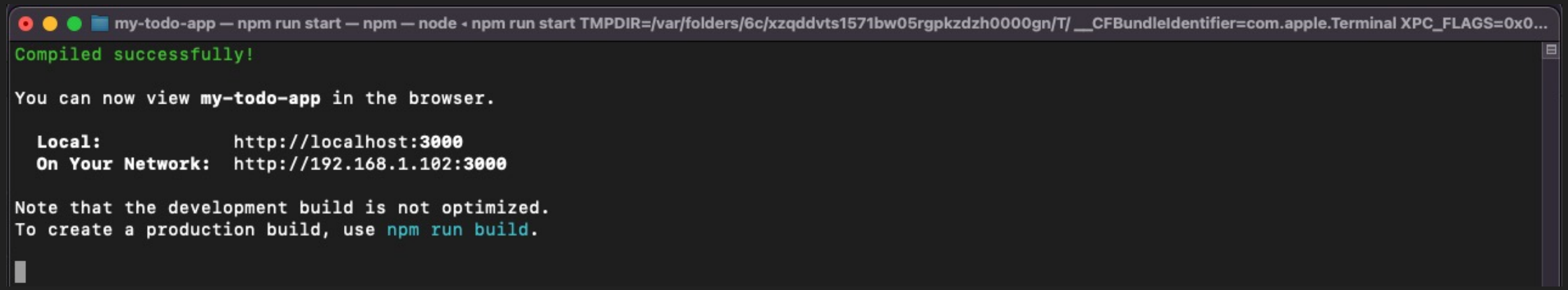
Create React App

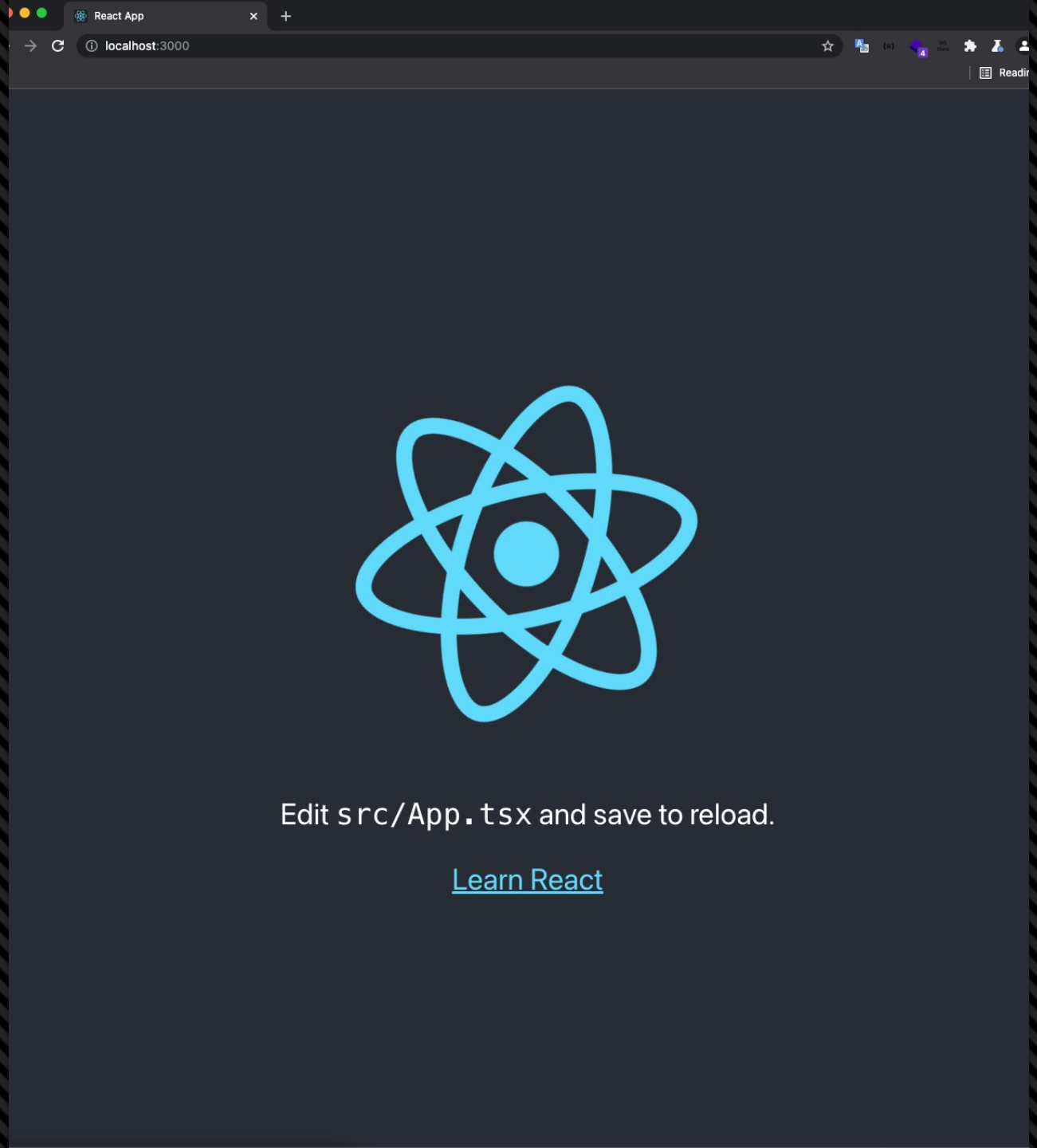


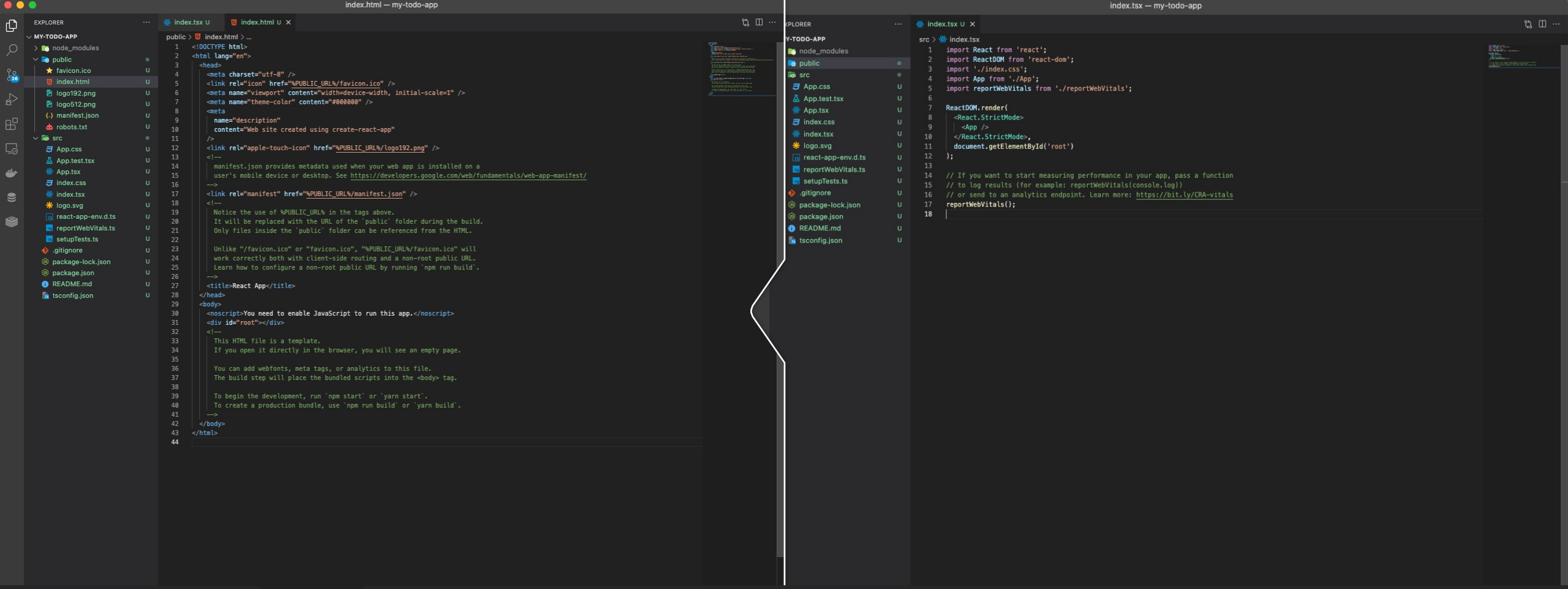


Run the React Application






React App
localhost:3000
Edit src/App.tsx and save to reload.
Learn React



React Render HTML

App.tsx — my-todo-app

EXPLORER

MY-TODO-APP

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

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

src > App.tsx > ...

```tsx
import React from 'react';
import logo from './logo.svg';
import './App.css';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.tsx</code> and save to reload.
        </p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn React
        </a>
      </header>
    </div>
  );
}

export default App;

```

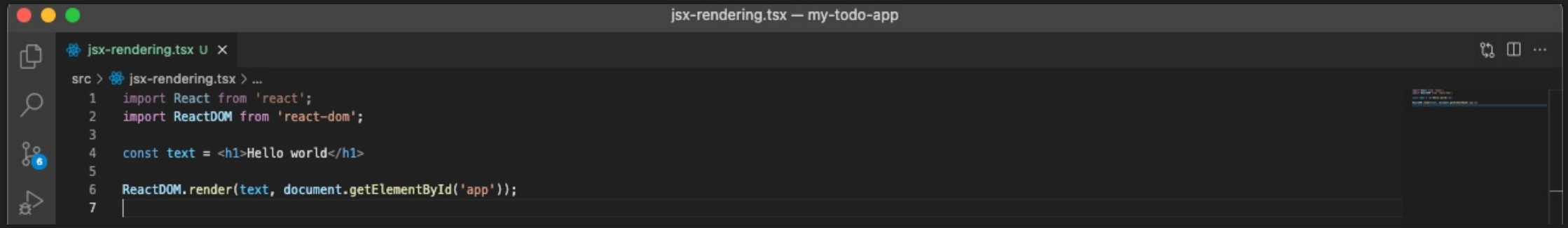
```
console.info(`Review`)
// Review
```

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>;
```

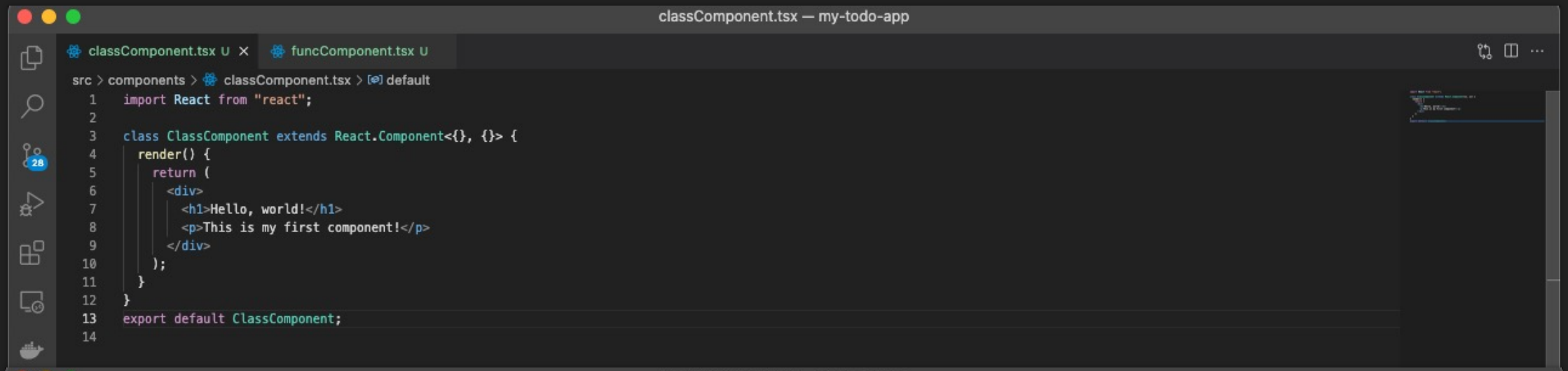

JSX Rendering

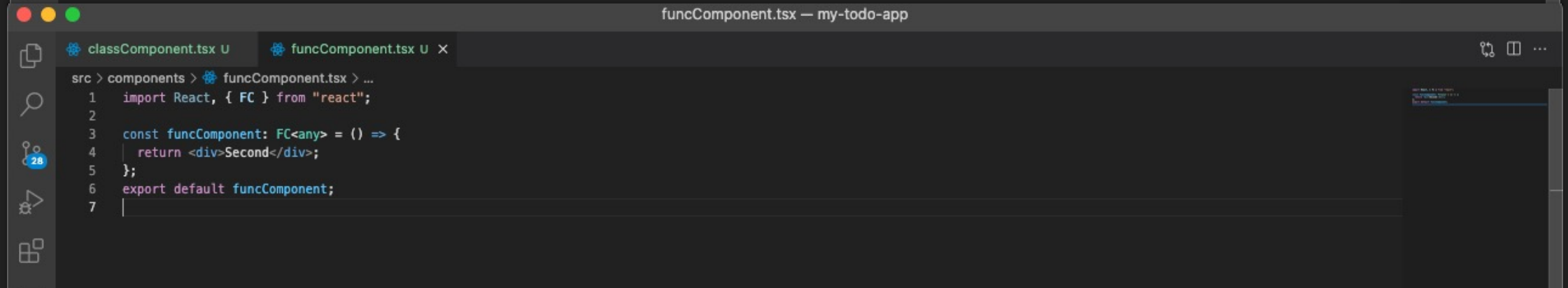


React Components

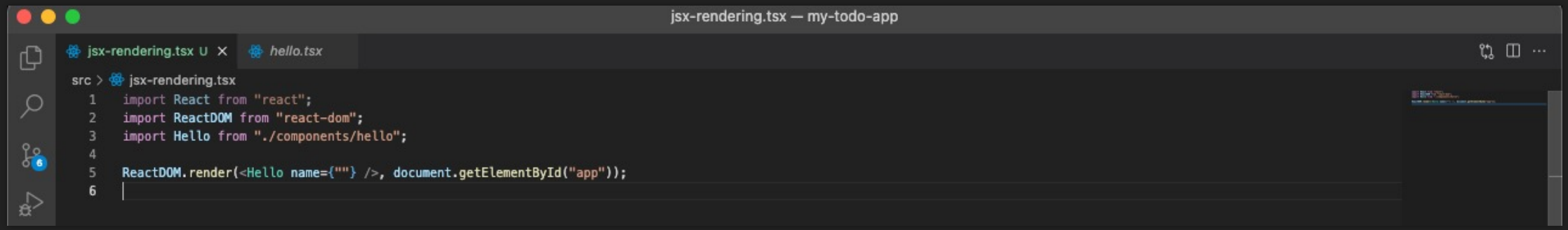
- Components are independent and reusable bits of code. They serve the same purpose as JavaScript functions, but work in isolation and return HTML.
- Components come in two types, Class components and Function components, in this tutorial we will concentrate on Function components.

Create Your First Component





Component Rendering



React Props

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

hello.tsx U App.tsx U

src > components > hello.tsx > ...

```tsx
import React, { FC } from "react";

interface IHello {
  name: string;
}

const Hello: FC<IHello> = ({name}) =>
  return <div>Hello, I am {name}</div
};
export default Hello;
```

hello.tsx U App.tsx U

c > App.tsx > ...

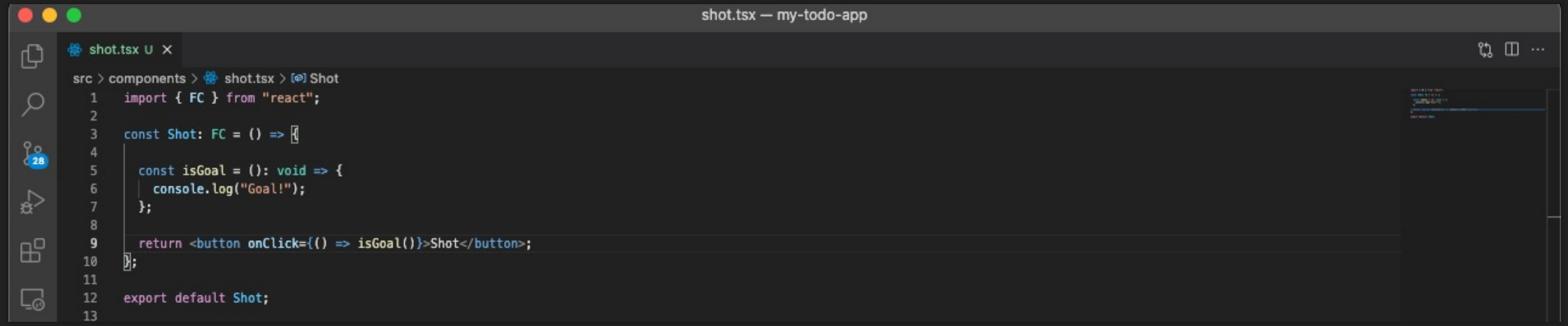
```tsx
import Hello from './components/hello';

function App() {
  return (
    <Hello name="Olgun DUTKAN"/>
  );
}

export default App;
```

React Props

React Events

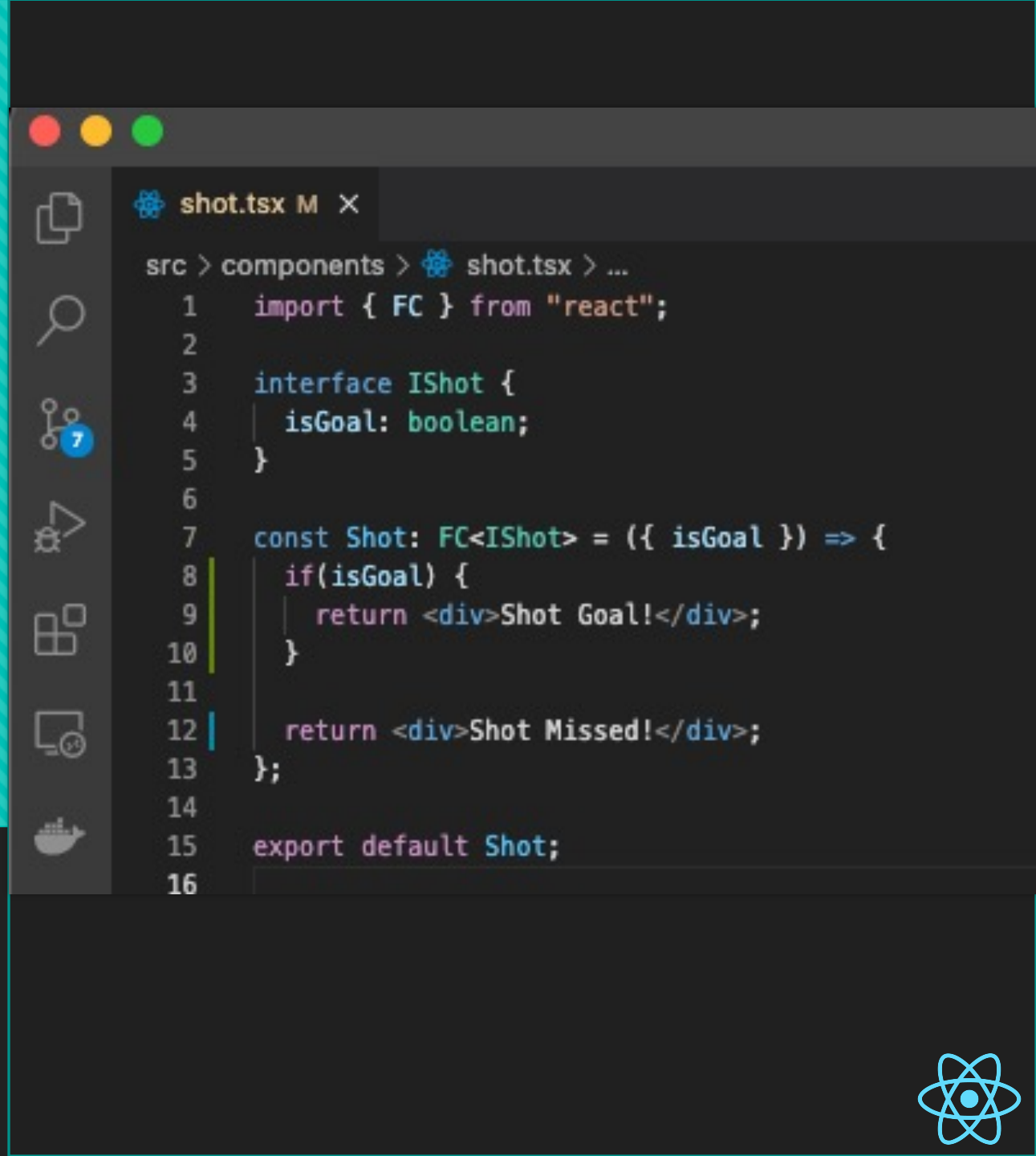
- Just like HTML DOM events, React can perform actions based on user events.
- React has the same events as HTML: click, change, mouseover etc.

React Events



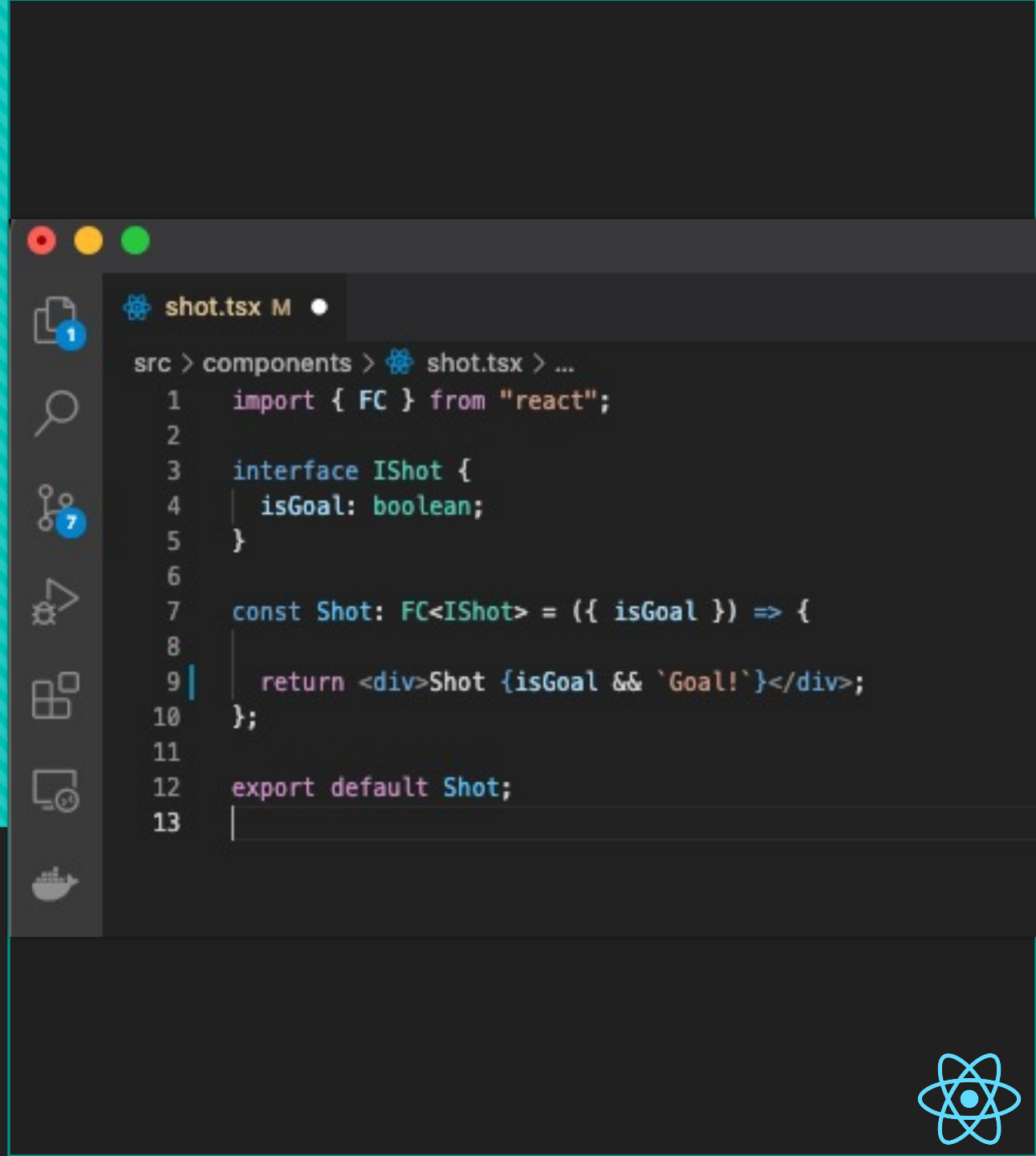
React Conditional Rendering

if Statement



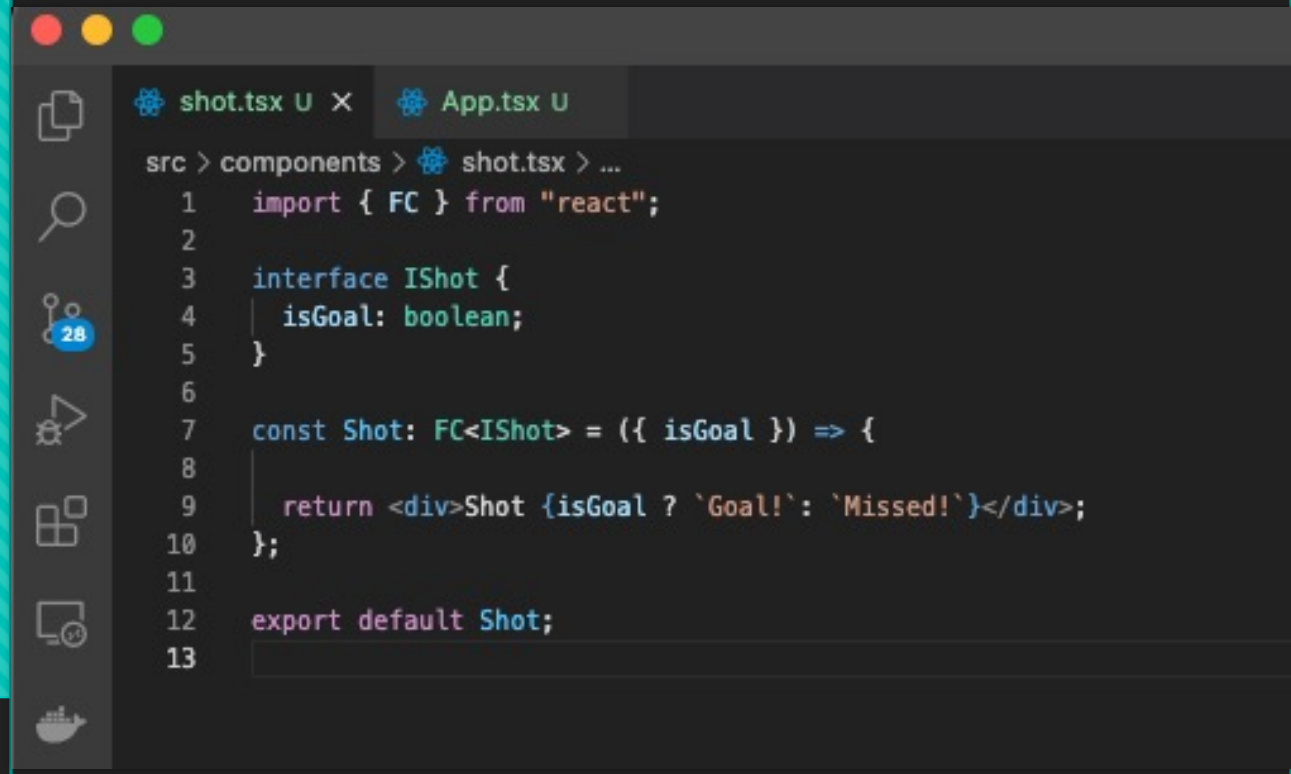
React Conditional Rendering

Logical && Operator



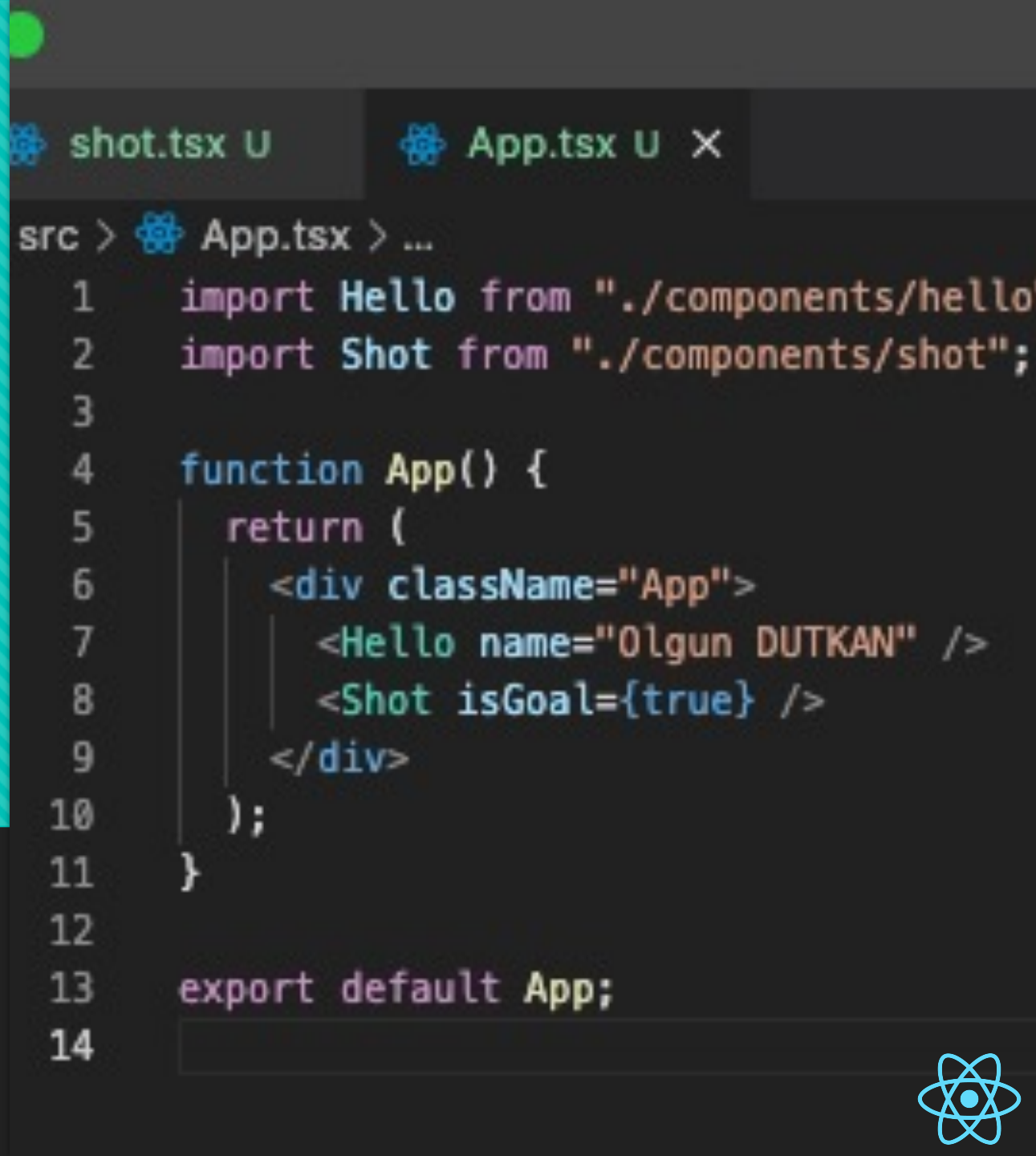
React Conditional Rendering

Logical && Operator

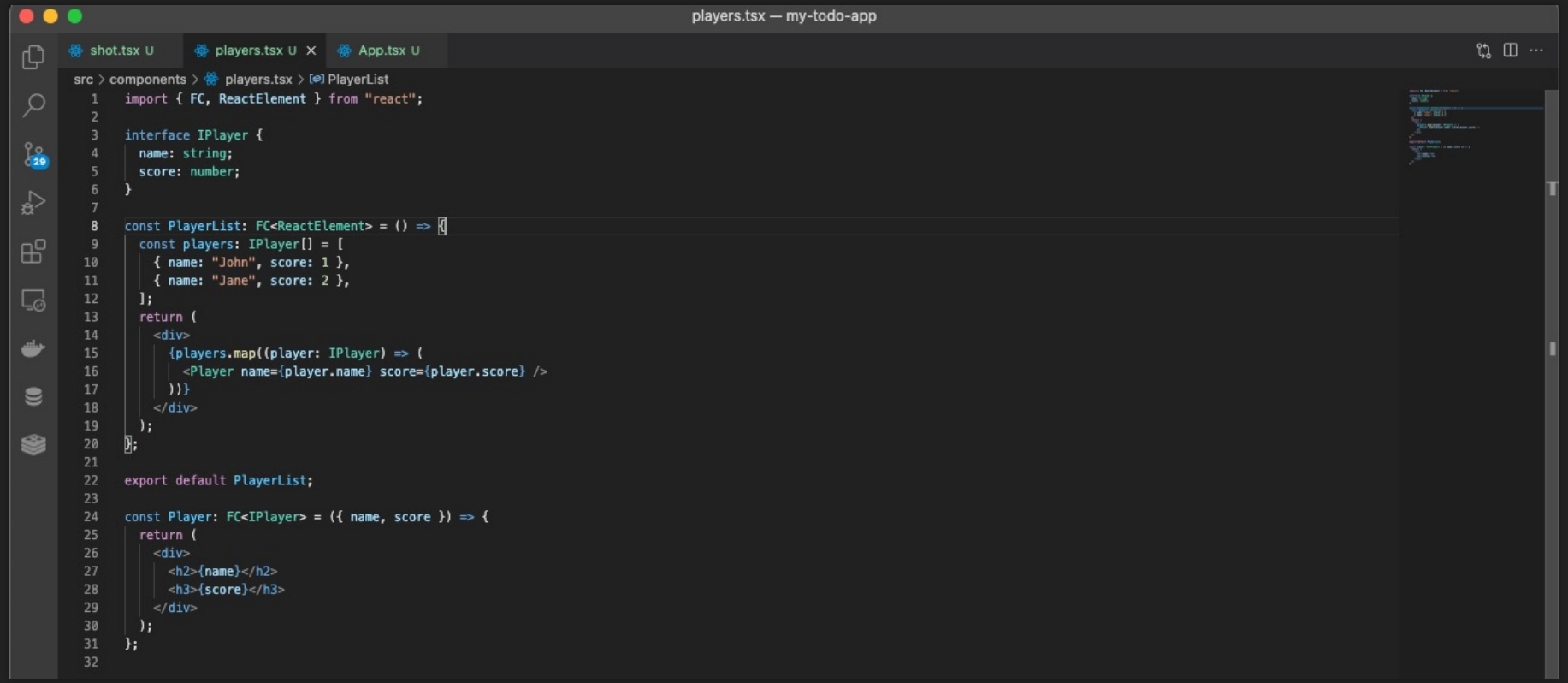


React Conditional Rendering

if Statement



React Lists


players.tsx — my-todo-app
shot.tsx U
players.tsx U
App.tsx U
src > components > players.tsx > PlayerList
import { FC, ReactElement } from "react";
interface IPlayer {
name: string;
score: number;
}
const PlayerList: FC<ReactElement> = () => {
const players: IPlayer[] = [
{ name: "John", score: 1 },
{ name: "Jane", score: 2 },
];
return (

{players.map((player: IPlayer) => (
<Player name={player.name} score={player.score} />
))}

);
};
export default PlayerList;
const Player: FC<IPlayer> = ({ name, score }) => {
return (

<h2>{name}</h2>
<h3>{score}</h3>

);
};

Keys

players.tsx — my-todo-app

shot.tsx U | players.tsx U | App.tsx U

src > components > players.tsx > PlayerList

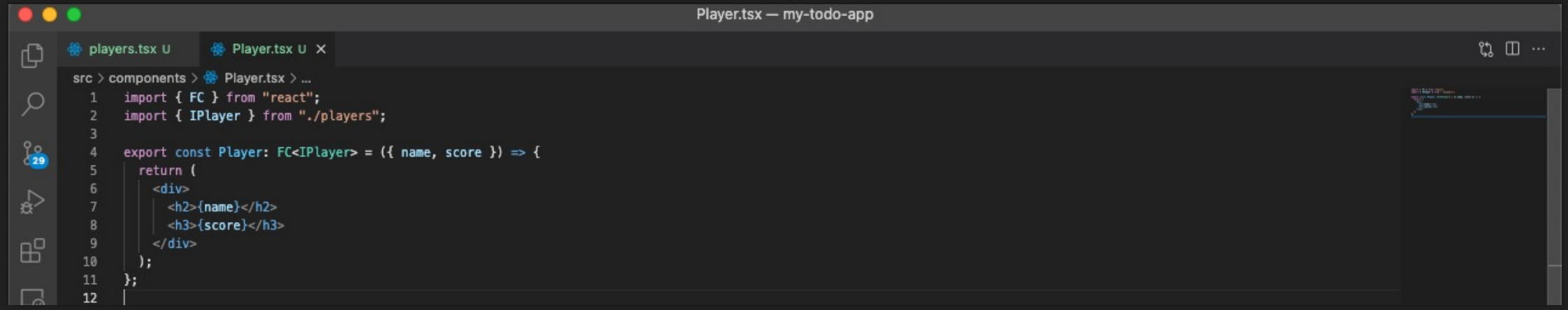
```tsx
import { FC, ReactElement } from "react";

interface IPlayer {
  name: string;
  score: number;
}

const PlayerList: FC<ReactElement> = () => {
  const players: IPlayer[] = [
    { name: "John", score: 1 },
    { name: "Jane", score: 2 },
  ];
  return (
    <ul>
      {players.map((player: IPlayer, i: number) => (
        <li key={i}>
          <Player name={player.name} score={player.score} />
        </li>
      ))}
    </ul>
  );
};

export default PlayerList;

const Player: FC<IPlayer> = ({ name, score }) => {
  return (
    <div>
      <h2>{name}</h2>
      <h3>{score}</h3>
    </div>
  );
};

```

```
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



Component Render Other Component

players.tsx — my-todo-app

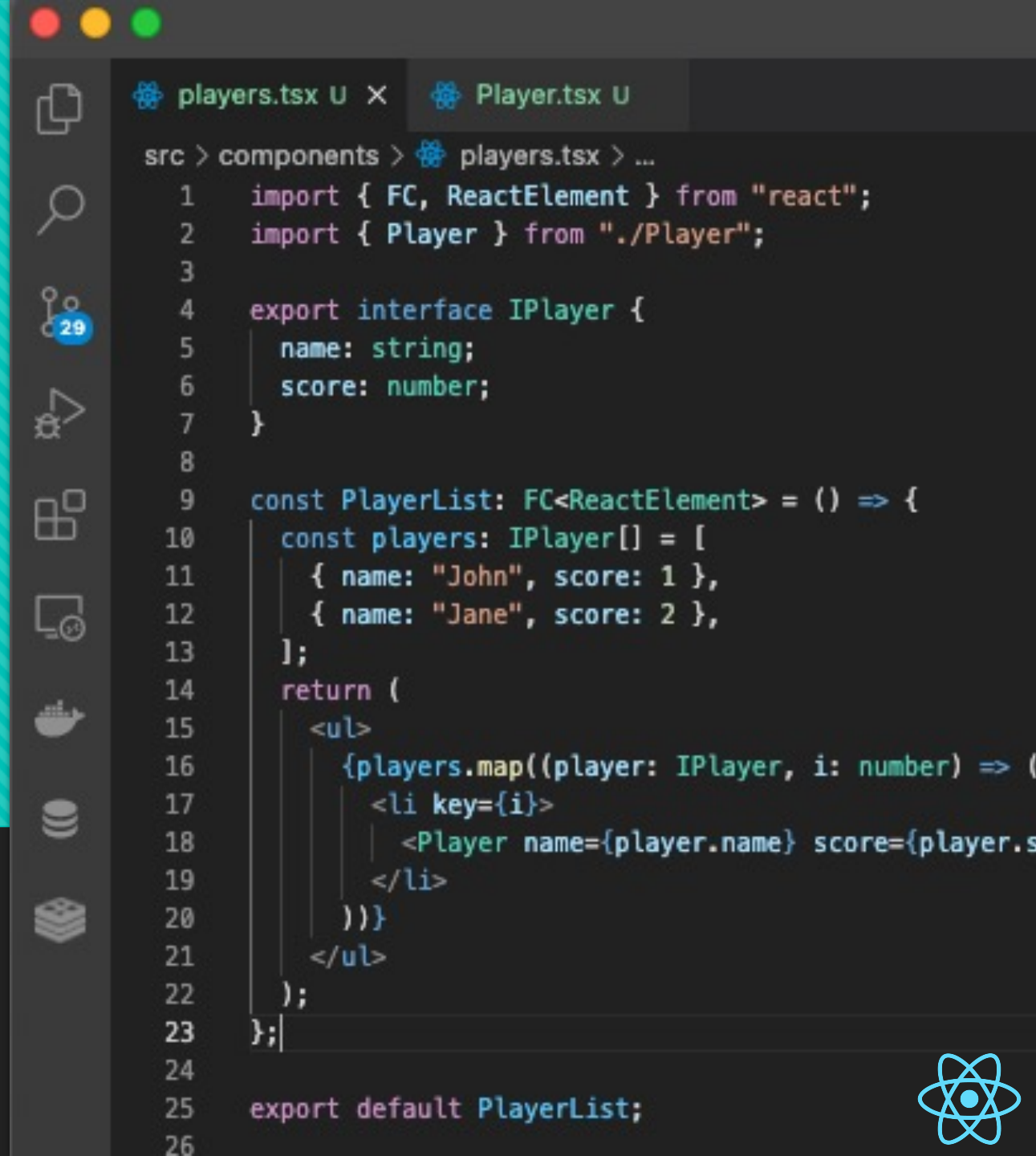
players.tsx U | Player.tsx U

src > components > players.tsx > ...

```tsx
import { FC, ReactElement } from "react";
import { Player } from "./Player";

export interface IPlayer {
  name: string;
  score: number;
}

const PlayerList: FC<ReactElement> = () => {
  const players: IPlayer[] = [
    { name: "John", score: 1 },
    { name: "Jane", score: 2 },
  ];
  return (
    <ul>
      {players.map((player: IPlayer, i: number) => (
        <li key={i}>
          <Player name={player.name} score={player.score} />
        </li>
      ))}
    </ul>
  );
};

export default PlayerList;
```

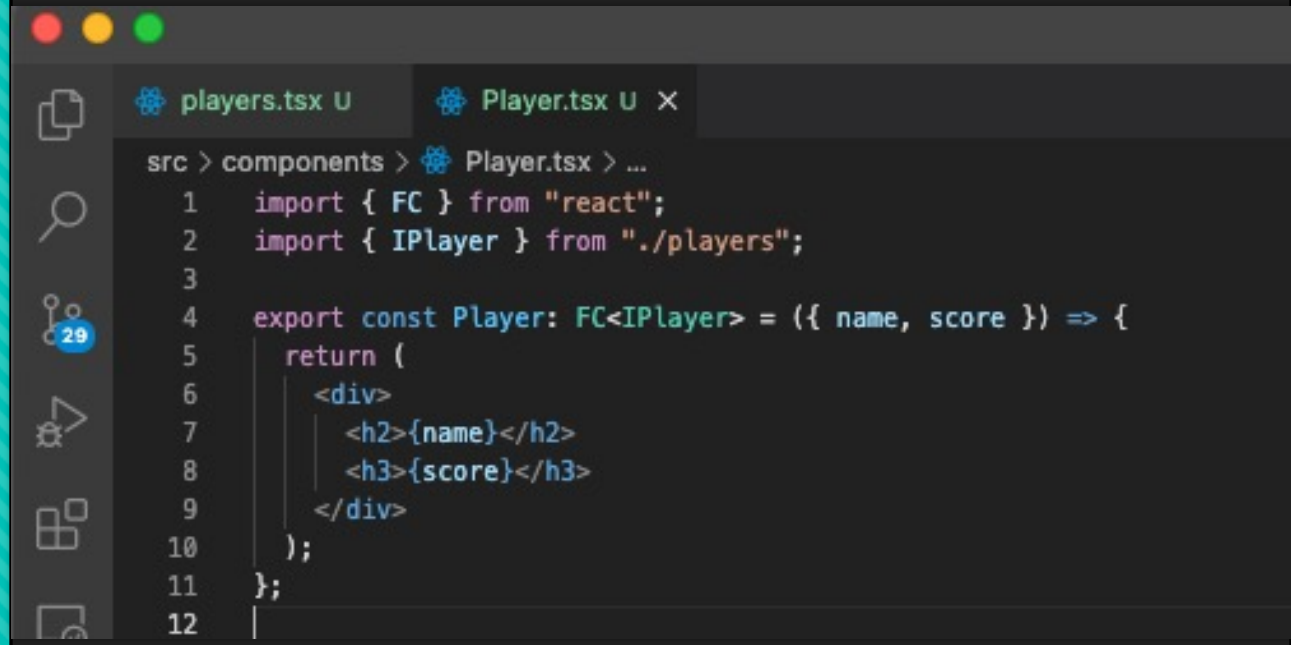

this.props

A component can *pass information* to another component.



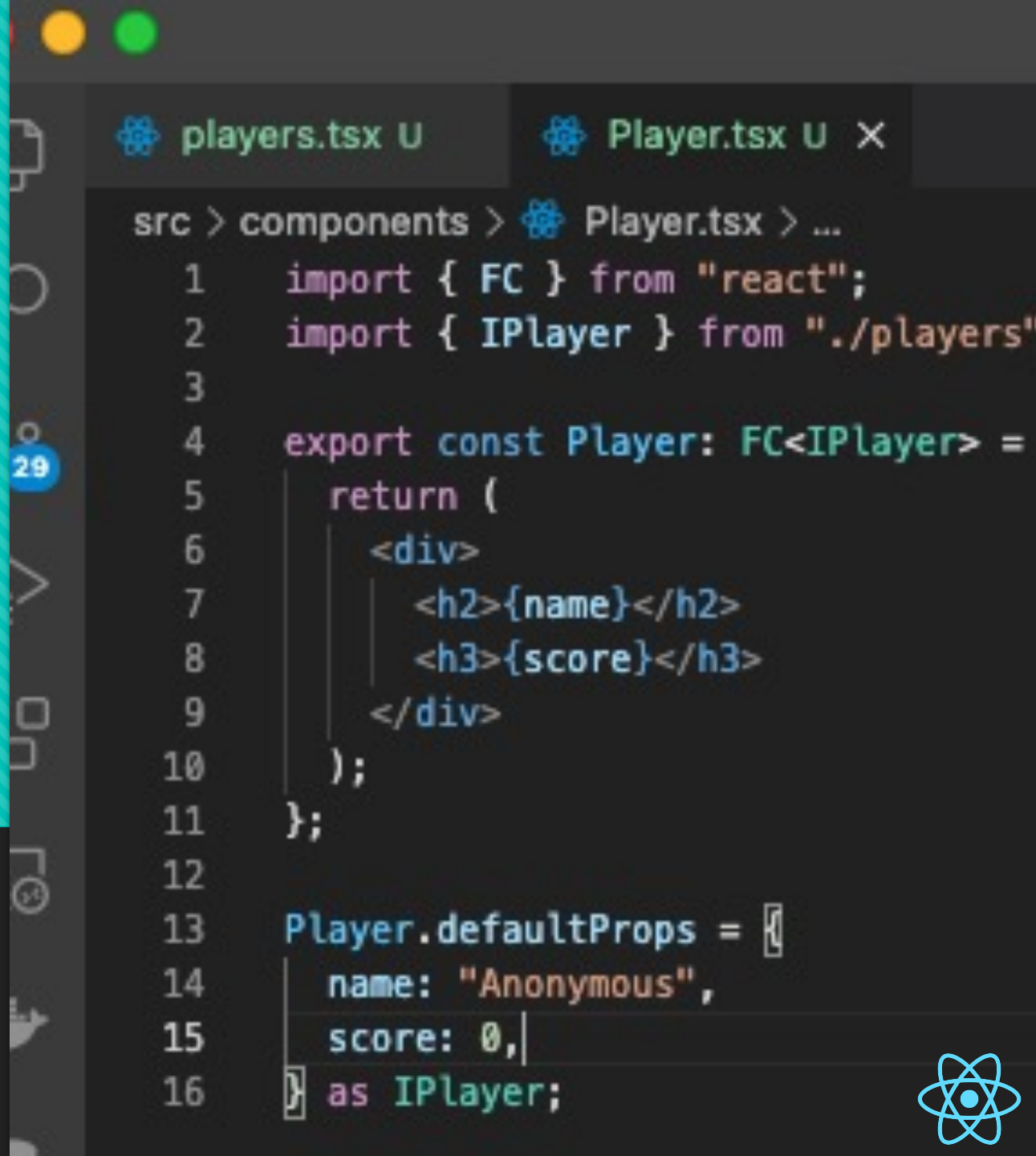
this.props

A component can *pass information* to another component.



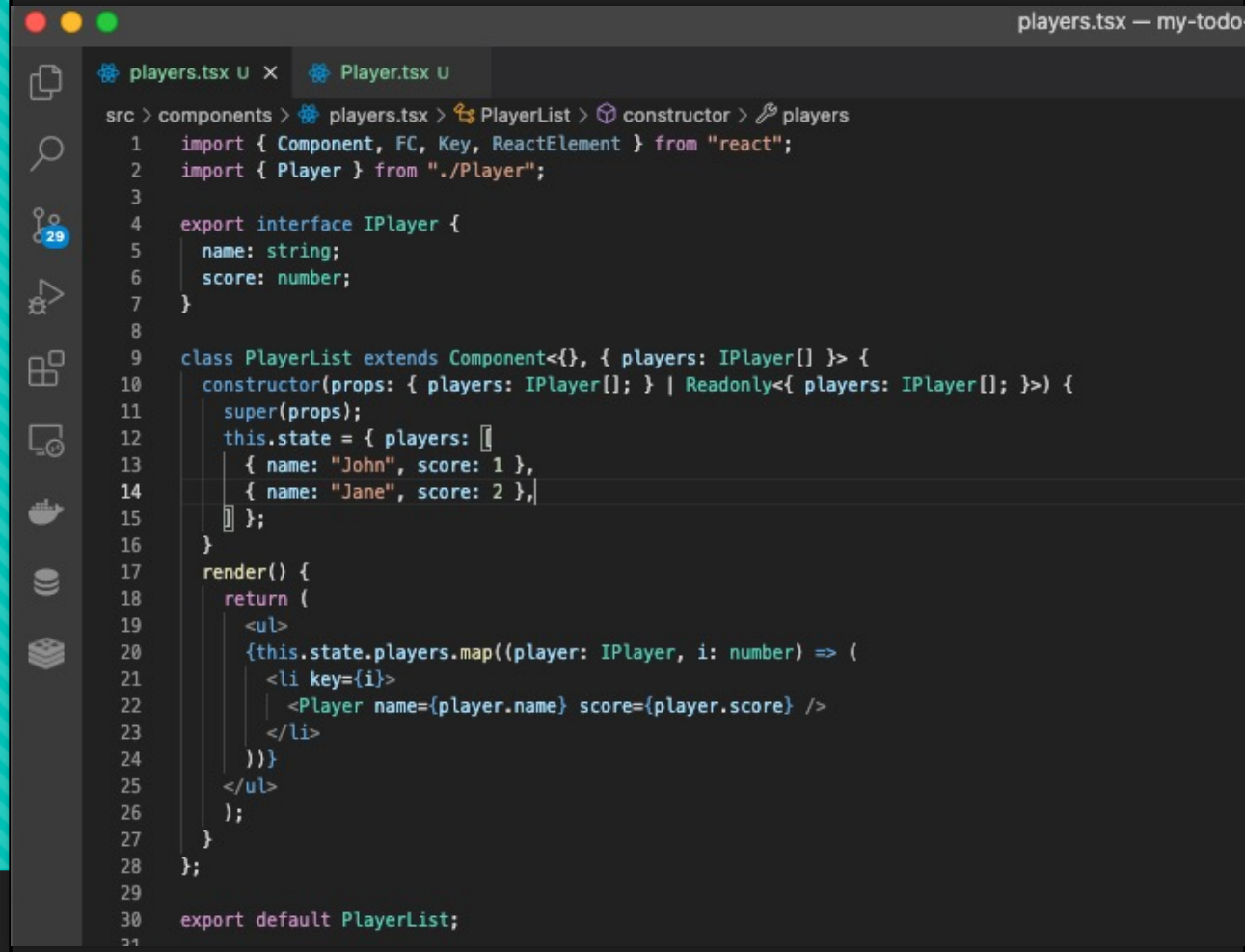
defaultProps

A component can *pass information* to another component.



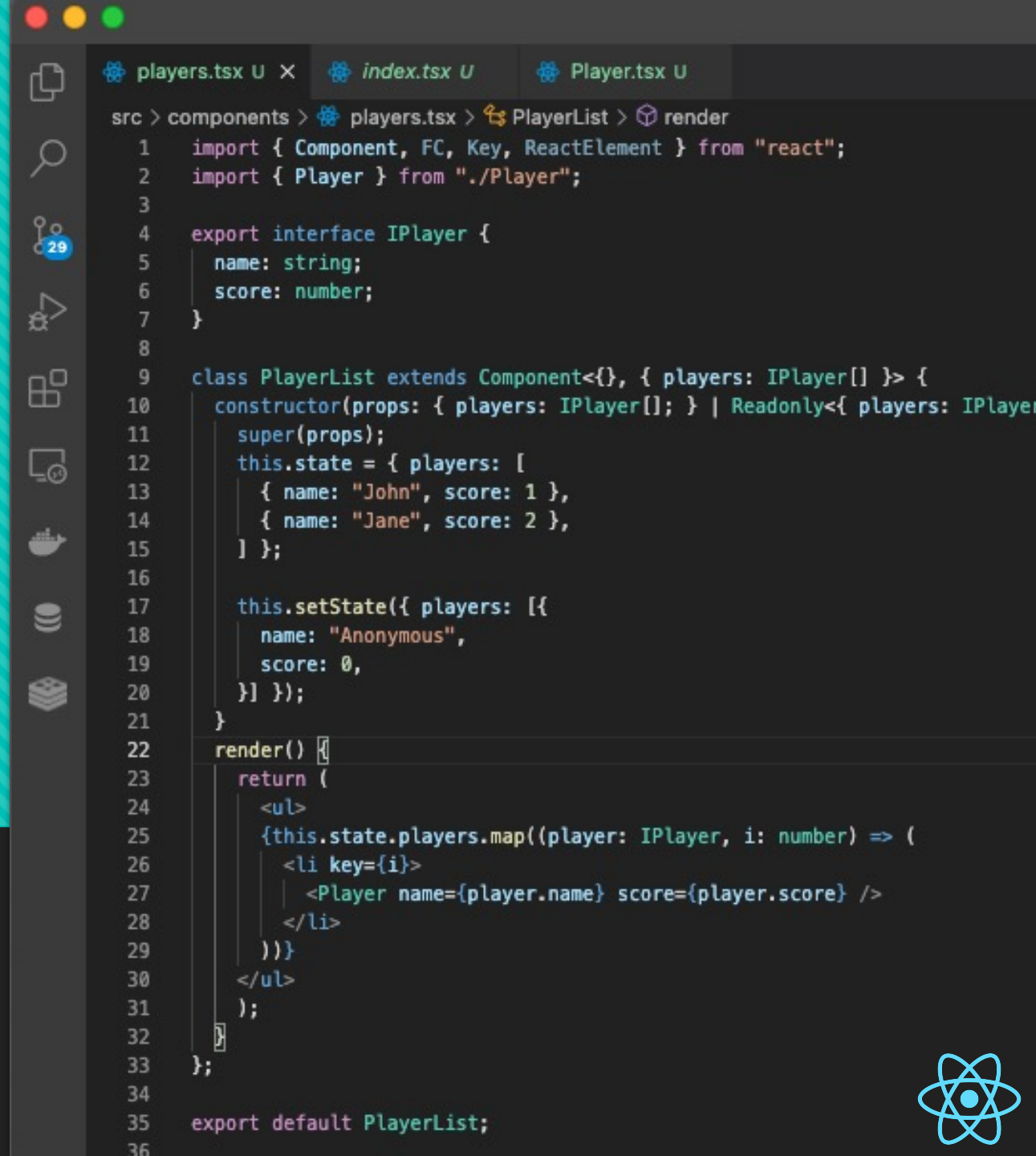
this.state

Dynamic information is information that can change.



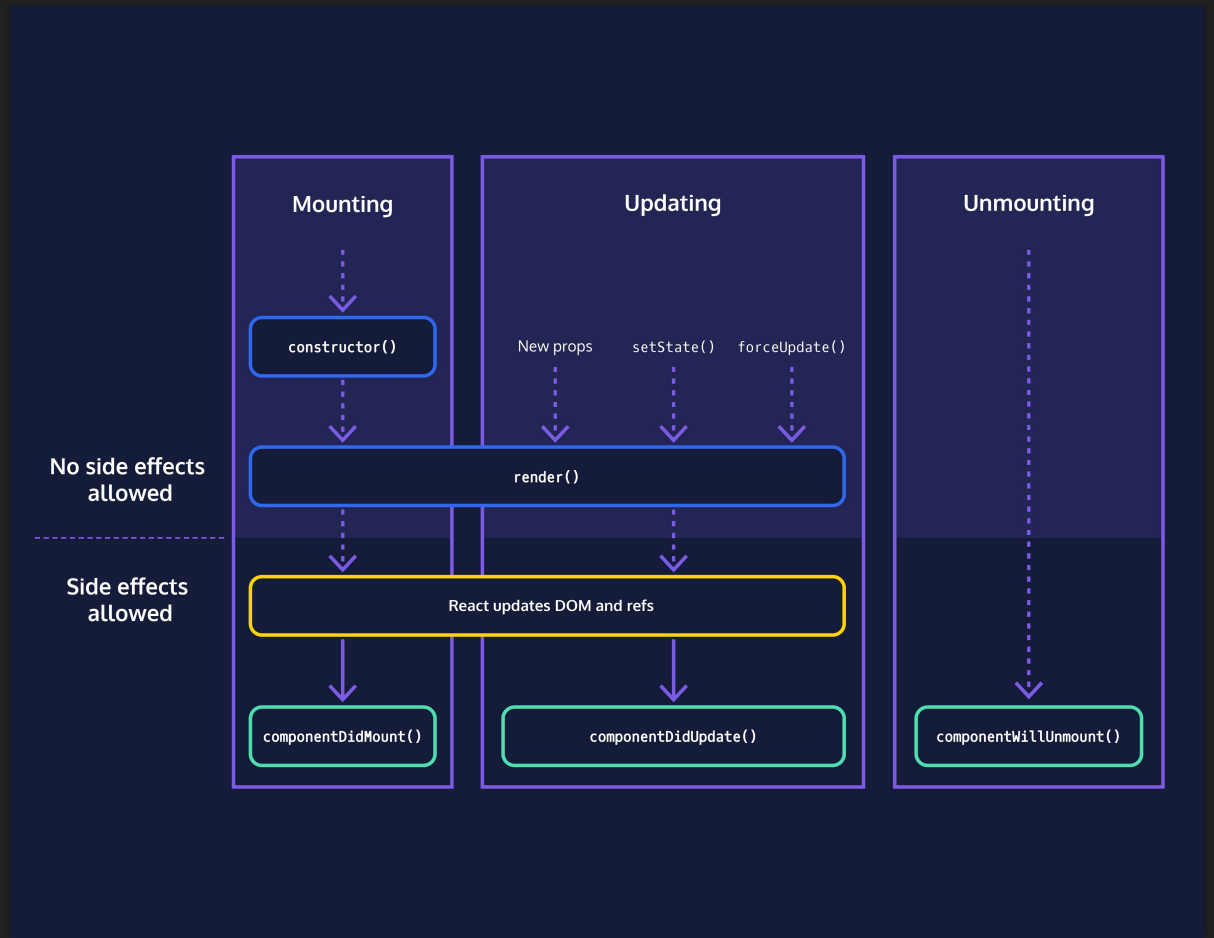
Update state with this.setState

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



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

The Component Lifecycle



players.tsx — my-todo-app

players.tsx U | IPlayer.tsx U | index.tsx U | Player.tsx U

src > components > players.tsx > PlayerList > componentDidUpdate

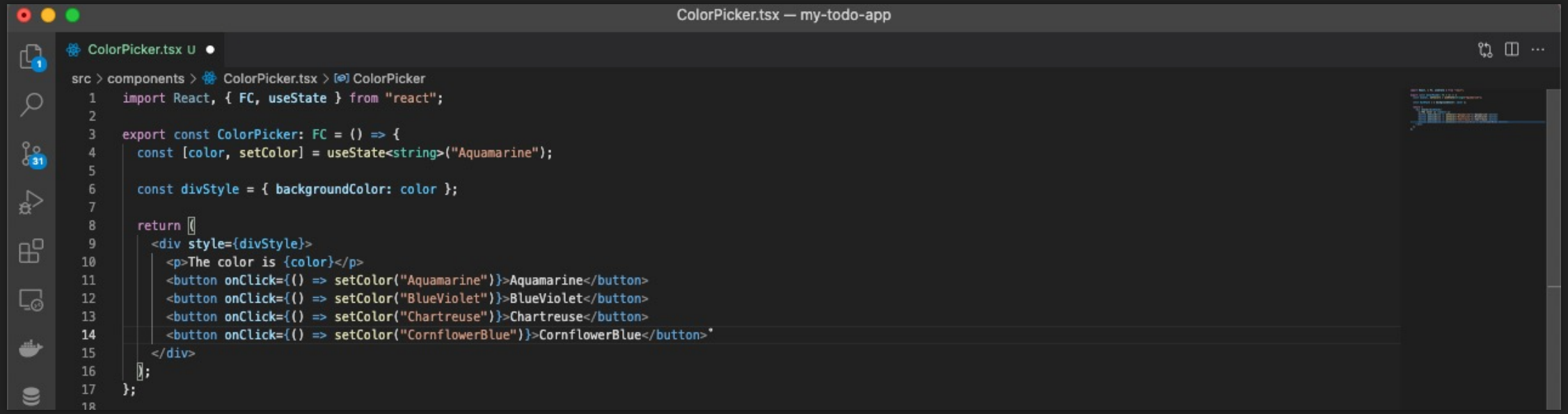
```tsx
import { Component, FC, Key, ReactElement } from "react";
import { IPlayer } from "./IPlayer";
import { Player } from "./Player";

class PlayerList extends Component<{}, { players: IPlayer[] }> {

  intervalID!: NodeJS.Timeout;

  constructor(
    props: { players: IPlayer[] } | Readonly<{ players: IPlayer[] }>
  ) {
    super(props);
    this.state = {
      players: [
        { name: "John", score: 1 },
        { name: "Jane", score: 2 },
      ],
    };
  }

  componentDidMount() {
    const oneSecond = 1000;
    this.intervalID = setInterval(() => {
      this.setState({
        players: [
          {
            name: "Anonymous",
            score: 0,
          },
        ],
      });
    }, oneSecond);
  }

  componentDidUpdate(prevProps: any) {
    console.log("prevProps", prevProps);
  }

  componentWillUnmount() {
    clearInterval(this.intervalID);
  }

  render() {
    return (
      <ul>
        {this.state.players.map((player: IPlayer, i: number) => (
          <li key={i}>
            <Player name={player.name} score={player.score} />
          </li>
        ))}
      </ul>
    );
  }
}

export default PlayerList;

```


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

Hooks

- useState
- useEffect
- useContext
- useReducer
- useRef

React.useState



React.useEffect

PageTitle.tsx — my-todo-app

ColorPicker.tsx U | PageTitle.tsx U | players.tsx U

src > components > PageTitle.tsx > PageTitle > constructor

```tsx
import React, { Component } from "react";

export default class PageTitle extends Component<{}, { name: string }> {
  constructor(props: any) {
    super(props);
    this.state = {
      name: "",
    };
  }

  componentDidMount() {
    document.title = this.state.name;
  }

  componentDidUpdate() {
    document.title == `Hi, ${this.state.name}`;
  }

  render() {
    return (
      <div>
        <p>Use the input field below to rename this page!</p>
        <input
          onChange={({ target }) => this.setState({ name: target.value })}
          value={this.state.name}
          type="text"
        />
      </div>
    );
  }
}
```

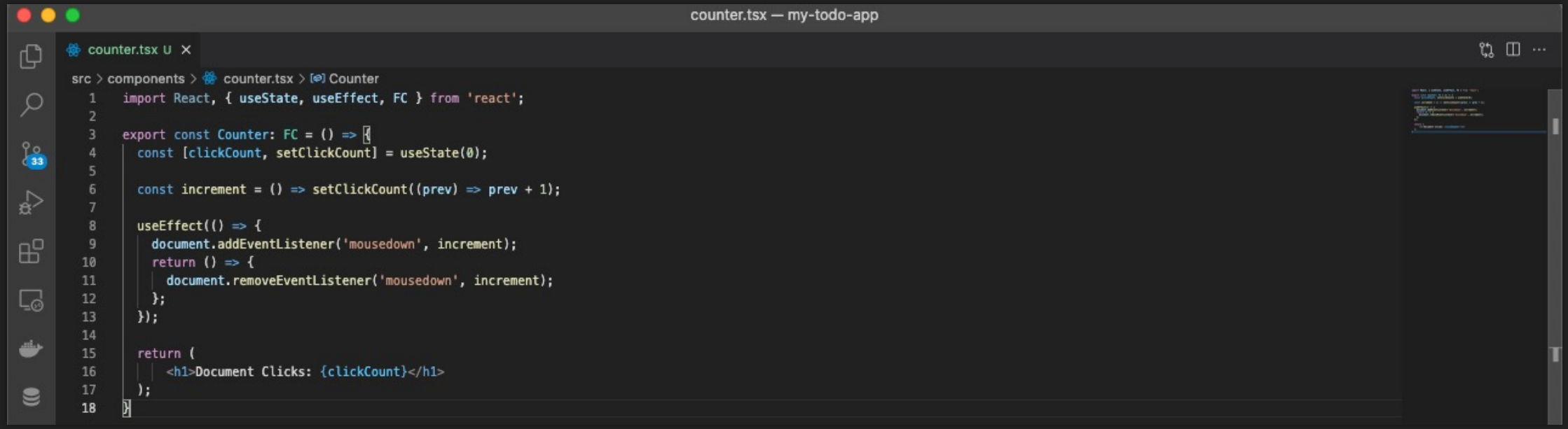
PageTitle.tsx — my-todo-app

PageTitle.tsx U

src > components > PageTitle.tsx > PageTitle

```tsx
import React, { useState, useEffect, FC } from 'react';

export const PageTitle: FC = () => {
  const [name, setName] = useState('');

  useEffect(() => {
    document.title = `Hi, ${name}`;
  });

  return (
    <div>
      <p>Use the input field below to rename this page!</p>
      <input onChange={({target}) => setName(target.value)} value={name} type='text' />
    </div>
  );
}
```


React.useEffect



```
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…