```
import React from "react";
import ReactDOM from "react-dom";

class App extends React.Component {
  constructor(props) {
    super(props);
    this.state = { lat: null };
  }

  render() {
    navigator.geolocation.getCurrentPosition(
      (position) => console.log(position),
      (err) => console.log(err)
    );

    return <div>latitude:{this.state.lat}</div>;
  }
}
ReactDOM.render(<App />, document.querySelector("#root"));
```

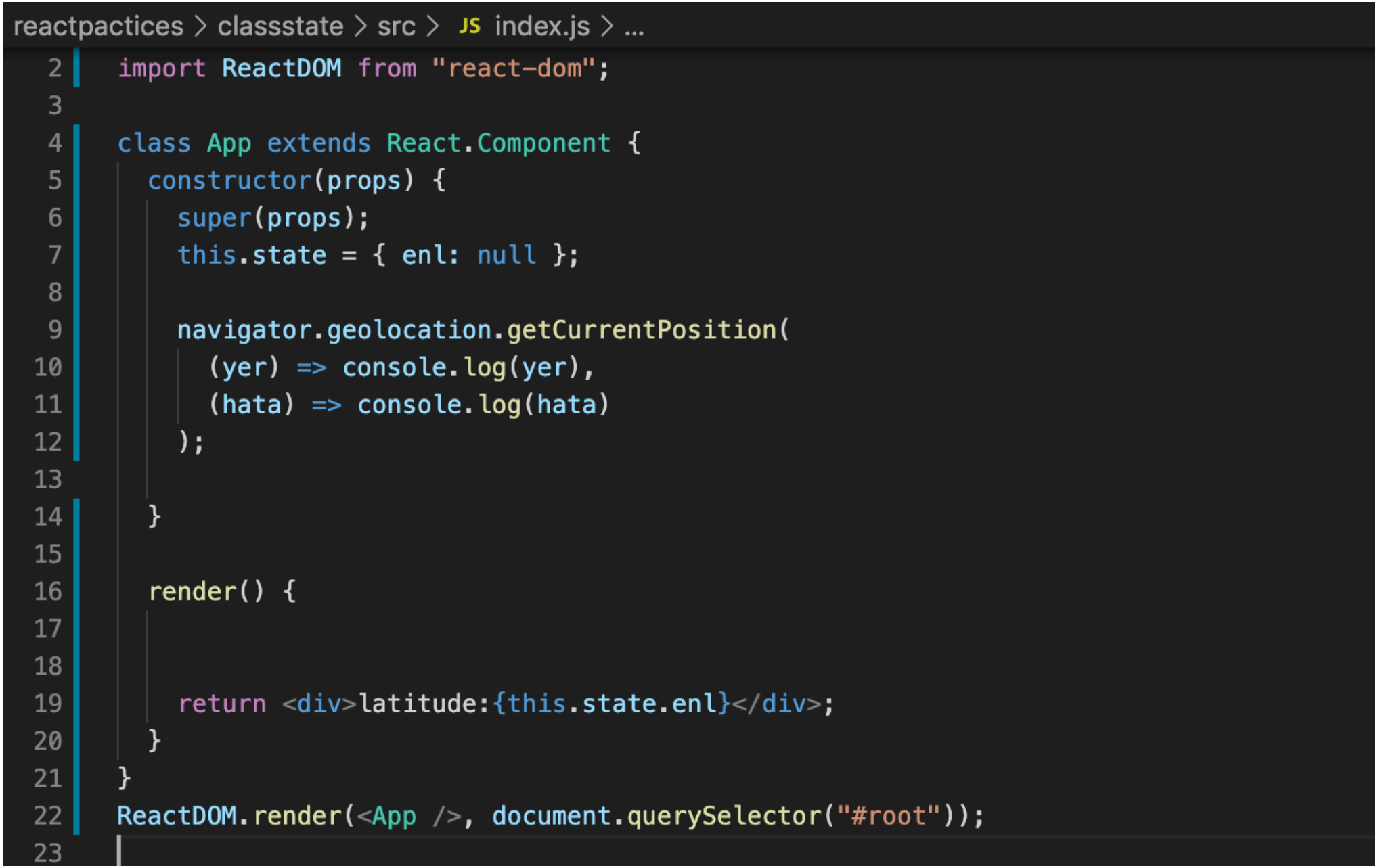
> *Since `window` is the global object, we can access `navigator` without specifying `window.navigator`*

The `window.navigator` property exposed by browsers points to a **Navigator object** which is a *container object* that makes a lot of Web Platform APIs available to us.

The `geolocation` object provides the following methods:

- `getCurrentPosition()`
- `watchPosition()`
- `clearWatch()`

The first one is used to get the current position coordinates. When we call this method for the first time, the browser automatically asks the user for the permission to share this information to us.


reactpactices > classstate > src > JS index.js > ...
2 import ReactDOM from "react-dom";
3
4 class App extends React.Component {
5 constructor(props) {
6 super(props);
7 this.state = { enl: null };
8
9 navigator.geolocation.getCurrentPosition(
10 (yer) => console.log(yer),
11 (hata) => console.log(hata)
12);
13
14 }
15
16 render() {
17
18
19 return latitude:{this.state.enl} ;
20 }
21 }
22 ReactDOM.render(<App />, document.querySelector("#root"));
23

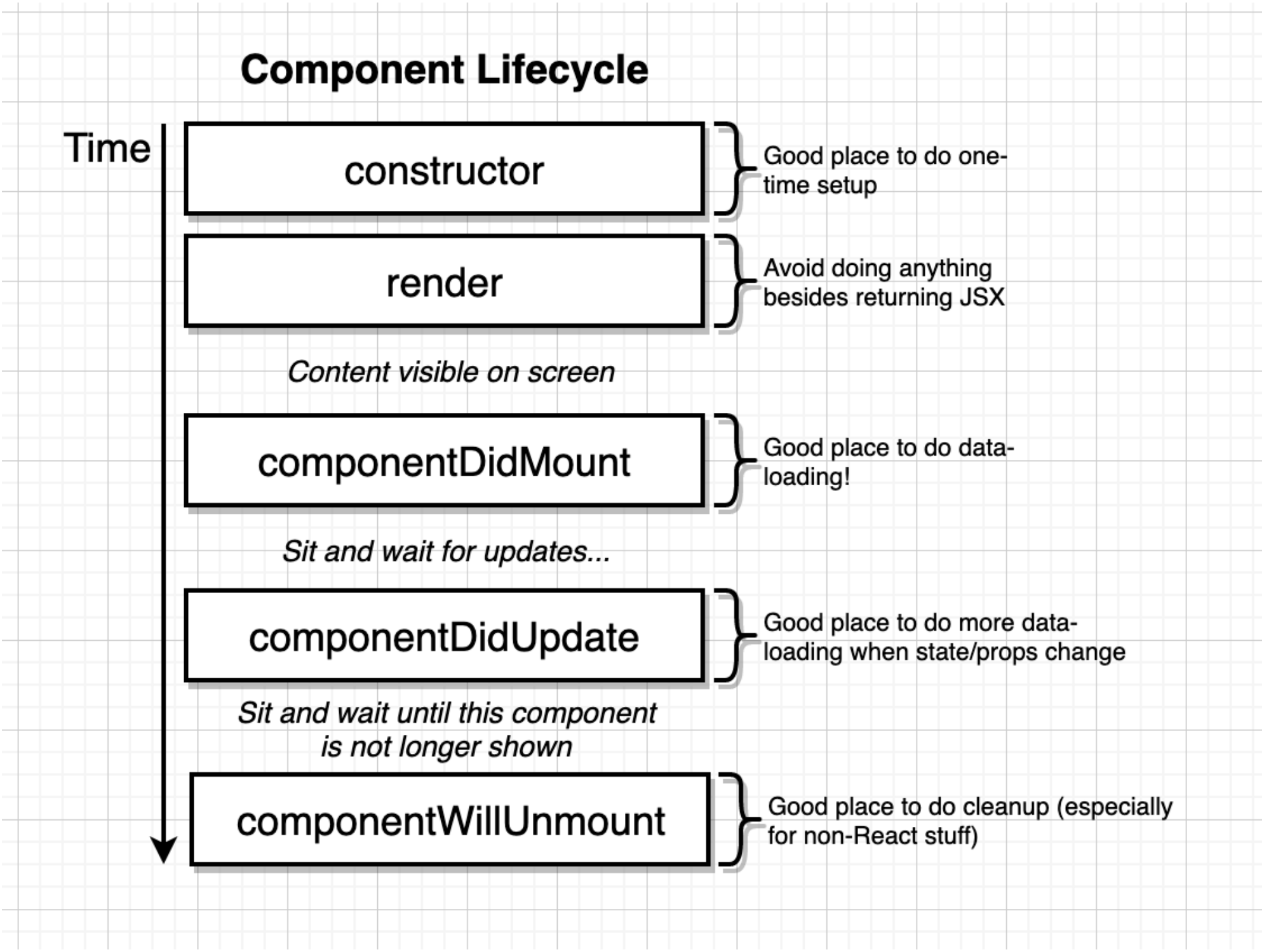
reactpactices > classstate > src > JS index.js > ...

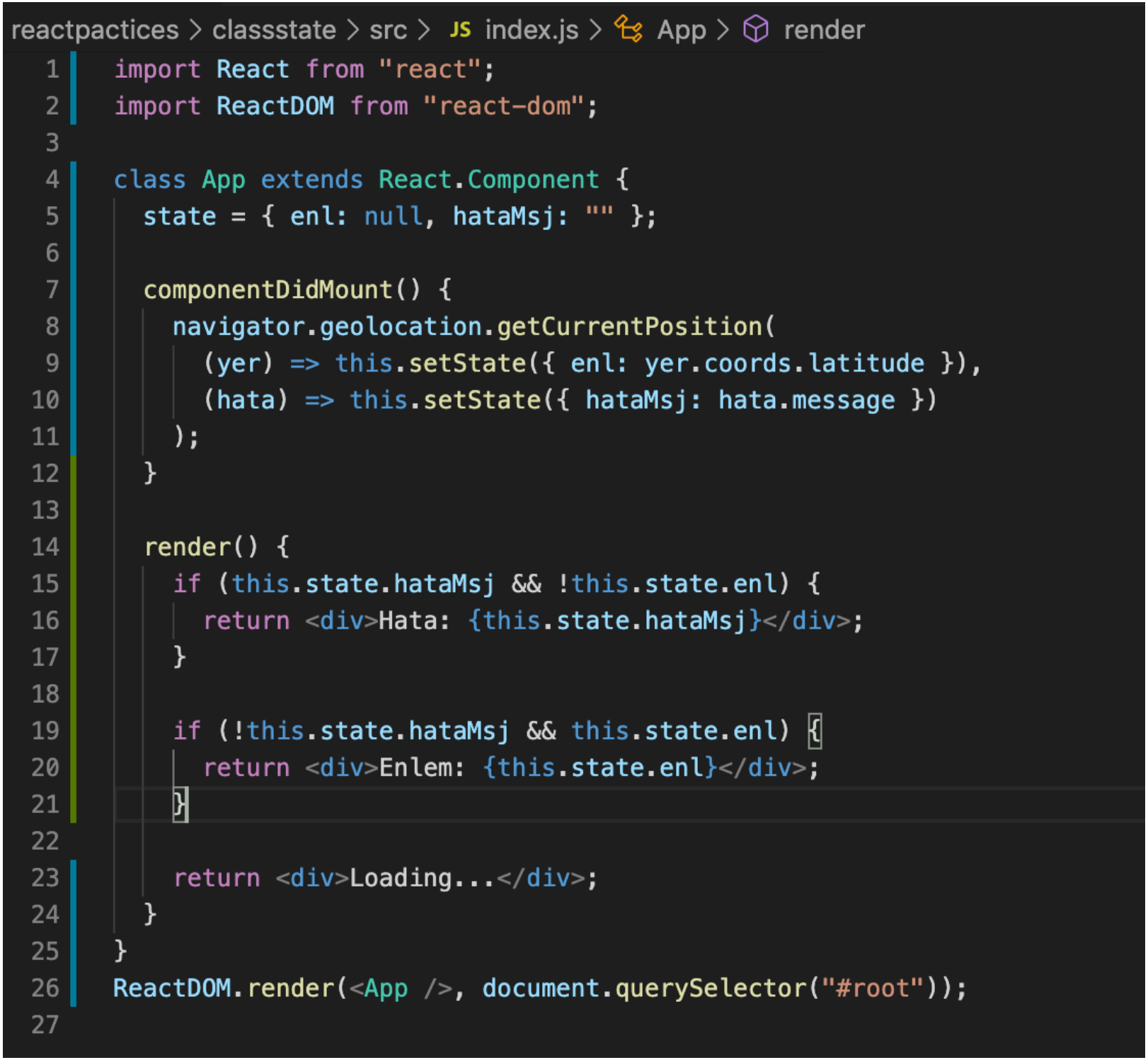
```
2   import ReactDOM from "react-dom";
3
4   class App extends React.Component {
5     constructor(props) {
6       super(props);
7       this.state = { enl: null };
8
9       navigator.geolocation.getCurrentPosition(
10        (yer) => {
11          this.setState({ enl: yer.coords.latitude });
12        },                    Not: this.state = {enl:yer.coord.latitude}
13        (hata) => console.log(hata)
14      );
15    }
16
17    render() {
18      return <div>latitude:{this.state.enl}</div>;
19    }
20  }
21  ReactDOM.render(<App />, document.querySelector("#root"));
22
```

```
import ReactDOM from "react-dom";

class App extends React.Component {
  constructor(props) {
    super(props);
    this.state = { enl: null, hataMsj: "" };

    navigator.geolocation.getCurrentPosition(
      (yer) => {
        this.setState({ enl: yer.coords.latitude });
      },
      (hata) => {
        this.setState({ hataMsj: hata.message });
      }
    );
  }

  render() {
    return (
      <div>
        Enlem: {this.state.enl}
        Hata: {this.state.hataMsj}
      </div>

    );
  }
}
ReactDOM.render(<App />, document.querySelector("#root"));
```

actpactices > classstate > src > JS index.js > App > render

```
import React from "react";
import ReactDOM from "react-dom";

class App extends React.Component {
  constructor(props) {
    super(props);
    this.state = { enl: null, hataMsj: "" };

    navigator.geolocation.getCurrentPosition(
      (yer) => {
        this.setState({ enl: yer.coords.latitude });
      },
      (hata) => {
        this.setState({ hataMsj: hata.message });
      }
    );
  }

  render() {
    if (this.state.hataMsj && !this.state.enl) {
      return <div>Hata: {this.state.hataMsj}</div>;
    }

    if (!this.state.hataMsj && this.state.enl) {
      return <div>Enlem: {this.state.enl}</div>;
    }

    return <div>Loading...</div>;
  }
}
ReactDOM.render(<App />, document.querySelector("#root"));

```


Component Lifecycle
Time
constructor
Good place to do one-time setup
render
Avoid doing anything besides returning JSX
Content visible on screen
componentDidMount
Good place to do data-loading!
Sit and wait for updates...
componentDidUpdate
Good place to do more data-loading when state/props change
Sit and wait until this component is not longer shown
componentWillUnmount
Good place to do cleanup (especially for non-React stuff)


reactpactices > classstate > src > JS index.js > App > render
1 import React from "react";
2 import ReactDOM from "react-dom";
3
4 class App extends React.Component {
5 state = { enl: null, hataMsj: "" };
6
7 componentDidMount() {
8 navigator.geolocation.getCurrentPosition(
9 (yer) => this.setState({ enl: yer.coords.latitude }),
10 (hata) => this.setState({ hataMsj: hata.message })
11);
12 }
13
14 render() {
15 if (this.state.hataMsj && !this.state.enl) {
16 return Hata: {this.state.hataMsj} ;
17 }
18
19 if (!this.state.hataMsj && this.state.enl) {
20 return Enlem: {this.state.enl} ;
21 }
22
23 return Loading... ;
24 }
25 }
26 ReactDOM.render(<App />, document.querySelector("#root"));
27

reactpactices > classstate > src > JS index.js > App > render

```
1  import React from "react";
2  import ReactDOM from "react-dom";
3  import SeasonDisplay from "./SeasonDisplay";
4
5  class App extends React.Component {
6    state = { enl: null, hataMsj: "" };
7
8    componentDidMount() {
9      navigator.geolocation.getCurrentPosition(
10       (yer) => this.setState({ enl: yer.coords.latitude }),
11       (hata) => this.setState({ hataMsj: hata.message })
12     );
13   }
14
15   render() {
16     if (this.state.hataMsj && !this.state.enl) {
17       return <div>Hata: {this.state.hataMsj}</div>;
18     }
19
20     if (!this.state.hataMsj && this.state.enl) {
21       return <SeasonDisplay enlem={this.state.enl} />;
22     }
23
24     return <div>Loading...</div>;
25   }
26 }
27 ReactDOM.render(<App />, document.querySelector("#root"));
28
```

JS index.js JS SeasonDisplay.js ●

reactpactices > classstate > src > JS SeasonDisplay.js > SeasonDisplay

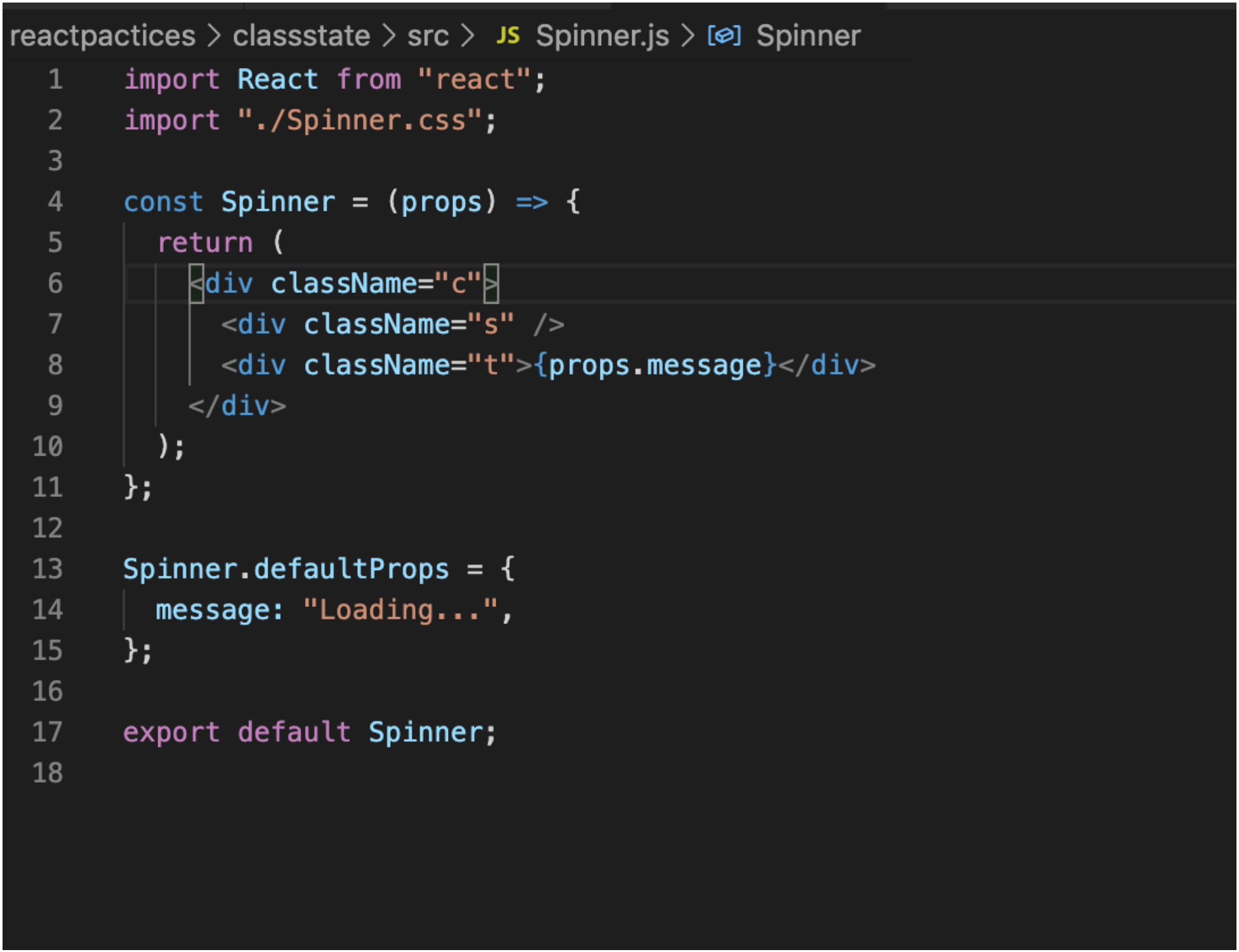
```
1   import React from 'react';
2
3   const getSeason = (en, ay) => {
4     if (ay > 2 && ay < 9) {
5       return en > 0 ? 'summer' : 'winter';
6     } else {
7       return en > 0 ? 'winter' : 'summer';
8     }
9   };
10
11  const SeasonDisplay = props => {
12    //console.log(props.enlem)
13    const season = getSeason(props.enlem, new Date().getMonth());
14
15    return (
16      <div>
17        SeasonDisplay
18      </div>
19    );
20  };
21
22  export default SeasonDisplay;
23
```

eactpactices > classstate > src > JS SeasonDisplay.js > SeasonDisplay

```
4   const seasonConfig = {
5     summer: {
6       text: "Yaz dostum!",
7       Background:
8         "https://images.unsplash.com/photo-1495107334309-fcf20504a5ab?ixlib=rb-1.2.1&ixid=eyJhcHBfaWQiOjEyMDd9&auto=fo
9     },
10    winter: {
11      text: "..az ilerde, bahar",
12      Background:
13        "https://images.unsplash.com/photo-1453306458620-5bbef13a5bca?ixlib=rb-1.2.1&ixid=eyJhcHBfaWQiOjEyMDd9&auto=fo
14    },
15  };
16
17  const getSeason = (en, ay) => {
18    if (ay > 2 && ay < 9) {
19      return en > 0 ? "summer" : "winter";
20    } else {
21      return en > 0 ? "winter" : "summer";
22    }
23  };
24
25  const SeasonDisplay = (props) => {
26    //console.log(props.enlem)
27    const season = getSeason(props.enlem, new Date().getMonth());
28    const { text, Background } = seasonConfig[season];
29
30    return (
31      <div
32        className={`season-display ${season}`}
33        style={{ background: `url(${Background})` }}
34      >
35        <h3>{text}</h3>
36      </div>
37    );
38  };
39
40  export default SeasonDisplay;
```

eactpactices > classstate > src > # SeasonDisplay.css > ...

```css
*,
*::after,
* ::before,
:root {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
.season-display {
  min-height: 100vh;
  position: relative;
}

h3 {
  font-size: 3rem;
  text-align: center;
  position: absolute;
  padding: 1rem;
  box-shadow: 0 0 20px #333;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
}

.winter h3 {
  background-color: aliceblue;
  color: blue;
}

.summer h3 {
  background-color: orange;
  color: red;
}

```


reactpactices > classstate > src > JS Spinner.js > Spinner
1 import React from "react";
2 import "./Spinner.css";
3
4 const Spinner = (props) => {
5 return (
6
7
8 {props.message}
9
10);
11 };
12
13 Spinner.defaultProps = {
14 message: "Loading...",
15 };
16
17 export default Spinner;
18

reactpactices > classstate > src > # Spinner.css > .c

```css
.c {
  position: absolute;
  width: 100%;
  height: 100%;
  background: #1d1f20;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
}
.t {
  margin-top: 5rem;
  font-size: 1rem;
  font-family: "Open Sans", "Helvetica Neue";
  color: whitesmoke;
  text-align: center;
}
.s {
  width: 10px;
  height: 10px;
  border-radius: 50%;
  float: left;
  background: rgb(231, 0, 0);
  transition: all 0.2s;
  animation: r5 1s 0s ease-out infinite;
}
@keyframes r5 {
  0% {
    box-shadow: 0 0 8px 6px rgb(231, 0, 0), 0 0 0px 0px transparent,
      0 0 0px 0px rgb(231, 0, 0);
  }
  10% {
    box-shadow: 0 0 8px 6px rgb(231, 0, 0), 0 0 12px 10px transparent,
      0 0 12px 14px rgb(231, 0, 0);
  }
  100% {
    box-shadow: 0 0 8px 6px rgba(26, 140, 255, 0), 0 0 0px 40px transparent,
      0 0 0px 100px rgba(26, 140, 255, 0);
  }
}
```

Ln 7, Col 3 (23 selected) Spaces: 2 UTF-8

reactpactices > classstate > src > JS index.js > App > render

```
1  import React from "react";
2  import ReactDOM from "react-dom";
3  import SeasonDisplay from "./SeasonDisplay";
4  import Spinner from "./Spinner";
5
6  class App extends React.Component {
7    state = { enl: null, hataMsj: "" };
8
9    componentDidMount() {
10     navigator.geolocation.getCurrentPosition(
11       (yer) => this.setState({ enl: yer.coords.latitude }),
12       (hata) => this.setState({ hataMsj: hata.message })
13     );
14   }
15
16   renderContent() {
17     if (this.state.hataMsj && !this.state.enl) {
18       return <div>Hata: {this.state.hataMsj}</div>;
19     }
20
21     if (!this.state.hataMsj && this.state.enl) {
22       return <SeasonDisplay enlem={this.state.enl} />;
23     }
24
25     return <Spinner message="Please accept location request" />;
26   }
27
28   render() {
29     return <div className="border red">{this.renderContent()}</div>;
30   }
31 }
32 ReactDOM.render(<App />, document.querySelector("#root"));
33
```