React JS

Declarative approach that developers needed

Short overview of the most popular tool for creating user interfaces

by ishandry

Get started

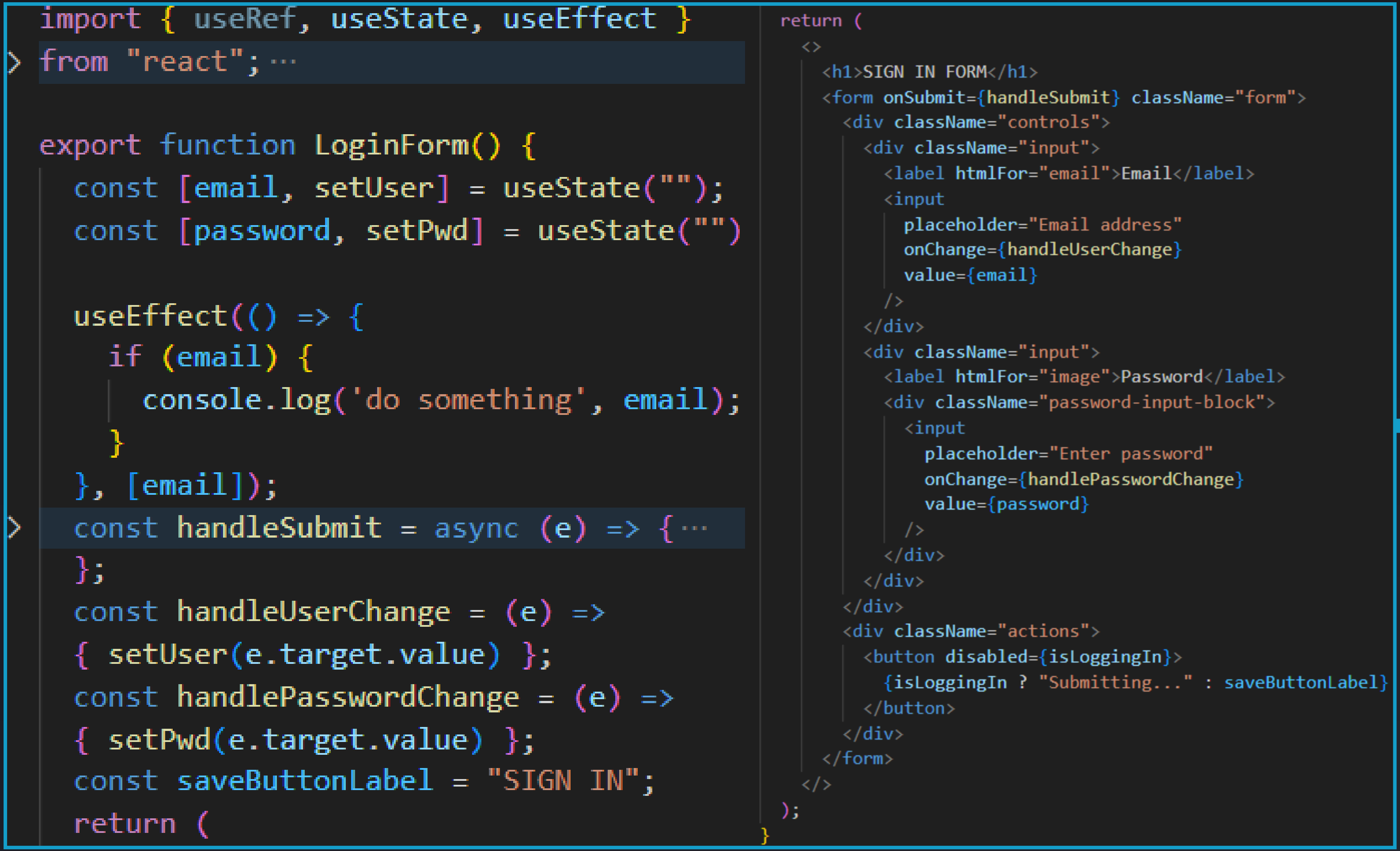
At the end of the presentation, you will:

understand 90% of React code

know how React works

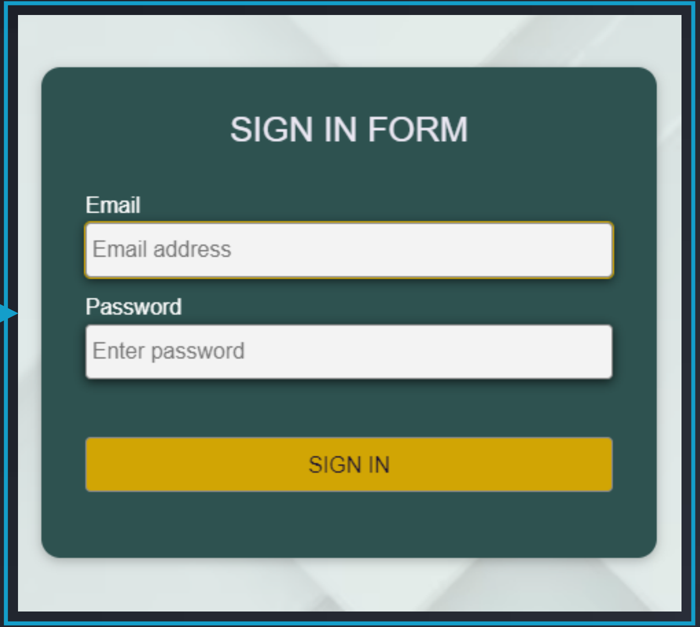
be aware of the tools frequently used with React

Hands on example: login form



```
import { useRef, useState, useEffect }
from "react"; ···

export function LoginForm() {
  const [email, setUser] = useState("");
  const [password, setPwd] = useState("")

  useEffect(() => {
    if (email) {
      console.log('do something', email);
    }
  }, [email]);
  const handleSubmit = async (e) => {···
  };
  const handleUserChange = (e) =>
  { setUser(e.target.value) };
  const handlePasswordChange = (e) =>
  { setPwd(e.target.value) };
  const saveButtonLabel = "SIGN IN";
  return (
```

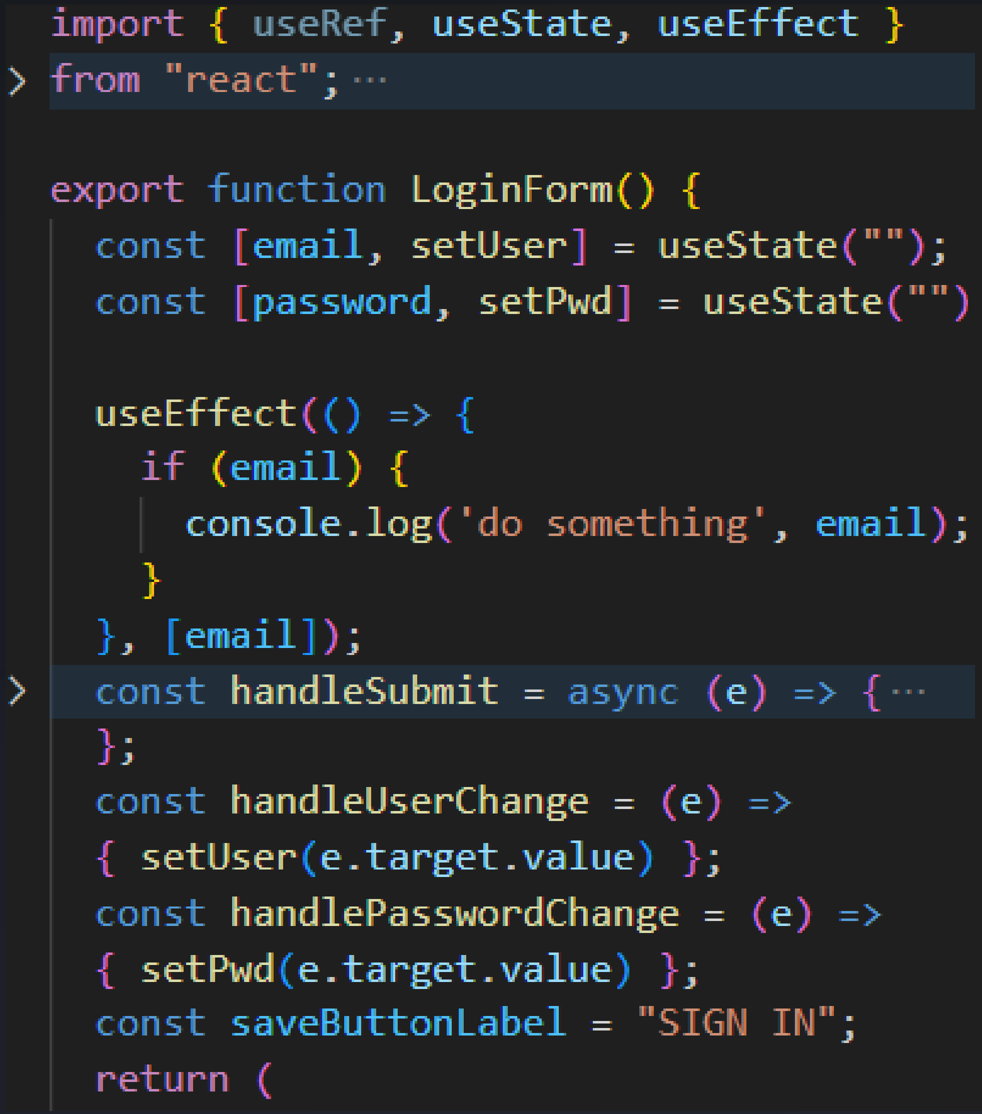
```
return (
  <>
    <h1>SIGN IN FORM</h1>
    <form onSubmit={handleSubmit} className="form">
      <div className="controls">
        <div className="input">
          <label htmlFor="email">Email</label>
          <input
            placeholder="Email address"
            onChange={handleUserChange}
            value={email}
          />
        </div>
        <div className="input">
          <label htmlFor="image">Password</label>
          <div className="password-input-block">
            <input
              placeholder="Enter password"
              onChange={handlePasswordChange}
              value={password}
            />
          </div>
        </div>
      </div>
      <div className="actions">
        <button disabled={isLoggingIn}>
          {isLoggingIn ? "Submitting..." : saveButtonLabel}
        </button>
      </div>
    </form>
  </>
);
}
```



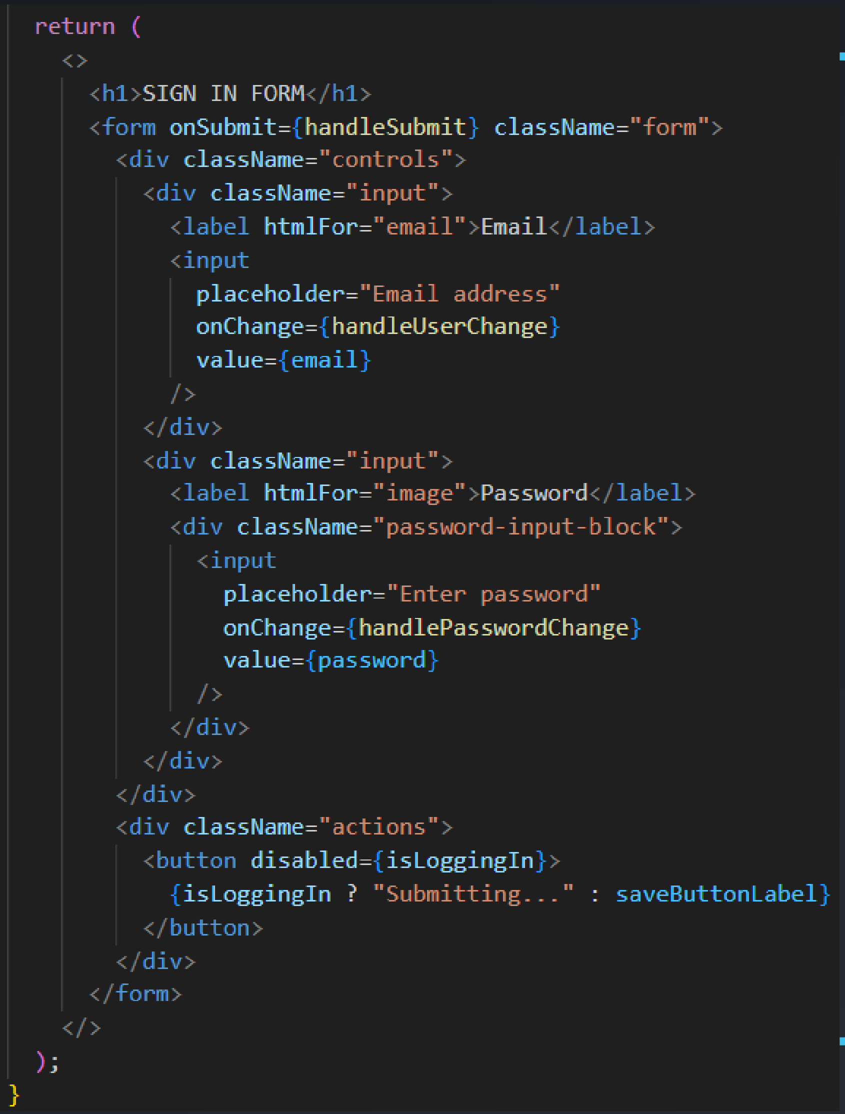
Login form breakdown

Imports

React functional component



```
import { useRef, useState, useEffect }
from "react";···

export function LoginForm() {
  const [email, setUser] = useState("");
  const [password, setPwd] = useState("")

  useEffect(() => {
    if (email) {
      console.log('do something', email);
    }
  }, [email]);
  const handleSubmit = async (e) => {···
  };
  const handleUserChange = (e) =>
  { setUser(e.target.value) };
  const handlePasswordChange = (e) =>
  { setPwd(e.target.value) };
  const saveButtonLabel = "SIGN IN";
  return (
```

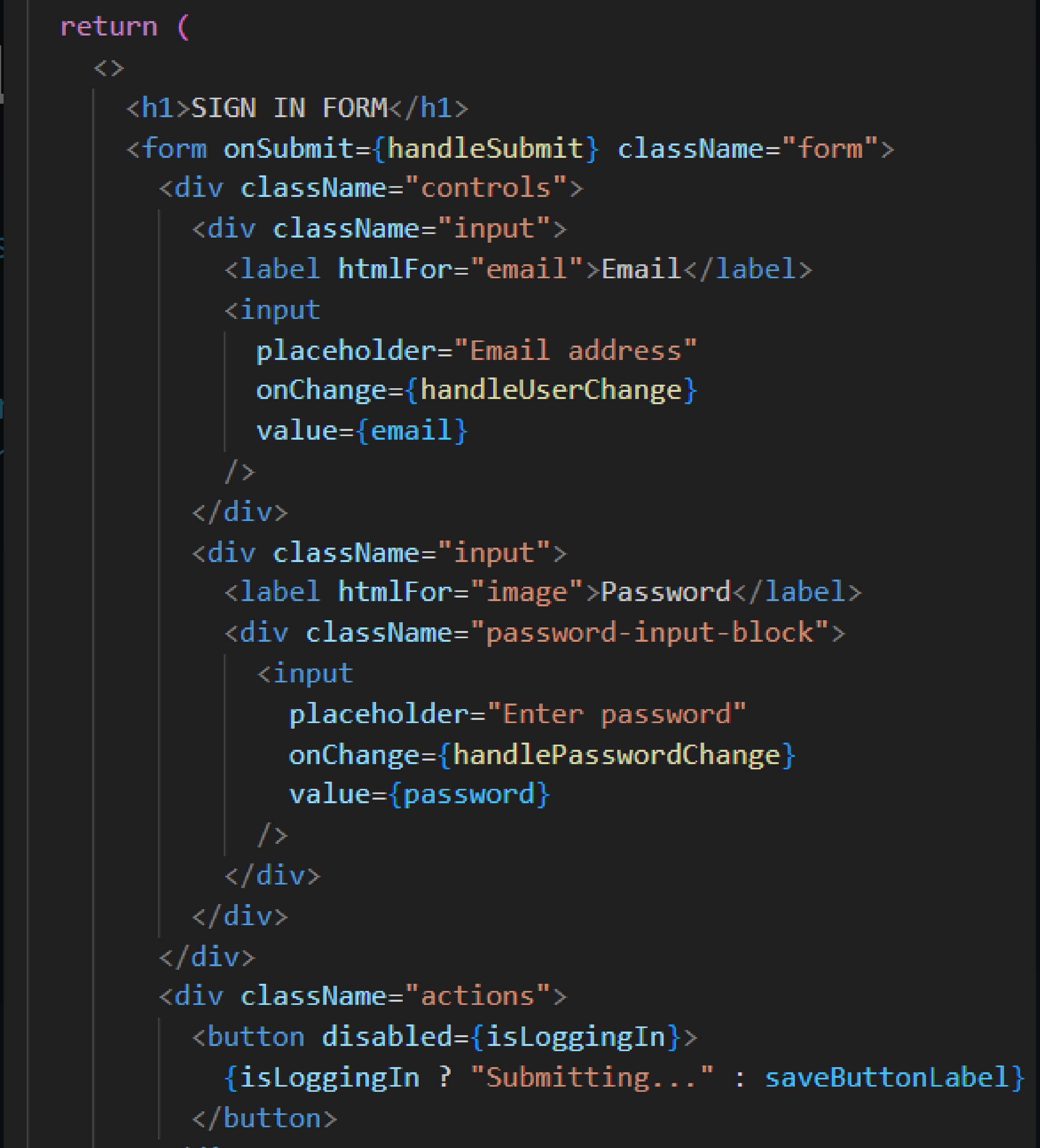


```
return (
  <>
    <h1>SIGN IN FORM</h1>
    <form onSubmit={handleSubmit} className="form">
      <div className="controls">
        <div className="input">
          <label htmlFor="email">Email</label>
          <input
            placeholder="Email address"
            onChange={handleUserChange}
            value={email}
          />
        </div>
        <div className="input">
          <label htmlFor="image">Password</label>
          <div className="password-input-block">
            <input
              placeholder="Enter password"
              onChange={handlePasswordChange}
              value={password}
            />
          </div>
        </div>
      </div>
      <div className="actions">
        <button disabled={isLoggingIn}>
          {isLoggingIn ? "Submitting..." : saveButtonLabel}
        </button>
      </div>
    </form>
  </>
);
}
```

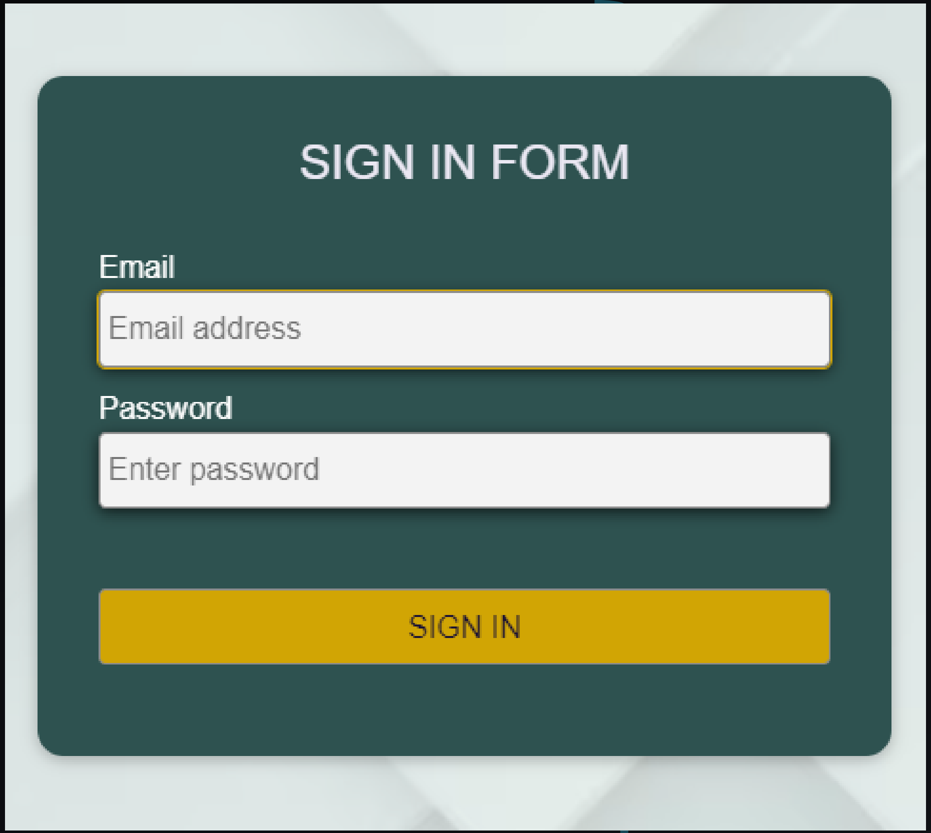
JSX

Imports

React
functio
compor



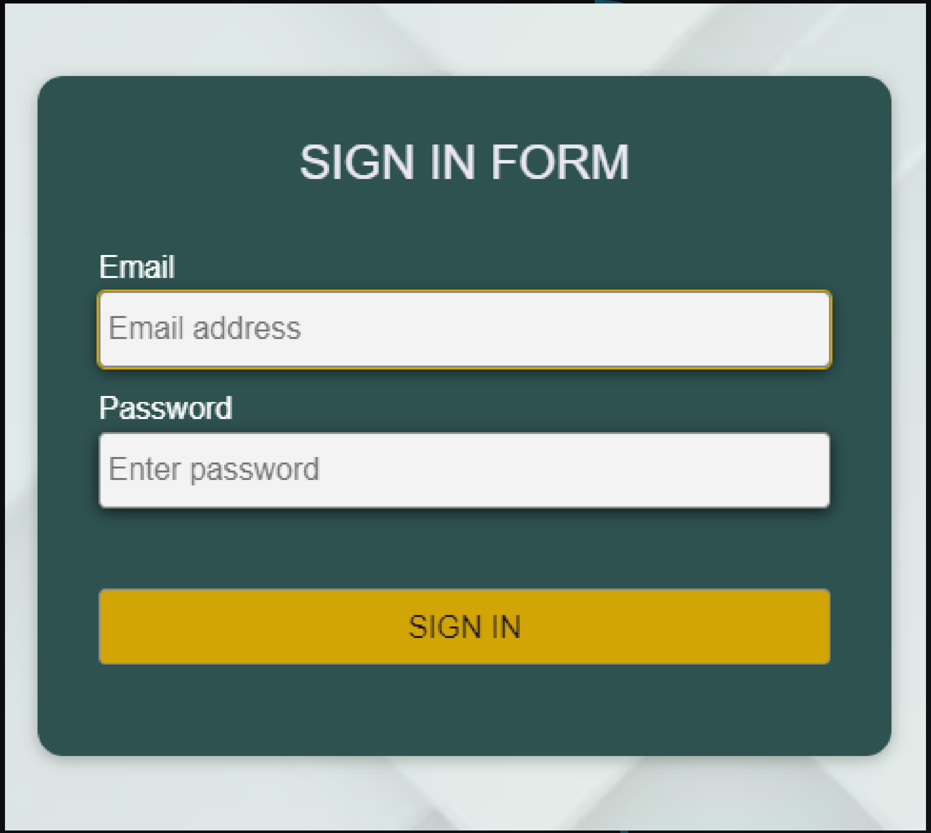
```
return (
  <>
    <h1>SIGN IN FORM</h1>
    <form onSubmit={handleSubmit} className="form">
      <div className="controls">
        <div className="input">
          <label htmlFor="email">Email</label>
          <input
            placeholder="Email address"
            onChange={handleUserChange}
            value={email}
          />
        </div>
        <div className="input">
          <label htmlFor="image">Password</label>
          <div className="password-input-block">
            <input
              placeholder="Enter password"
              onChange={handlePasswordChange}
              value={password}
            />
          </div>
        </div>
      </div>
      <div className="actions">
        <button disabled={isLoggingIn}>
          {isLoggingIn ? "Submitting..." : saveButtonLabel}
        </button>
```



Imports

React
functio
compo

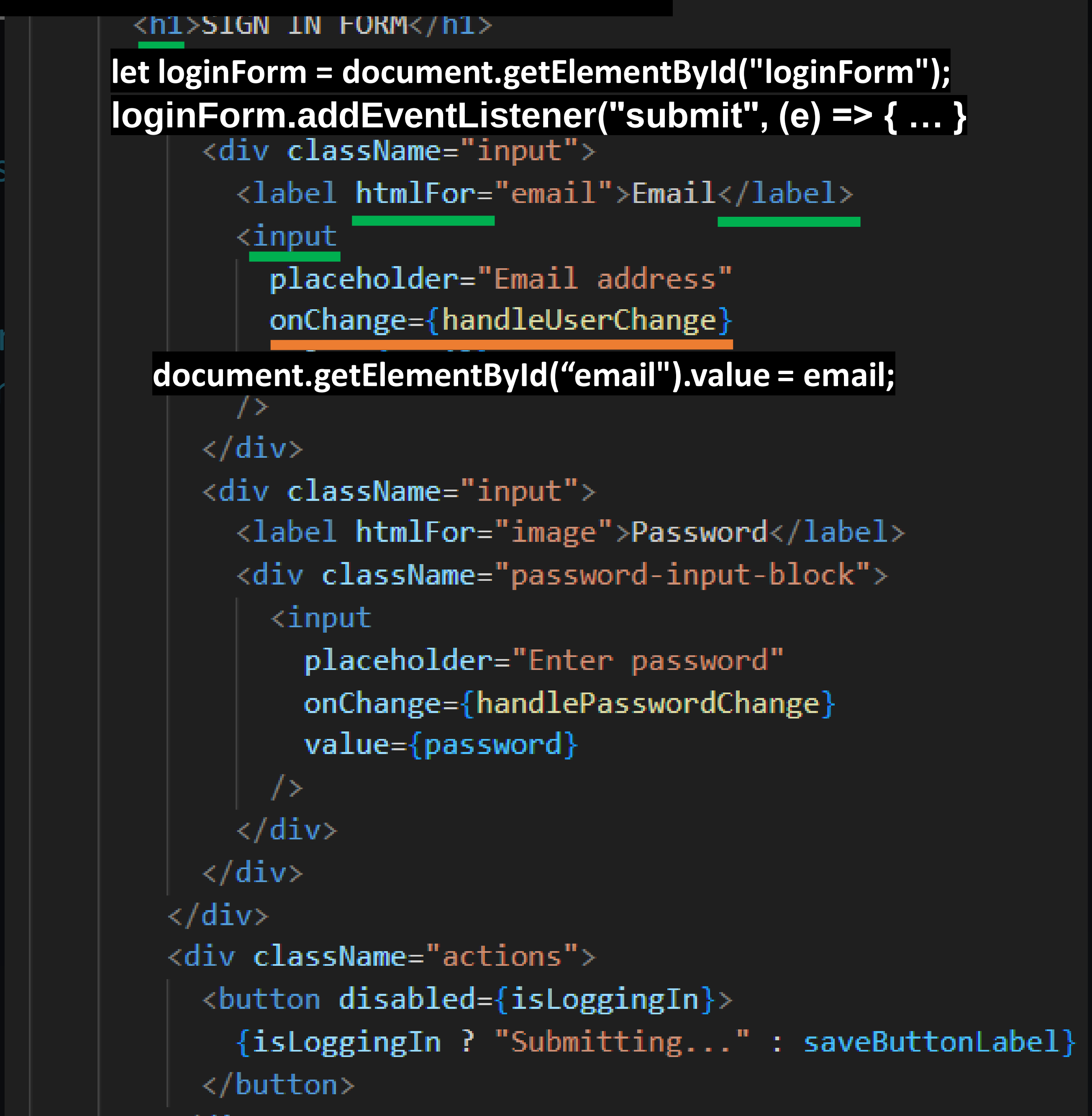
```
return (
  <>
    <h1>SIGN IN FORM</h1>
    <form onSubmit={handleSubmit} className="form">
      <div className="controls">
        <div className="input">
          <label htmlFor="email">Email</label>
          <input
            placeholder="Email address"
            onChange={handleUserChange}
            value={email}
          />
        </div>
        <div className="input">
          <label htmlFor="image">Password</label>
          <div className="password-input-block">
            <input
              placeholder="Enter password"
              onChange={handlePasswordChange}
              value={password}
            />
          </div>
        </div>
      </div>
      <div className="actions">
        <button disabled={isLoggingIn}>
          {isLoggingIn ? "Submitting..." : saveButtonLabel}
        </button>
```



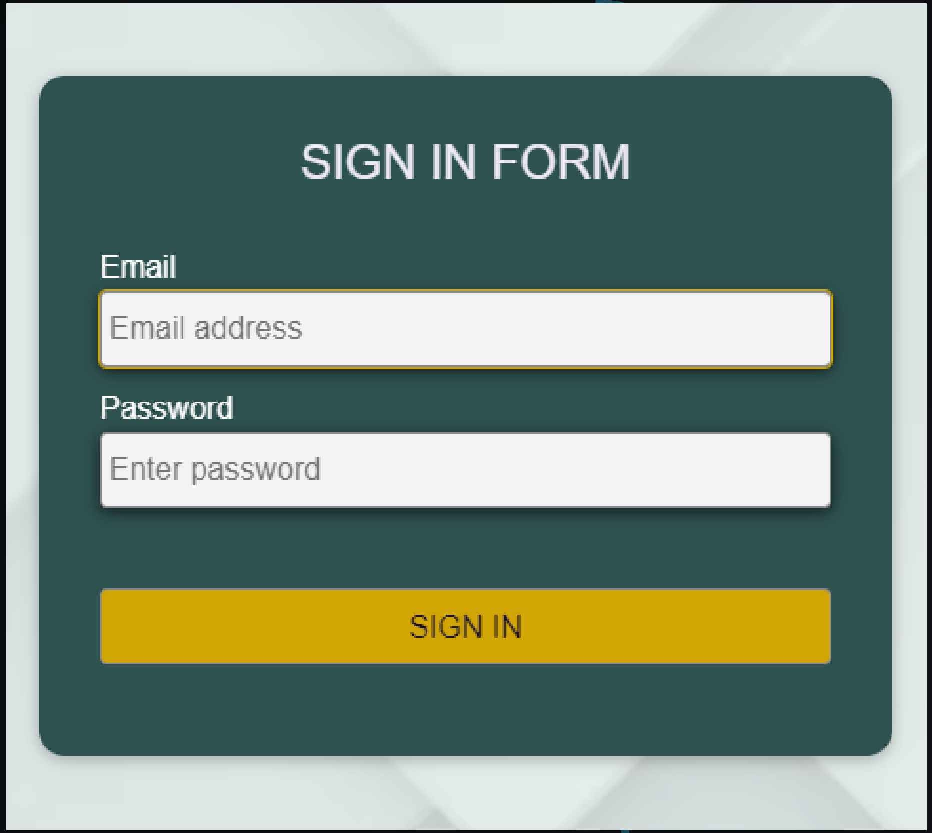
One can think of it as:

Imports

React
functior
compor

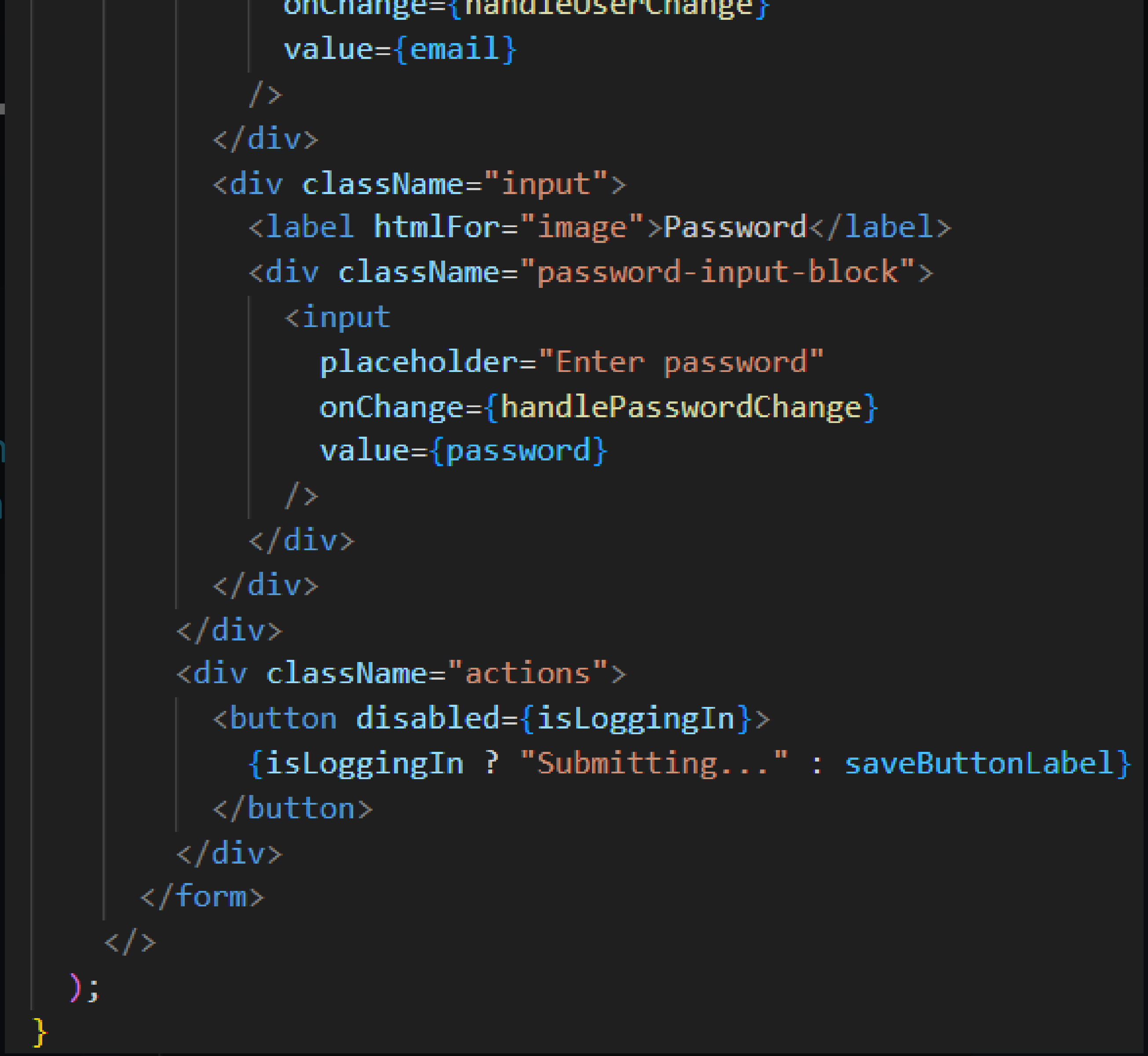


```
<h1>SIGN IN FORM</h1>
let loginForm = document.getElementById("loginForm");
loginForm.addEventListener("submit", (e) => { … }
    <div className="input">
      <label htmlFor="email">Email</label>
      <input
        placeholder="Email address"
        onChange={handleUserChange}
document.getElementById(“email").value = email;
      />
    </div>
    <div className="input">
      <label htmlFor="image">Password</label>
      <div className="password-input-block">
        <input
          placeholder="Enter password"
          onChange={handlePasswordChange}
          value={password}
        />
      </div>
    </div>
  </div>
  <div className="actions">
    <button disabled={isLoggingIn}>
      {isLoggingIn ? "Submitting..." : saveButtonLabel}
    </button>
```



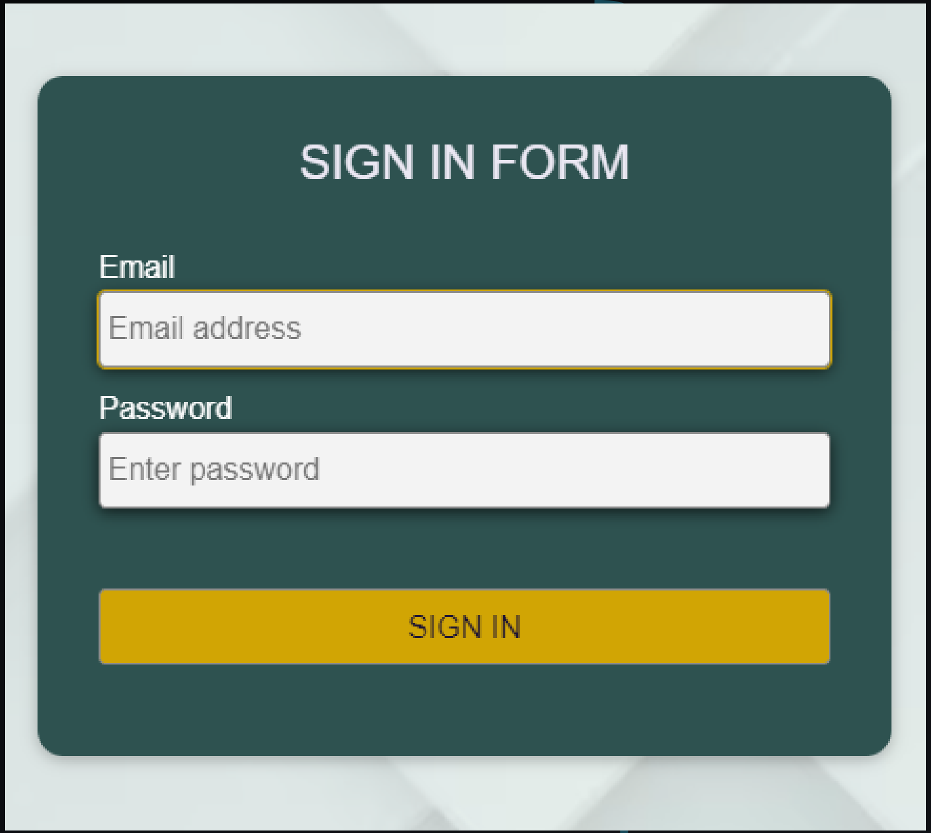
Imports

React function compon

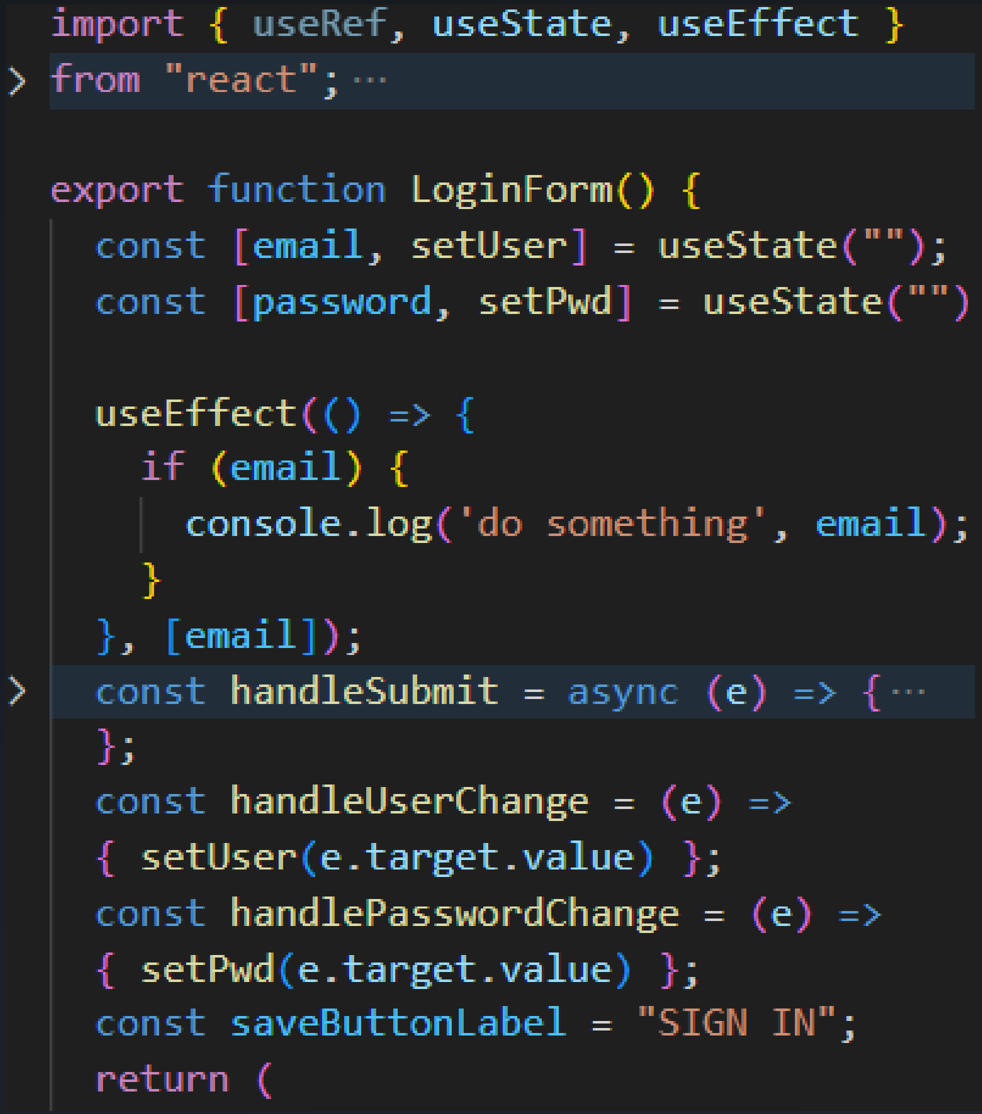


```
              onChange={handleUserChange}
              value={email}
            />
          </div>
          <div className="input">
            <label htmlFor="image">Password</label>
            <div className="password-input-block">
              <input
                placeholder="Enter password"
                onChange={handlePasswordChange}
                value={password}
              />
            </div>
          </div>
        </div>
        <div className="actions">
          <button disabled={isLoggingIn}>
            {isLoggingIn ? "Submitting..." : saveButtonLabel}
          </button>
        </div>
      </form>
    </>
  );
}
```

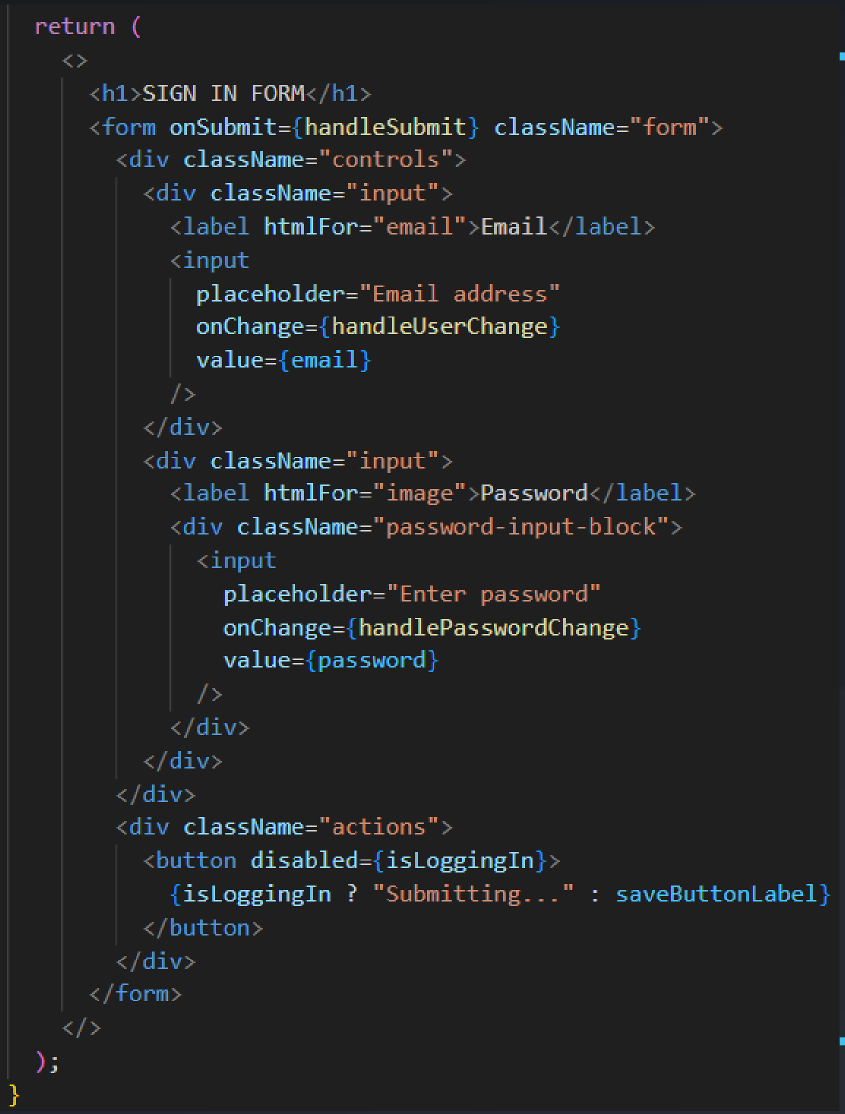
```
{ setPwd(e.target.value) };
const saveButtonLabel = "SIGN IN";
return (
```



Login form breakdown



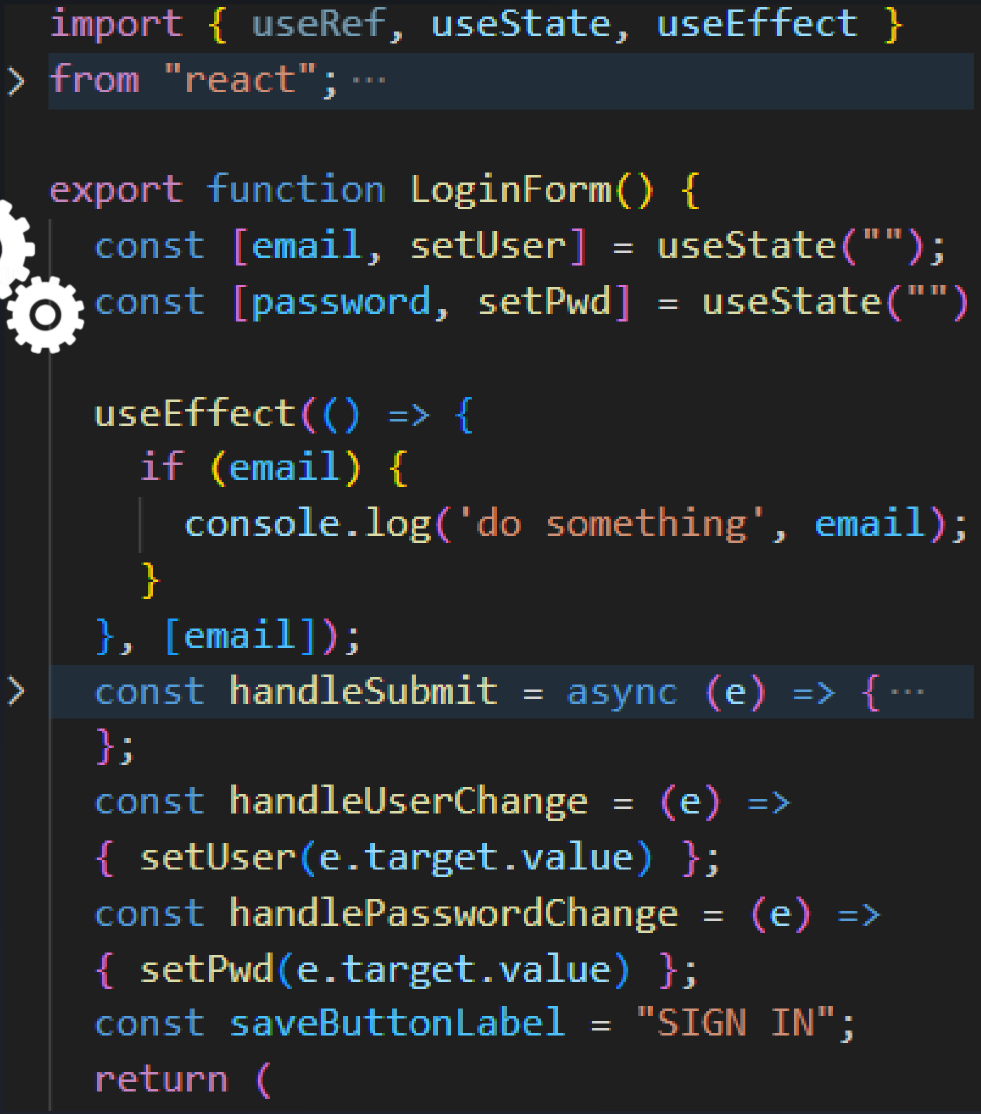
```
import { useRef, useState, useEffect }
> from "react"; ···

export function LoginForm() {
  const [email, setUser] = useState("");
  const [password, setPwd] = useState("")

  useEffect(() => {
    if (email) {
      console.log('do something', email);
    }
  }, [email]);
> const handleSubmit = async (e) => {···
  };
  const handleUserChange = (e) =>
  { setUser(e.target.value) };
  const handlePasswordChange = (e) =>
  { setPwd(e.target.value) };
  const saveButtonLabel = "SIGN IN";
  return (
```



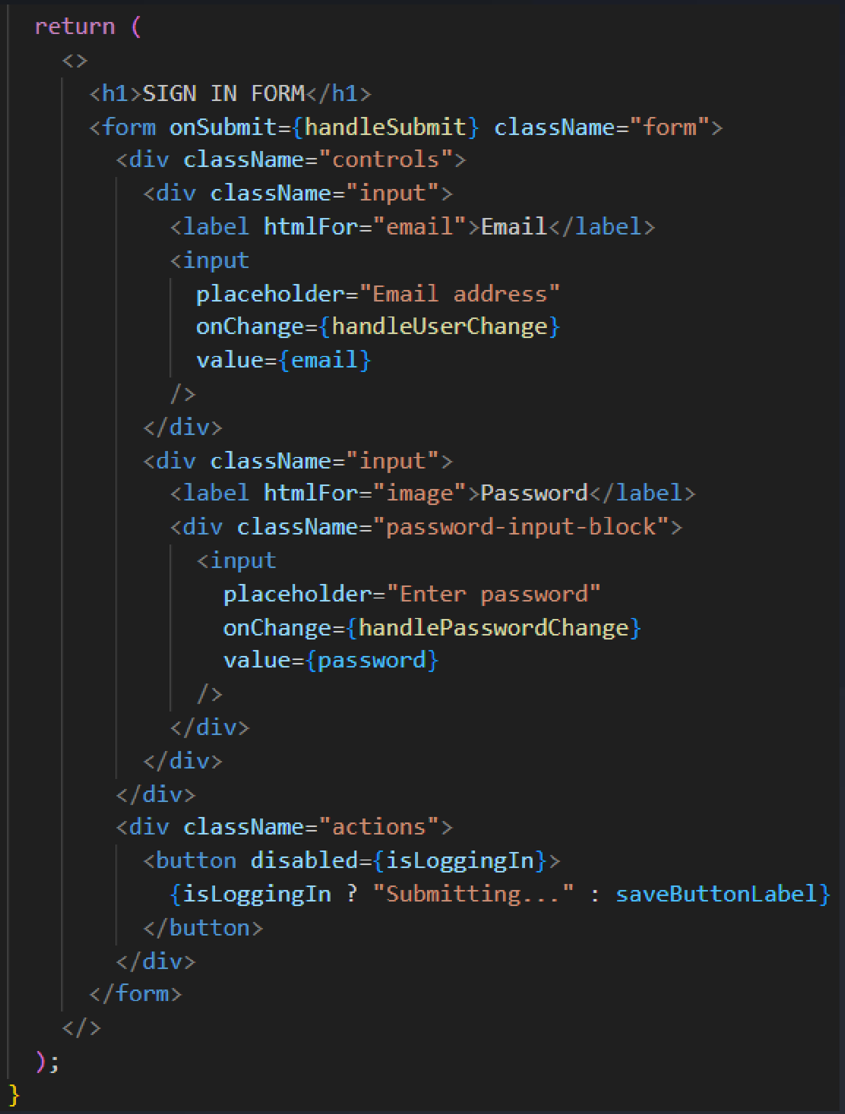
```
return (
  <>
    <h1>SIGN IN FORM</h1>
    <form onSubmit={handleSubmit} className="form">
      <div className="controls">
        <div className="input">
          <label htmlFor="email">Email</label>
          <input
            placeholder="Email address"
            onChange={handleUserChange}
            value={email}
          />
        </div>
        <div className="input">
          <label htmlFor="image">Password</label>
          <div className="password-input-block">
            <input
              placeholder="Enter password"
              onChange={handlePasswordChange}
              value={password}
            />
          </div>
        </div>
      </div>
      <div className="actions">
        <button disabled={isLoggingIn}>
          {isLoggingIn ? "Submitting..." : saveButtonLabel}
        </button>
      </div>
    </form>
  </>
);
}
```

JSX

Component's state. UseState hook

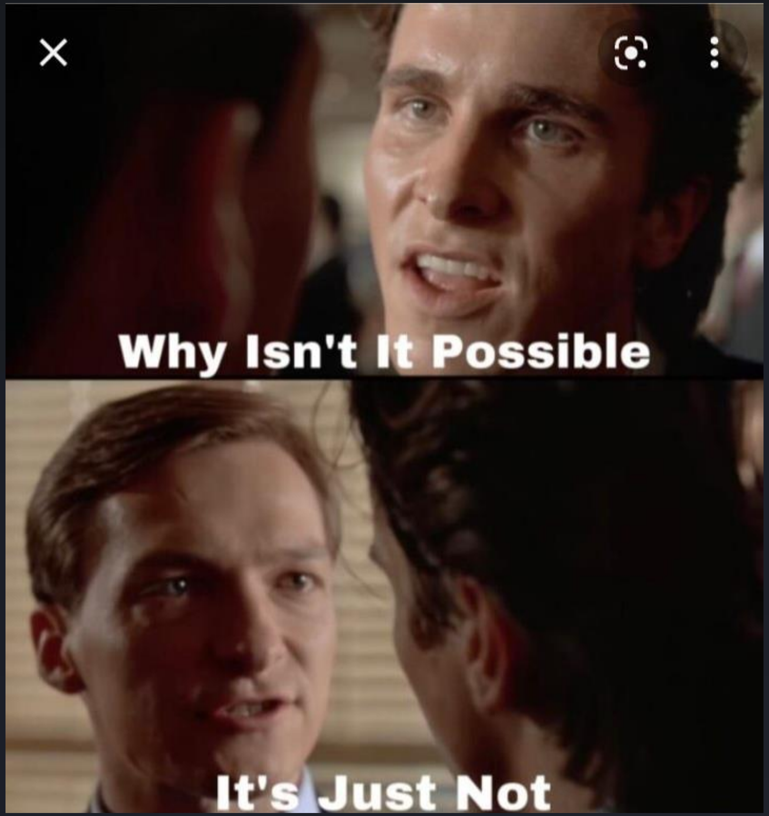


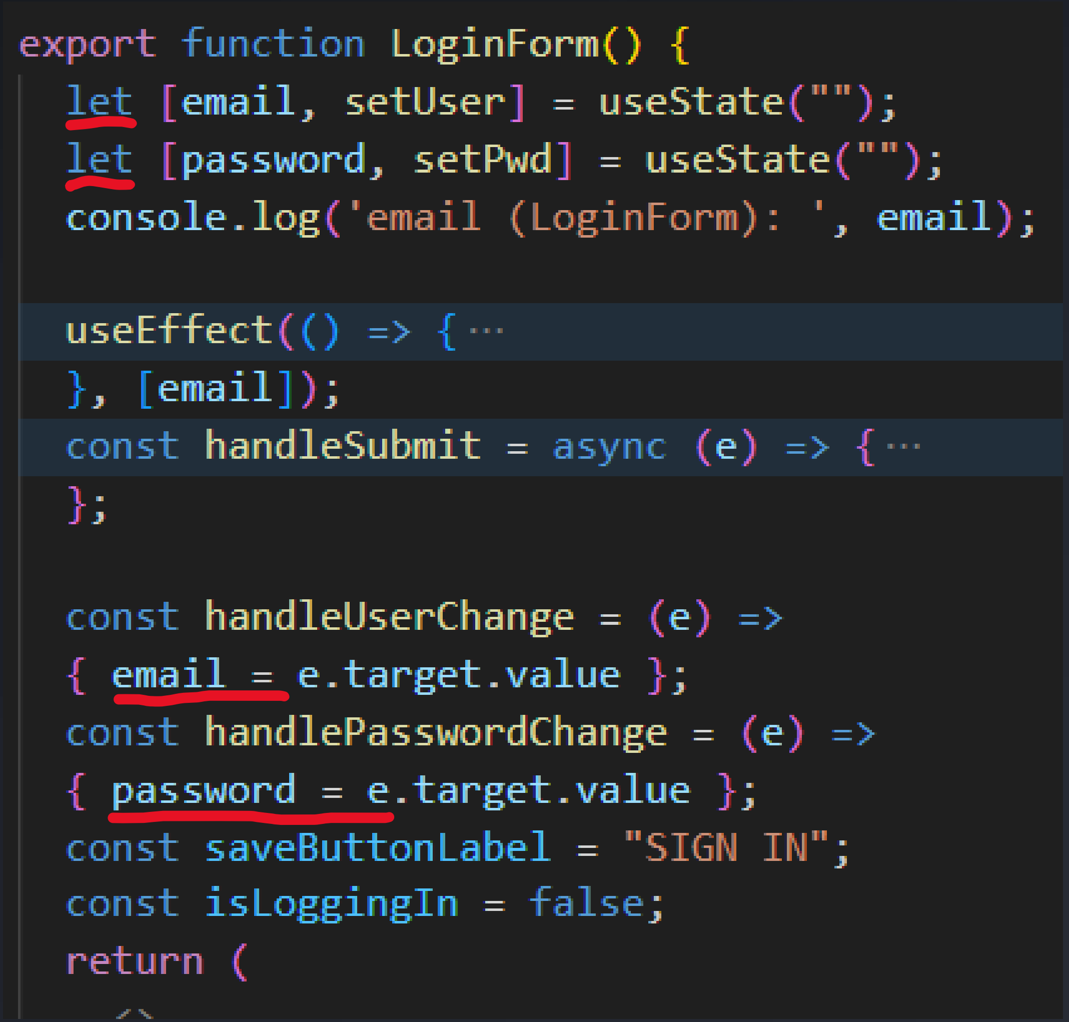
```
import { useRef, useState, useEffect }
from "react";…

export function LoginForm() {
  const [email, setUser] = useState("");
  const [password, setPwd] = useState("")

  useEffect(() => {
    if (email) {
      console.log('do something', email);
    }
  }, [email]);
  const handleSubmit = async (e) => {…
  };
  const handleUserChange = (e) =>
  { setUser(e.target.value) };
  const handlePasswordChange = (e) =>
  { setPwd(e.target.value) };
  const saveButtonLabel = "SIGN IN";
  return (
```



```
return (
  <>
    <h1>SIGN IN FORM</h1>
    <form onSubmit={handleSubmit} className="form">
      <div className="controls">
        <div className="input">
          <label htmlFor="email">Email</label>
          <input
            placeholder="Email address"
            onChange={handleUserChange}
            value={email}
          />
        </div>
        <div className="input">
          <label htmlFor="image">Password</label>
          <div className="password-input-block">
            <input
              placeholder="Enter password"
              onChange={handlePasswordChange}
              value={password}
            />
          </div>
        </div>
      </div>
      <div className="actions">
        <button disabled={isLoggingIn}>
          {isLoggingIn ? "Submitting..." : saveButtonLabel}
        </button>
      </div>
    </form>
  </>
);
}
```

Component’s state. How it works



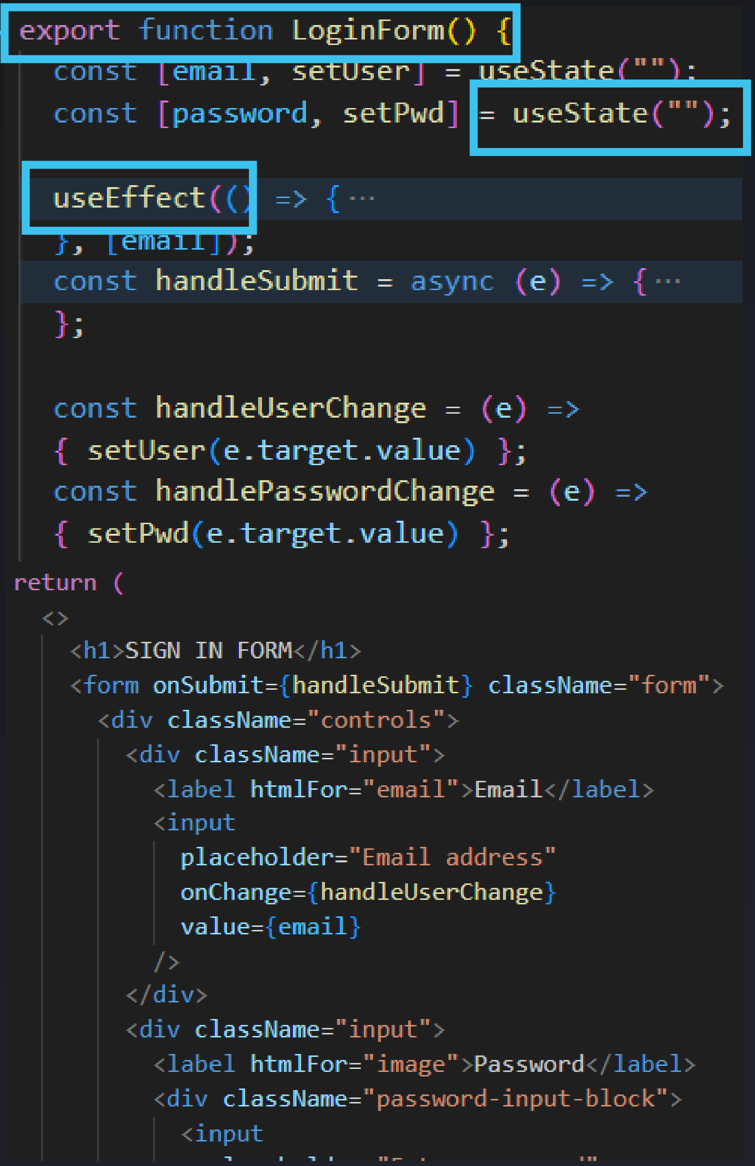


```
export function LoginForm() {
  let [email, setUser] = useState("");
  let [password, setPwd] = useState("");
  console.log('email (LoginForm): ', email);

  useEffect(() => {⋯
  }, [email]);
  const handleSubmit = async (e) => {⋯
  };

  const handleUserChange = (e) =>
  { email = e.target.value };
  const handlePasswordChange = (e) =>
  { password = e.target.value };
  const saveButtonLabel = "SIGN IN";
  const isLoggingIn = false;
  return (
```

✖

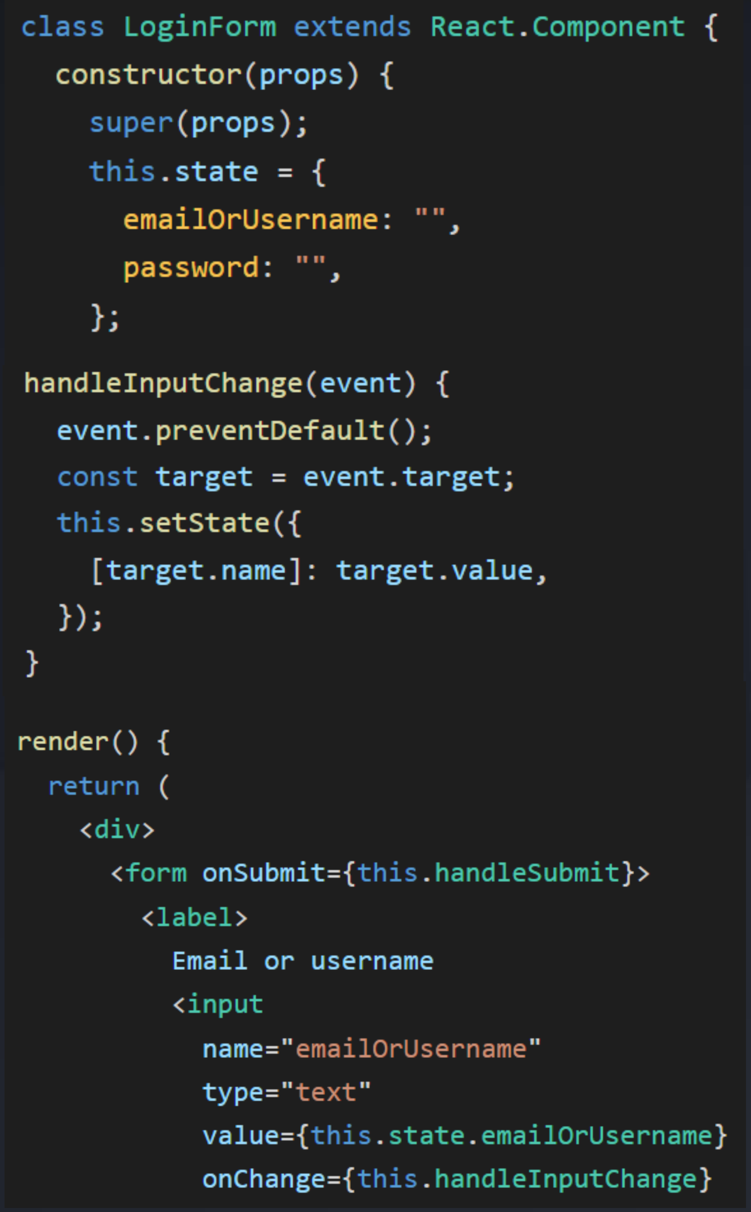
A word about class-based components

Current standard. Introduced in React v16.8



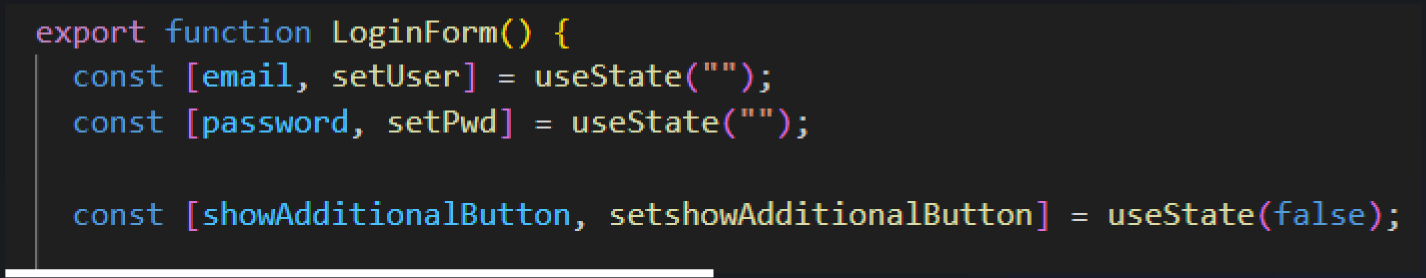
```
export function LoginForm() {
  const [email, setUser] = useState("");
  const [password, setPwd] = useState("");

  useEffect(() => {⋯
  }, [email]);
  const handleSubmit = async (e) => {⋯
  };

  const handleUserChange = (e) =>
  { setUser(e.target.value) };
  const handlePasswordChange = (e) =>
  { setPwd(e.target.value) };
return (
  <>
    <h1>SIGN IN FORM</h1>
    <form onSubmit={handleSubmit} className="form">
      <div className="controls">
        <div className="input">
          <label htmlFor="email">Email</label>
          <input
            placeholder="Email address"
            onChange={handleUserChange}
            value={email}
          />
        </div>
        <div className="input">
          <label htmlFor="image">Password</label>
          <div className="password-input-block">
            <input
```

VS

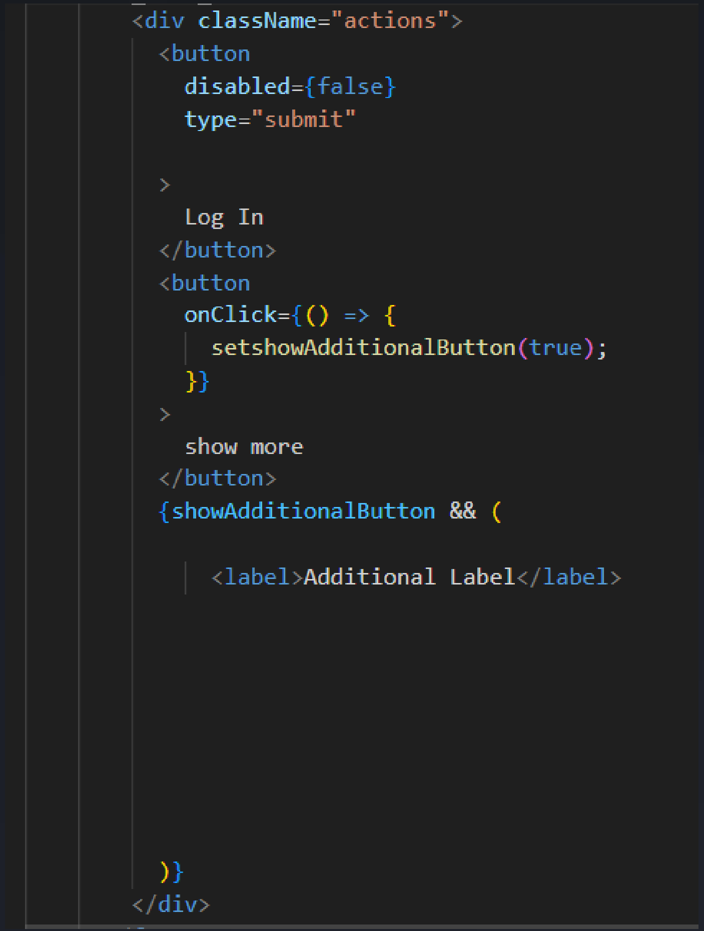


```
class LoginForm extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      emailOrUsername: "",
      password: "",
    };

handleInputChange(event) {
  event.preventDefault();
  const target = event.target;
  this.setState({
    [target.name]: target.value,
  });
}

render() {
  return (
    <div>
      <form onSubmit={this.handleSubmit}>
        <label>
          Email or username
          <input
            name="emailOrUsername"
            type="text"
            value={this.state.emailOrUsername}
            onChange={this.handleInputChange}
```

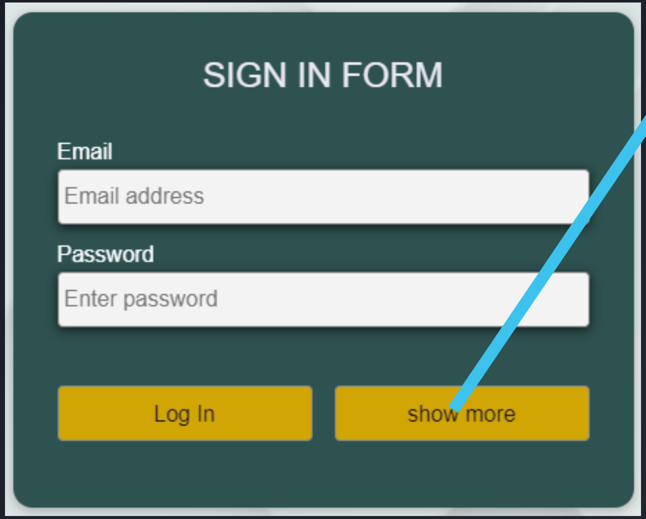
Add more state

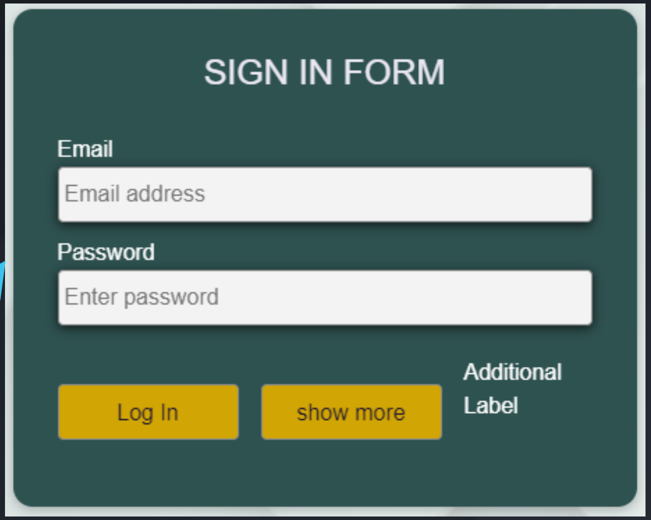


```
export function LoginForm() {
  const [email, setUser] = useState("");
  const [password, setPwd] = useState("");

  const [showAdditionalButton, setshowAdditionalButton] = useState(false);
```



```
<div className="actions">
  <button
    disabled={false}
    type="submit"

  >
    Log In
  </button>
  <button
    onClick={() => {
      setshowAdditionalButton(true);
    }}
  >
    show more
  </button>
  {showAdditionalButton && (

      <label>Additional Label</label>







  )}
</div>
```





Add more state

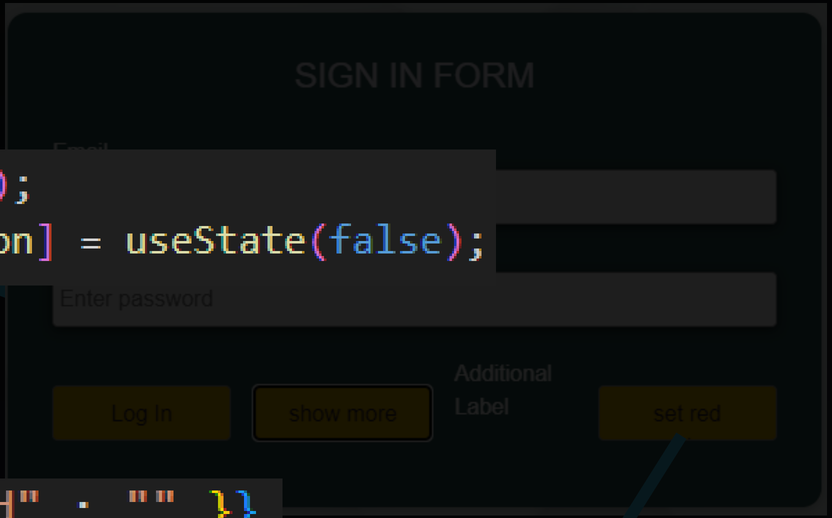
```
export function Log
  const [email, set
  const [password,

  const [showAdditionalButton, setshowAdditionalButton] = useState(false);
```

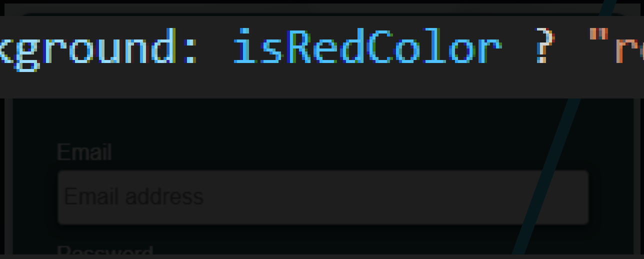
```
const [isRedColor, setIsRedColor] = useState(false);
const [showAdditionalButton, setshowAdditionalButton] = useState(false);
```

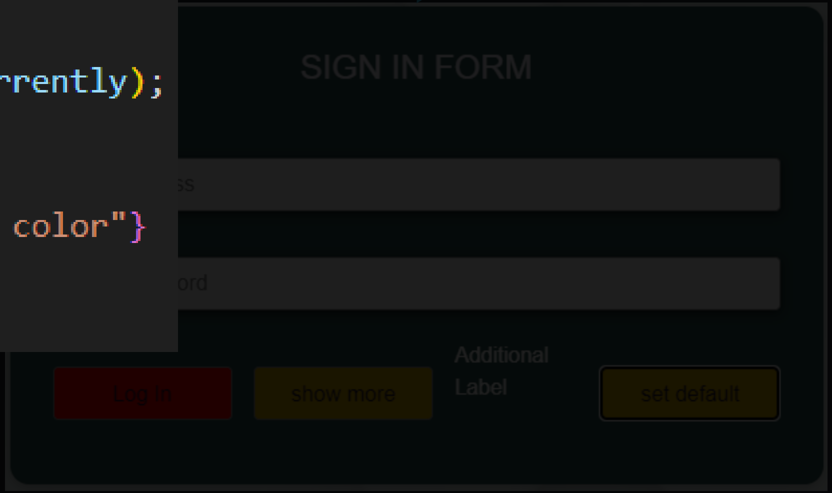
```
<div className="actions">
  <button
    disabled={false}
    type="submit"
  >
    Log In
  </button>
  <button
    onClick={() => {
      setshowAdditionalButton(true);
    }}
  >
    show more
  </button>
  {showAdditionalButton && (

      <label>Additional Label</label>

  )}
</div>
```

```
style={{ background: isRedColor ? "red" : "" }}
```

```
<>
  <label>Additional Label</label>
  <button
    onClick={() => {
      setIsRedColor((isRedCurrently) => !isRedCurrently);
    }}
  >
    {isRedColor ? "set default color" : "set red color"}
  </button>
</>
```



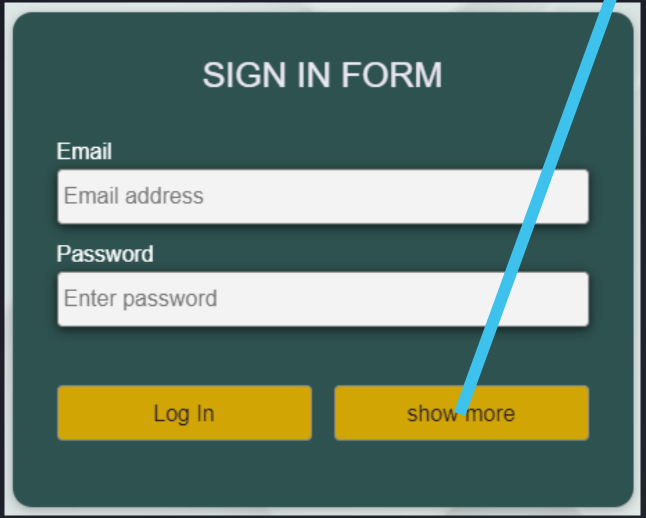


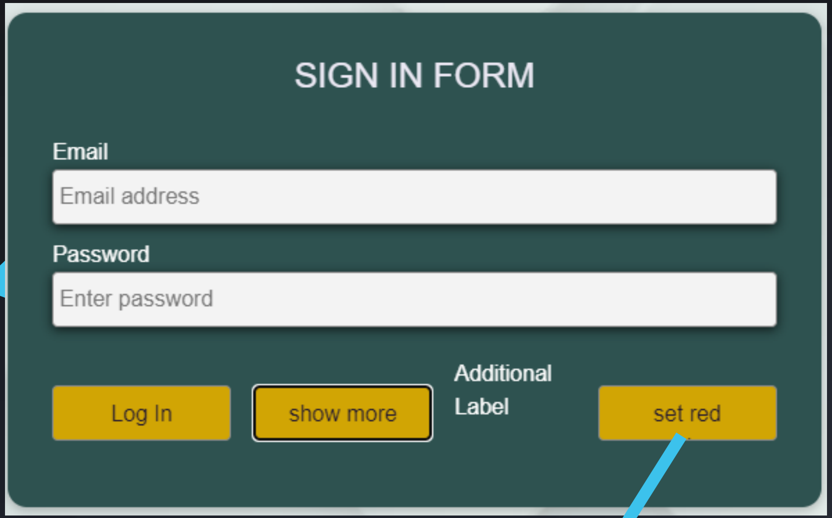


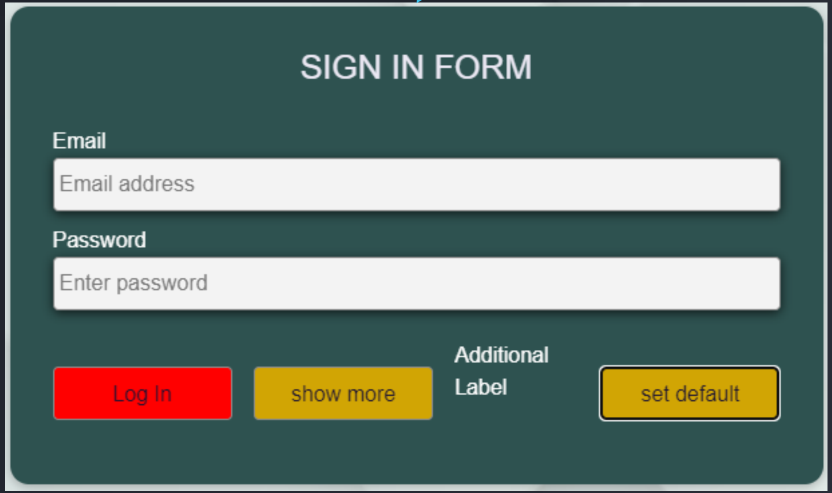
Add more state

```
export function LoginForm() {
  const [email, setUser] = useState("");
  const [password, setPwd] = useState("");
  const [isRedColor, setIsRedColor] = useState(false);
  const [showAdditionalButton, setshowAdditionalButton] = useState(false);
```

```
    <div className="actions">
      <button
        disabled={false}
        type="submit"
        style={{ background: isRedColor ? "red" : "" }}
      >
        Log In
      </button>
      <button
        onClick={() => {
          setshowAdditionalButton(true);
        }}
      >
        show more
      </button>
      {showAdditionalButton && (
        <>
          <label>Additional Label</label>
          <button
            onClick={() => {
              setIsRedColor((isRedCurrently) => !isRedCurrently);
            }}
          >
            {isRedColor ? "set default color" : "set red color"}
          </button>
        </>
      )}
    </div>
```







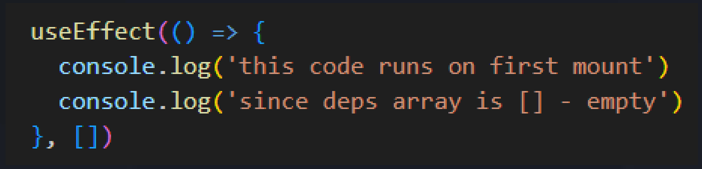
Component's life cycle. UseEffect()

Mounting Phase

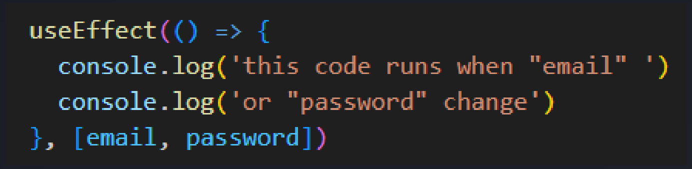
Updating Phase

Unmounting Phase

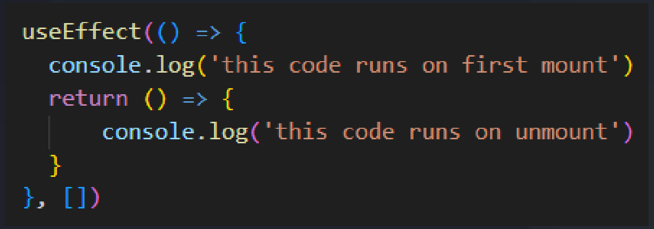
React hooks:



```
useEffect(() => {
  console.log('this code runs on first mount')
  console.log('since deps array is [] - empty')
}, [])
```



```
useEffect(() => {
  console.log('this code runs when "email" ')
  console.log('or "password" change')
}, [email, password])
```



```
useEffect(() => {
  console.log('this code runs on first mount')
  return () => {
      console.log('this code runs on unmount')
  }
}, [])
```

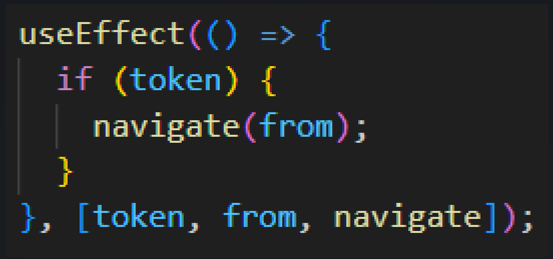
React class-based components:

Constructor()
getDerivedStateFromProps()
componentDidMount()

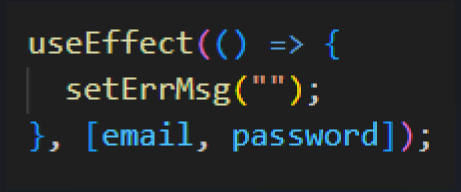
shouldComponentUpdate()
componentWillUpdate()
componentDidUpdate()

componentWillUnmount()

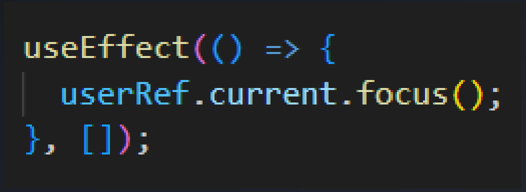
UseEffect() Usage examples



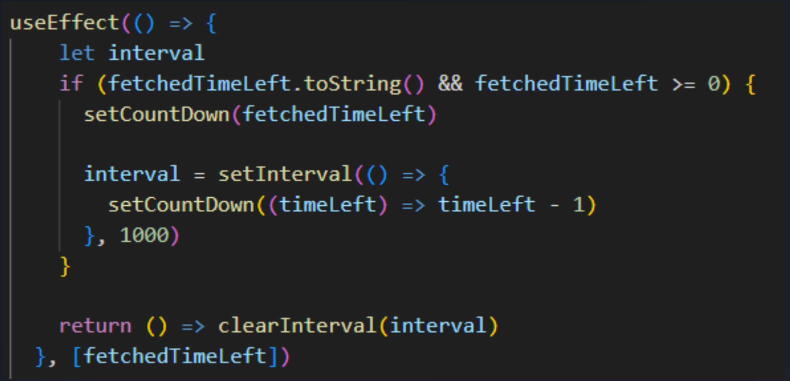
```
useEffect(() => {
  if (token) {
    navigate(from);
  }
}, [token, from, navigate]);
```



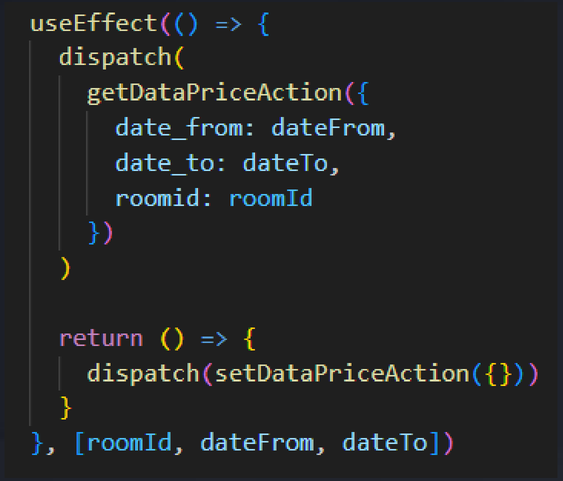
```
useEffect(() => {
  setErrMsg("");
}, [email, password]);
```



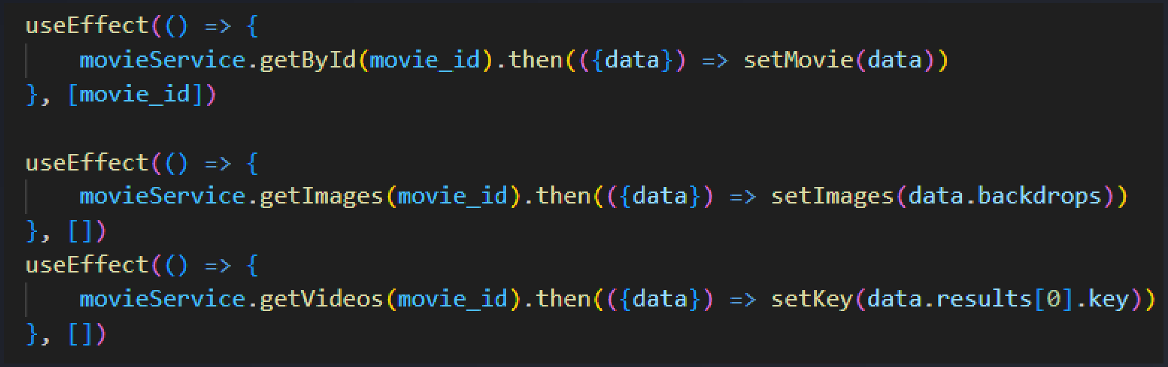
```
useEffect(() => {
  userRef.current.focus();
}, []);
```



```
useEffect(() => {
    let interval
    if (fetchedTimeLeft.toString() && fetchedTimeLeft >= 0) {
      setCountDown(fetchedTimeLeft)

      interval = setInterval(() => {
        setCountDown((timeLeft) => timeLeft - 1)
      }, 1000)
    }

    return () => clearInterval(interval)
  }, [fetchedTimeLeft])
```

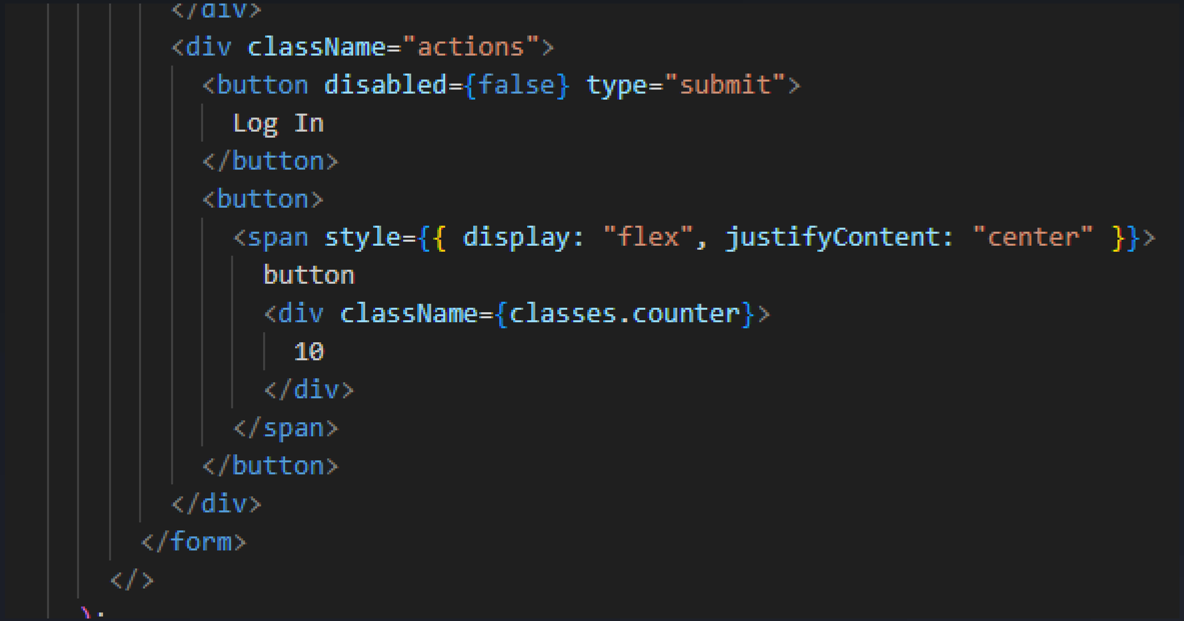


```
useEffect(() => {
  dispatch(
    getDataPriceAction({
      date_from: dateFrom,
      date_to: dateTo,
      roomid: roomId
    })
  )

  return () => {
    dispatch(setDataPriceAction({}))
  }
}, [roomId, dateFrom, dateTo])
```

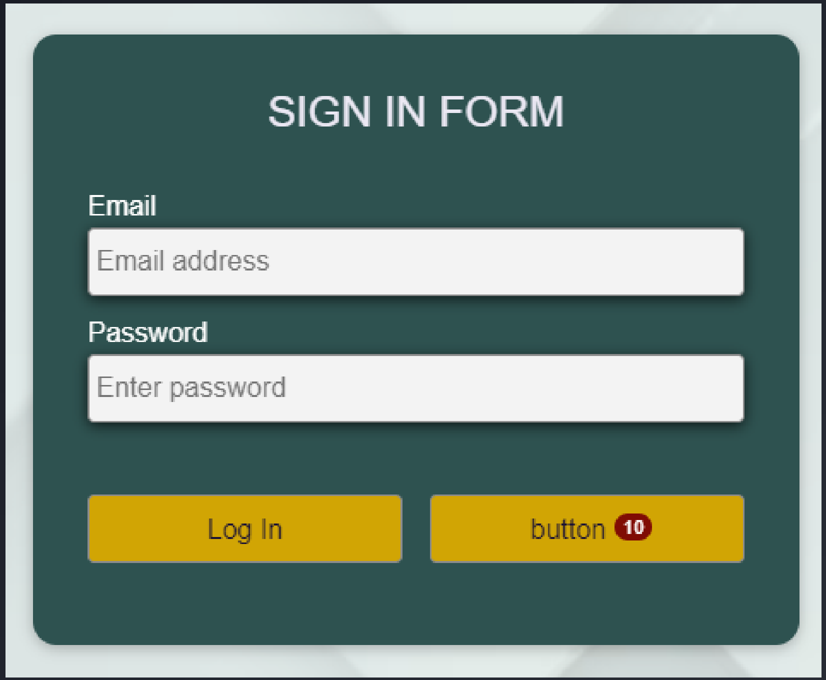


```
useEffect(() => {
    movieService.getById(movie_id).then(({data}) => setMovie(data))
}, [movie_id])

useEffect(() => {
    movieService.getImages(movie_id).then(({data}) => setImages(data.backdrops))
}, [])
useEffect(() => {
    movieService.getVideos(movie_id).then(({data}) => setKey(data.results[0].key))
}, [])
```

Props. Custom JSX elements



```
      </div>
      <div className="actions">
        <button disabled={false} type="submit">
          Log In
        </button>
        <button>
          <span style={{ display: "flex", justifyContent: "center" }}>
            button
            <div className={classes.counter}>
              10
            </div>
          </span>
        </button>
      </div>
    </form>
  </>
);
```



SIGN IN FORM

Email

Email address

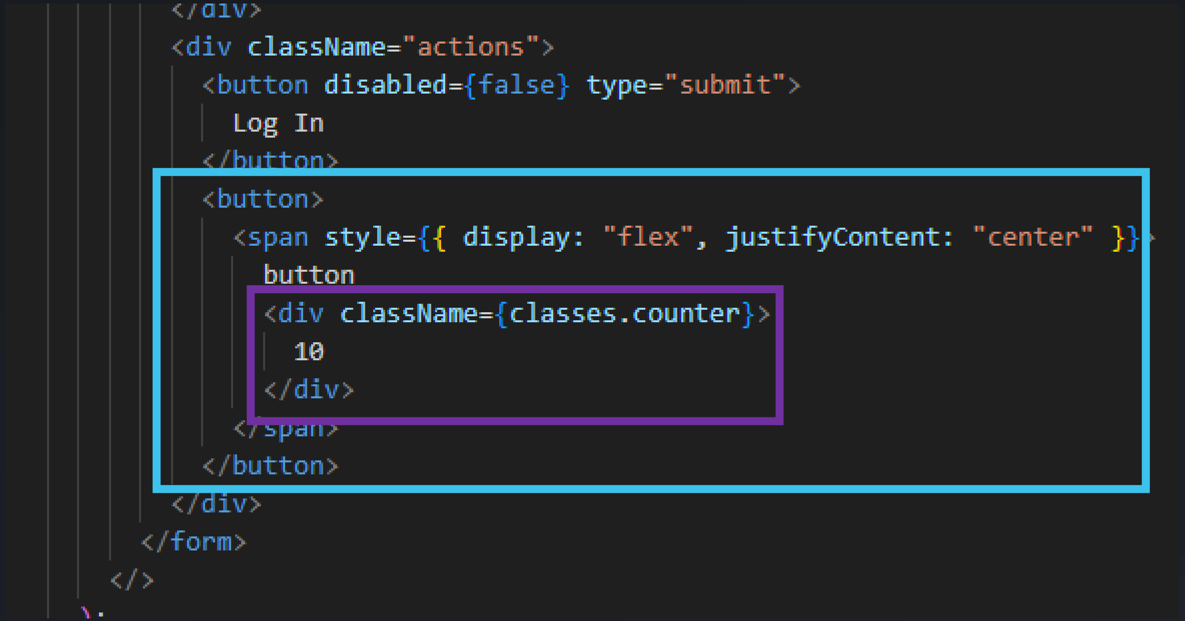
Password

Enter password

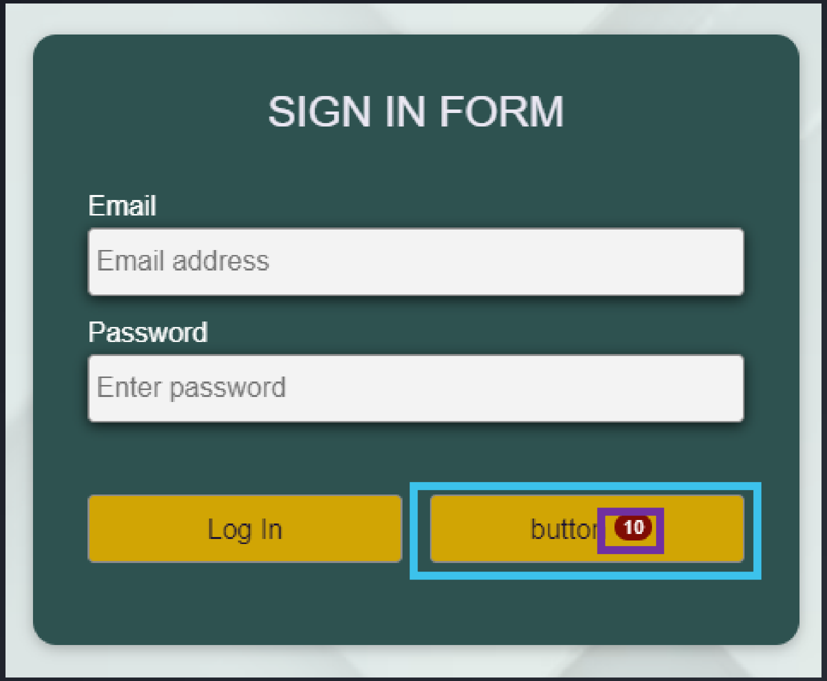
Log In

button 10

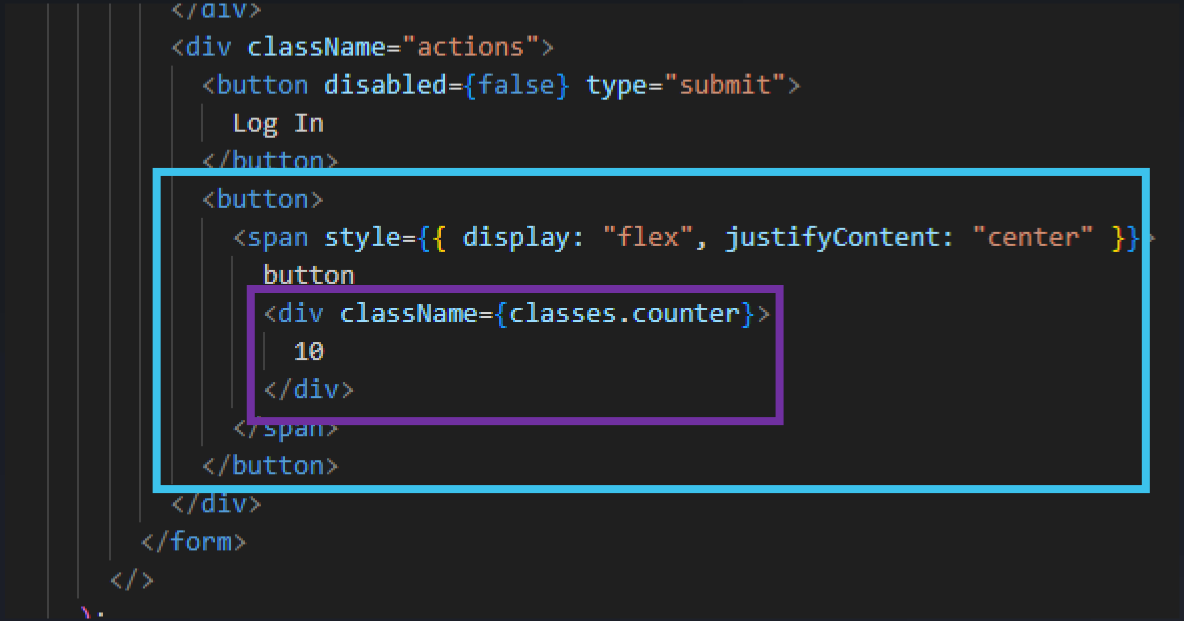
Props. Custom JSX elements



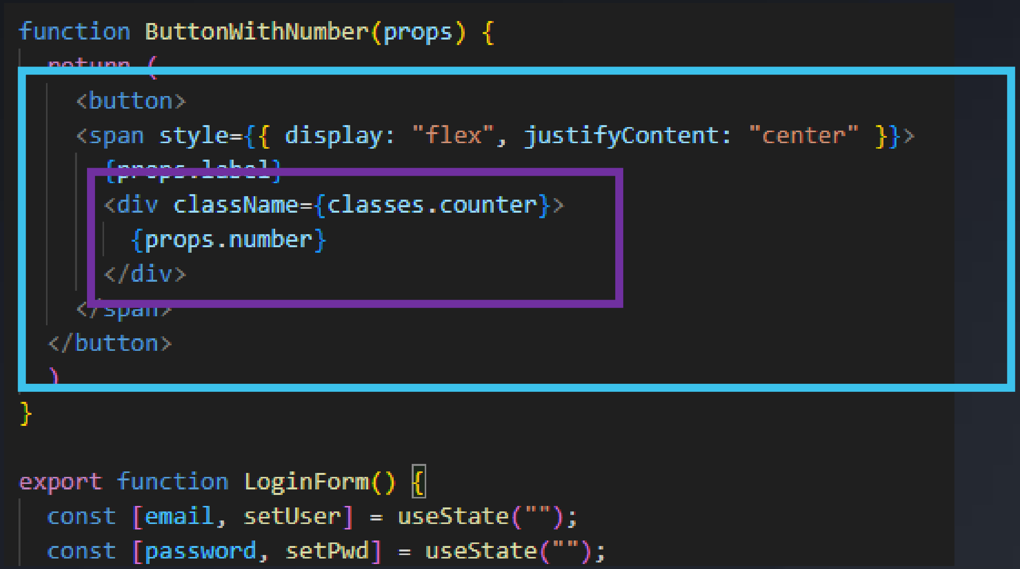
```
    </div>
    <div className="actions">
      <button disabled={false} type="submit">
        Log In
      </button>
      <button>
        <span style={{ display: "flex", justifyContent: "center" }}>
          button
          <div className={classes.counter}>
            10
          </div>
        </span>
      </button>
    </div>
  </form>
</>
);
```



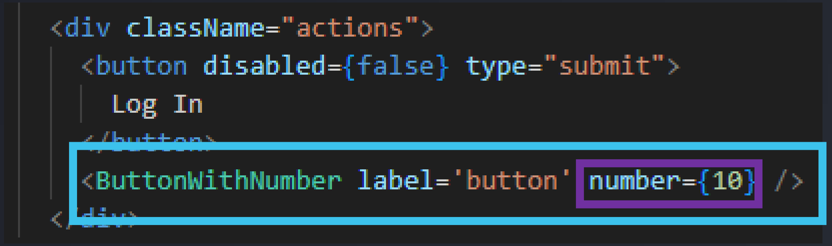
Props. Custom JSX elements



```
        </div>
        <div className="actions">
          <button disabled={false} type="submit">
            Log In
          </button>
          <button>
            <span style={{ display: "flex", justifyContent: "center" }}>
              button
              <div className={classes.counter}>
                10
              </div>
            </span>
          </button>
        </div>
      </form>
    </>
  );
```

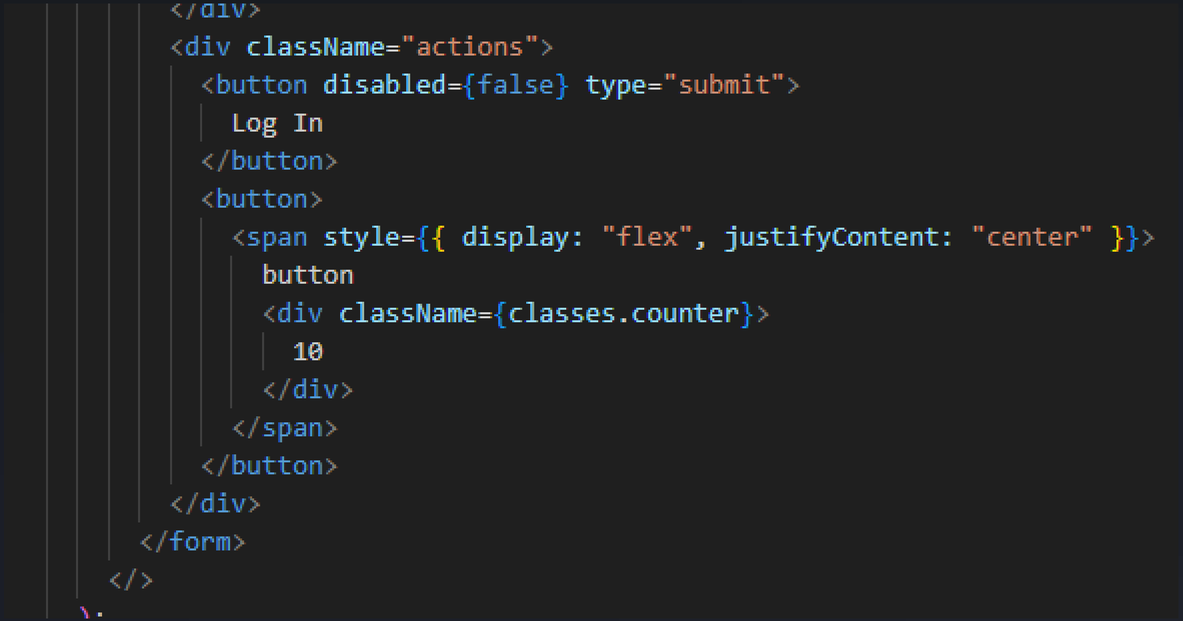


```
function ButtonWithNumber(props) {
  return (
    <button>
      <span style={{ display: "flex", justifyContent: "center" }}>
        {props.label}
        <div className={classes.counter}>
          {props.number}
        </div>
      </span>
    </button>
  )
}

export function LoginForm() {
  const [email, setUser] = useState("");
  const [password, setPwd] = useState("");
```

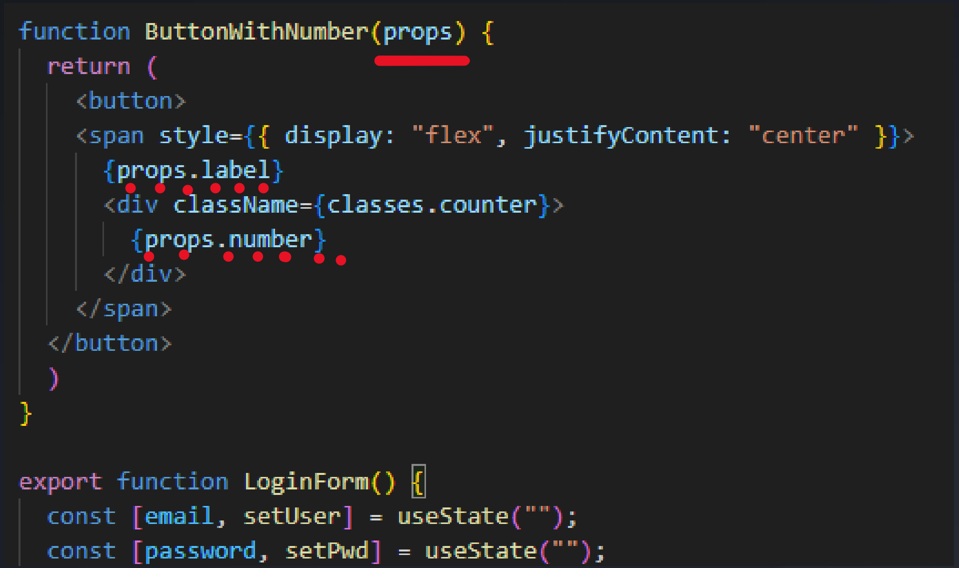


```
<div className="actions">
  <button disabled={false} type="submit">
    Log In
  </button>
  <ButtonWithNumber label='button' number={10} />
</div>
```

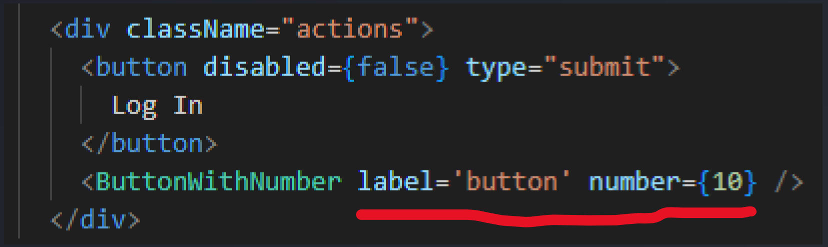
Props. Custom JSX elements



```
      </div>
      <div className="actions">
        <button disabled={false} type="submit">
          Log In
        </button>
        <button>
          <span style={{ display: "flex", justifyContent: "center" }}>
            button
            <div className={classes.counter}>
              10
            </div>
          </span>
        </button>
      </div>
    </form>
  </>
);
```

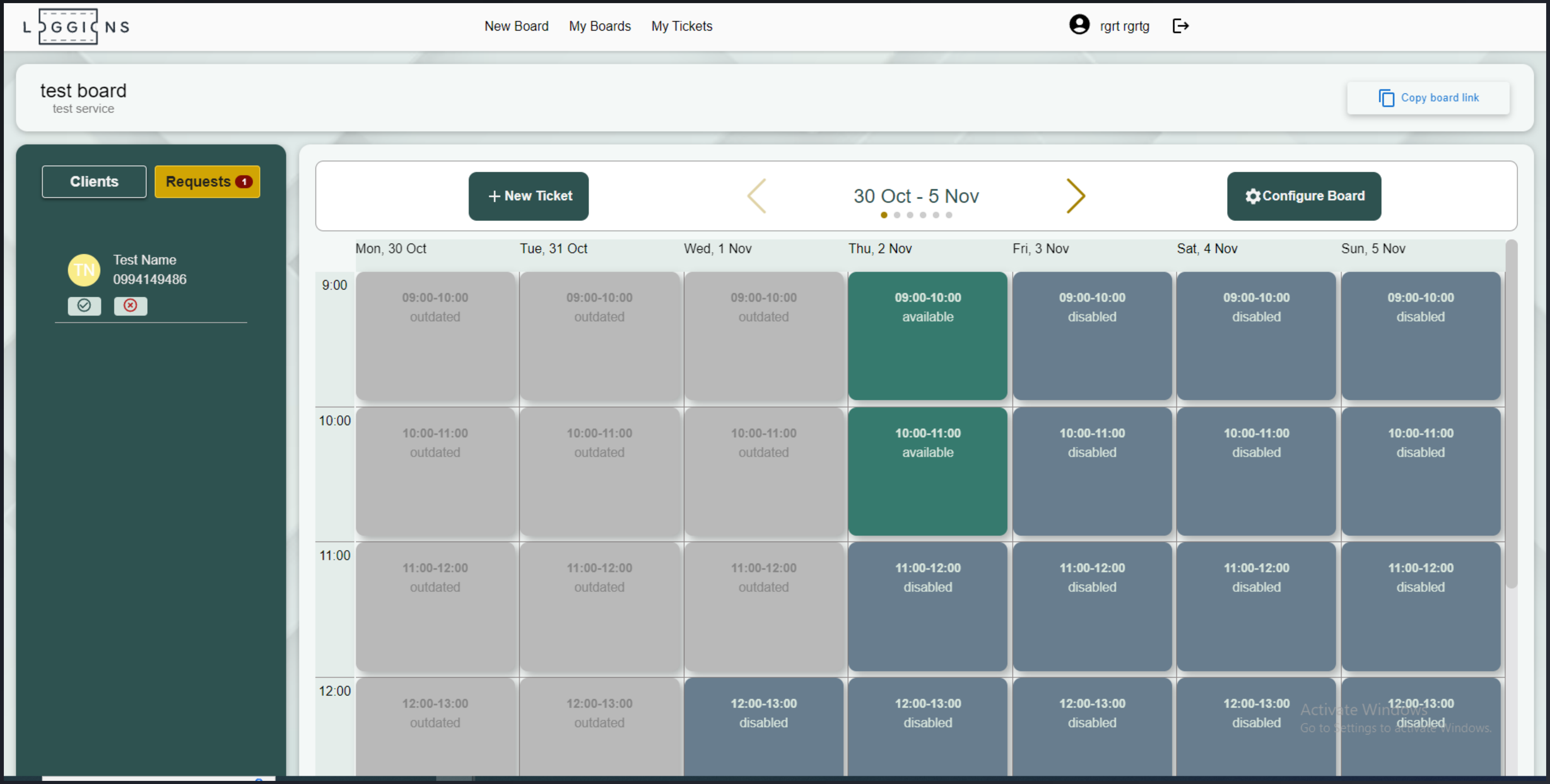


```
function ButtonWithNumber(props) {
  return (
    <button>
    <span style={{ display: "flex", justifyContent: "center" }}>
      {props.label}
      <div className={classes.counter}>
        {props.number}
      </div>
    </span>
  </button>
  )
}

export function LoginForm() {
  const [email, setUser] = useState("");
  const [password, setPwd] = useState("");
```



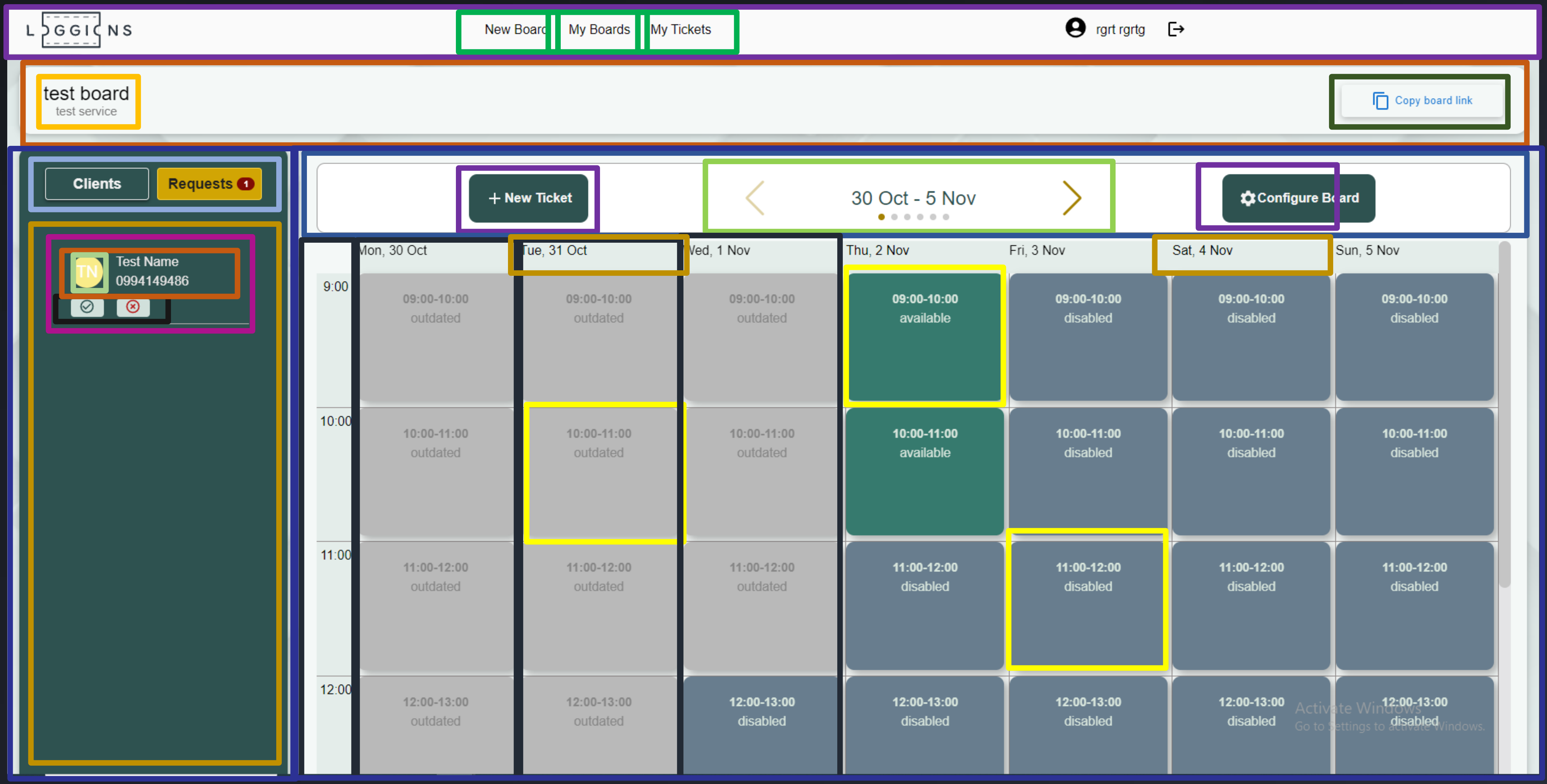
```
<div className="actions">
  <button disabled={false} type="submit">
    Log In
  </button>
  <ButtonWithNumber label='button' number={10} />
</div>
```

Custom JSX elements

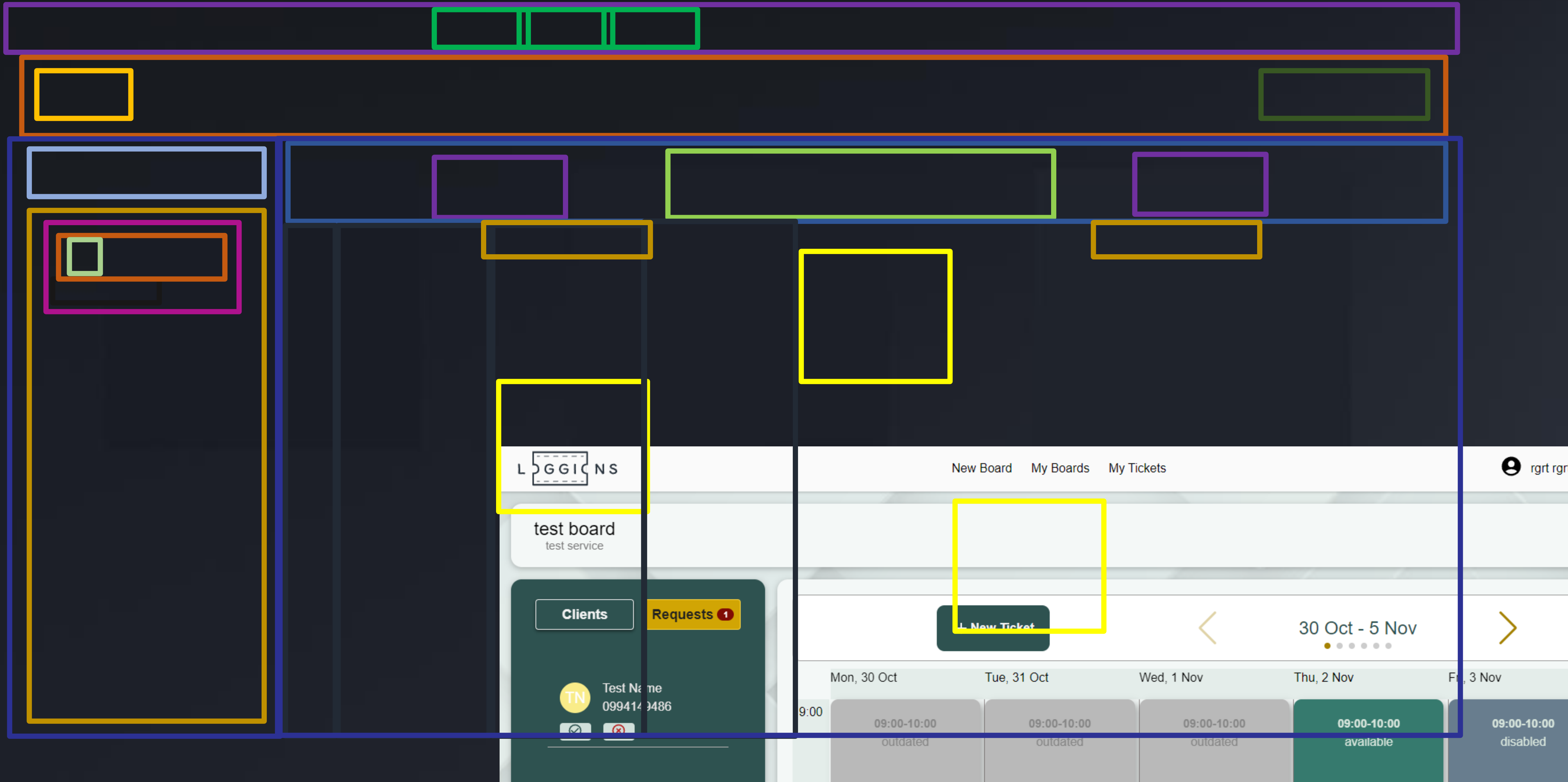


	Mon, 30 Oct	Tue, 31 Oct	Wed, 1 Nov	Thu, 2 Nov	Fri, 3 Nov	Sat, 4 Nov	Sun, 5 Nov
9:00	09:00-10:00 outdated	09:00-10:00 outdated	09:00-10:00 outdated	09:00-10:00 available	09:00-10:00 disabled	09:00-10:00 disabled	09:00-10:00 disabled
10:00	10:00-11:00 outdated	10:00-11:00 outdated	10:00-11:00 outdated	10:00-11:00 available	10:00-11:00 disabled	10:00-11:00 disabled	10:00-11:00 disabled
11:00	11:00-12:00 outdated	11:00-12:00 outdated	11:00-12:00 outdated	11:00-12:00 disabled	11:00-12:00 disabled	11:00-12:00 disabled	11:00-12:00 disabled
12:00	12:00-13:00 outdated	12:00-13:00 outdated	12:00-13:00 disabled	12:00-13:00 disabled	12:00-13:00 disabled	12:00-13:00 disabled	12:00-13:00 disabled

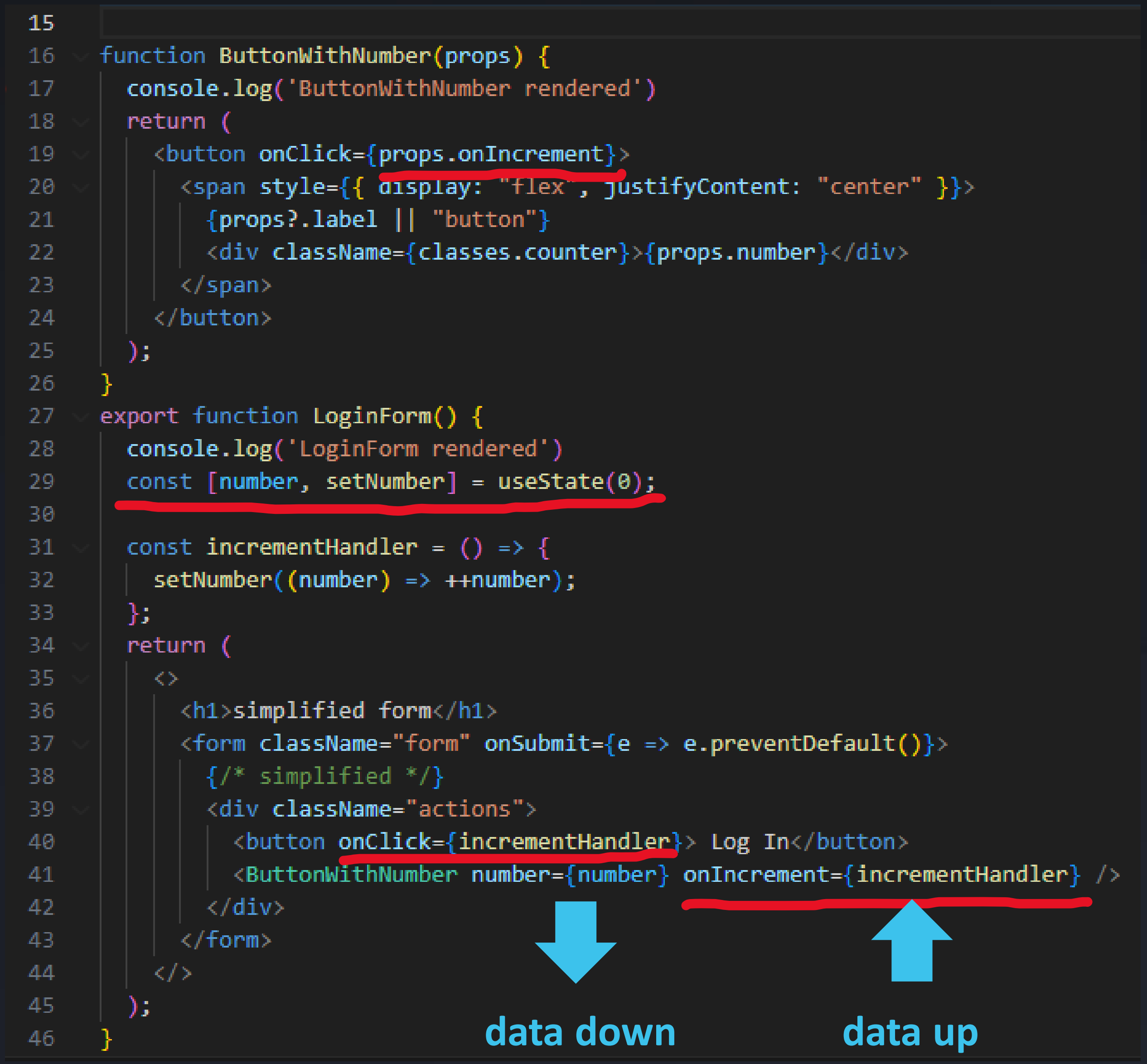
Custom JSX elements



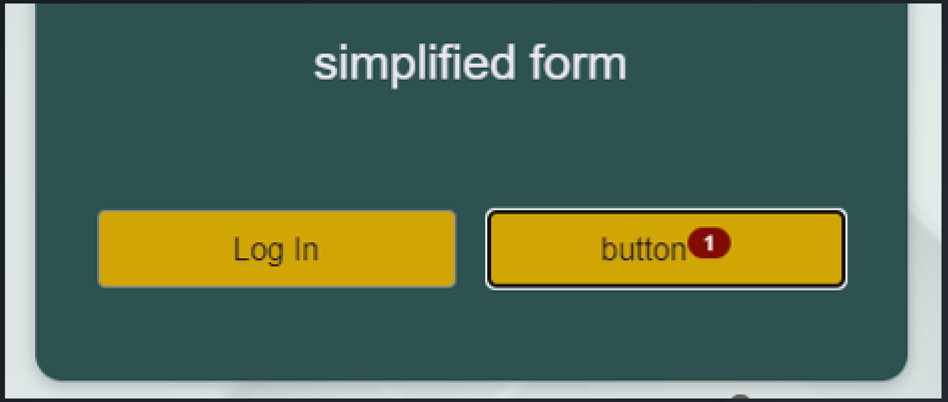
Custom JSX elements

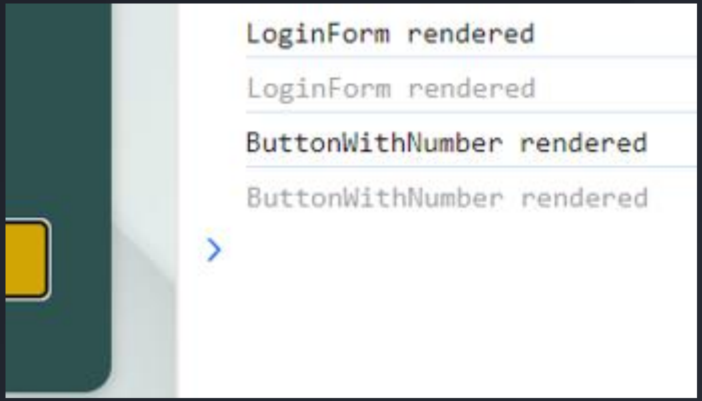


Props. Passing data up and down the tree



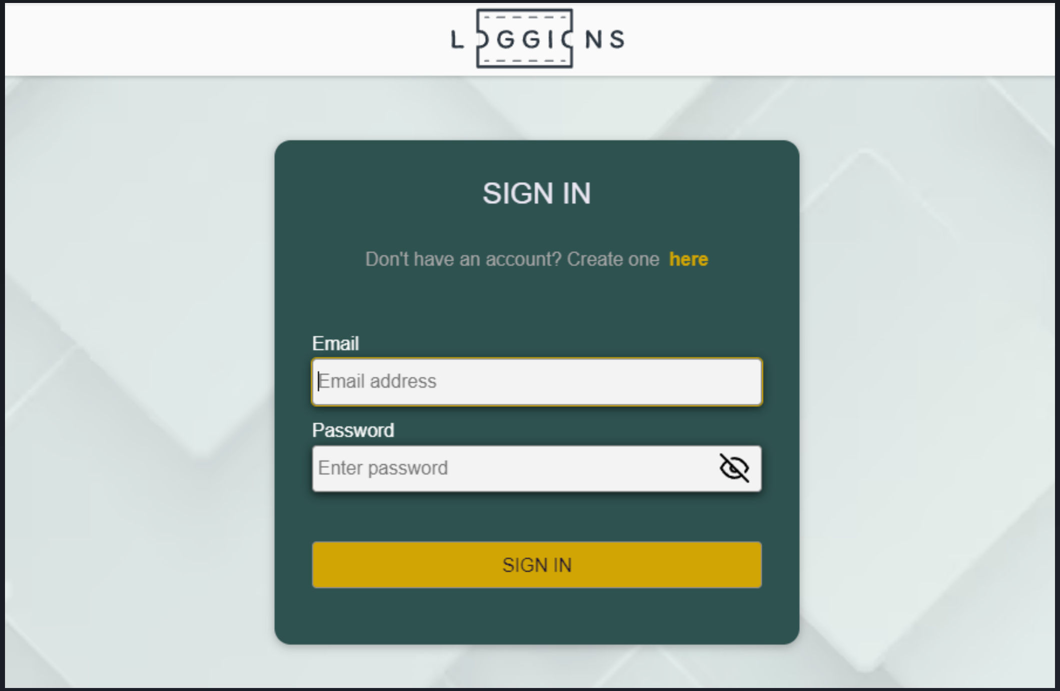
```
function ButtonWithNumber(props) {
  console.log('ButtonWithNumber rendered')
  return (
    <button onClick={props.onIncrement}>
      <span style={{ display: "flex", justifyContent: "center" }}>
        {props?.label || "button"}
        <div className={classes.counter}>{props.number}</div>
      </span>
    </button>
  );
}
export function LoginForm() {
  console.log('LoginForm rendered')
  const [number, setNumber] = useState(0);

  const incrementHandler = () => {
    setNumber((number) => ++number);
  };
  return (
    <>
      <h1>simplified form</h1>
      <form className="form" onSubmit={e => e.preventDefault()}>
        {/* simplified */}
        <div className="actions">
          <button onClick={incrementHandler}> Log In</button>
          <ButtonWithNumber number={number} onIncrement={incrementHandler} />
        </div>
      </form>
    </>
  );
}
```

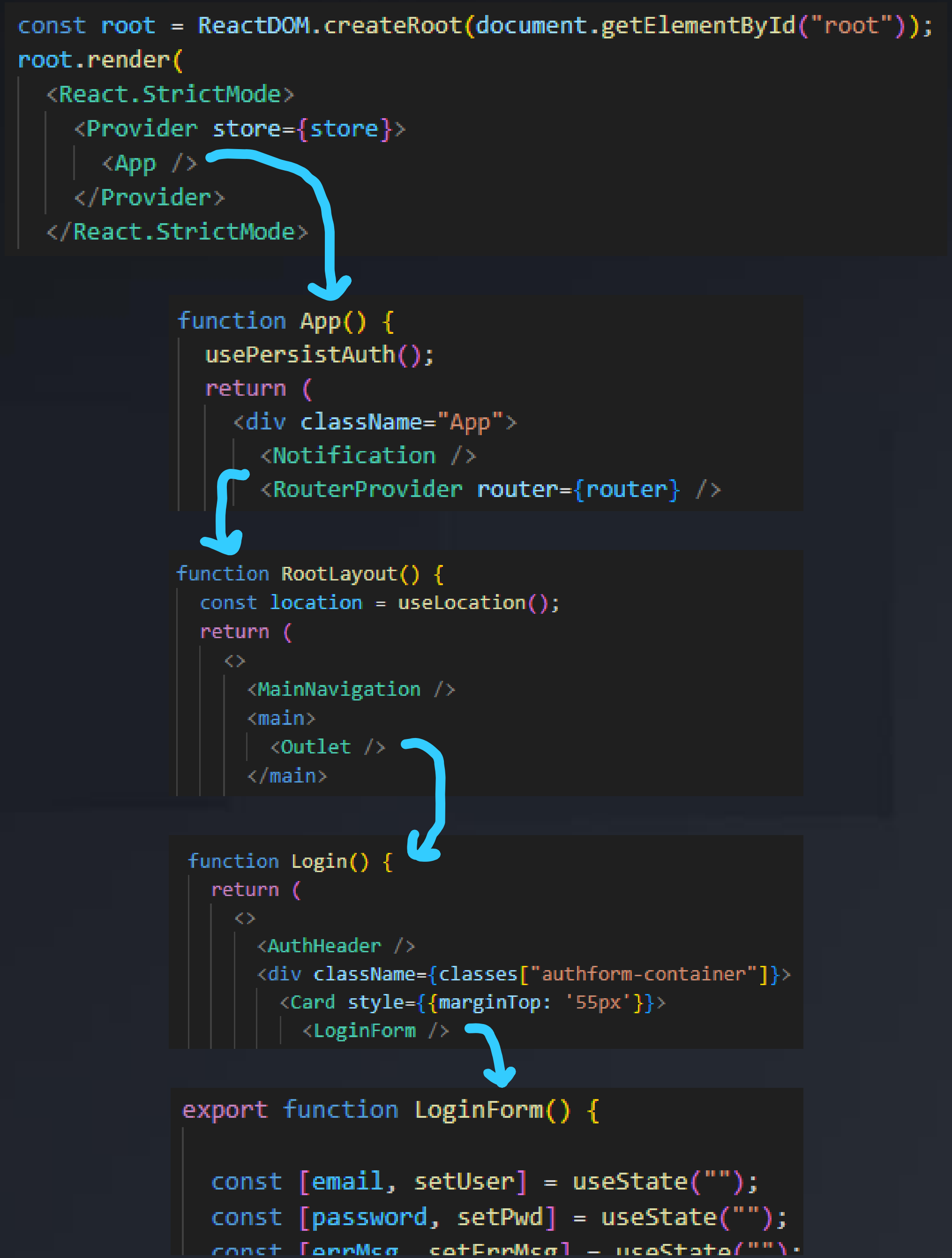




```
LoginForm rendered
LoginForm rendered
ButtonWithNumber rendered
ButtonWithNumber rendered
```

The tree. SPA





```
const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <React.StrictMode>
    <Provider store={store}>
      <App />
    </Provider>
  </React.StrictMode>
```

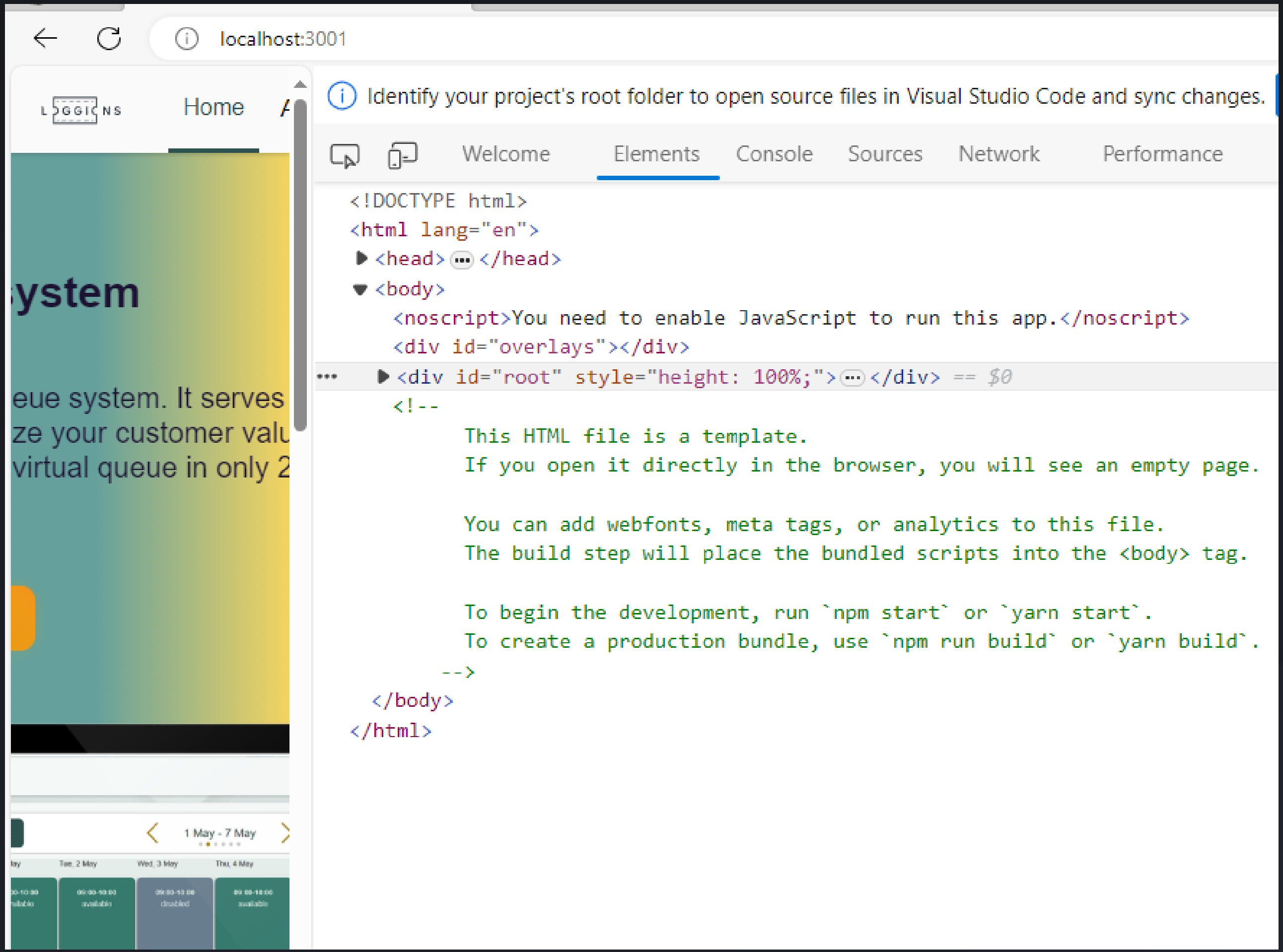
```
function App() {
  usePersistAuth();
  return (
    <div className="App">
      <Notification />
      <RouterProvider router={router} />
```

```
function RootLayout() {
  const location = useLocation();
  return (
    <>
      <MainNavigation />
      <main>
        <Outlet />
      </main>
```

```
function Login() {
  return (
    <>
      <AuthHeader />
      <div className={classes["authform-container"]}>
        <Card style={{marginTop: '55px'}}>
          <LoginForm />
```

```
export function LoginForm() {

  const [email, setUser] = useState("");
  const [password, setPwd] = useState("");
  const [errMsg, setErrMsg] = useState("");
```

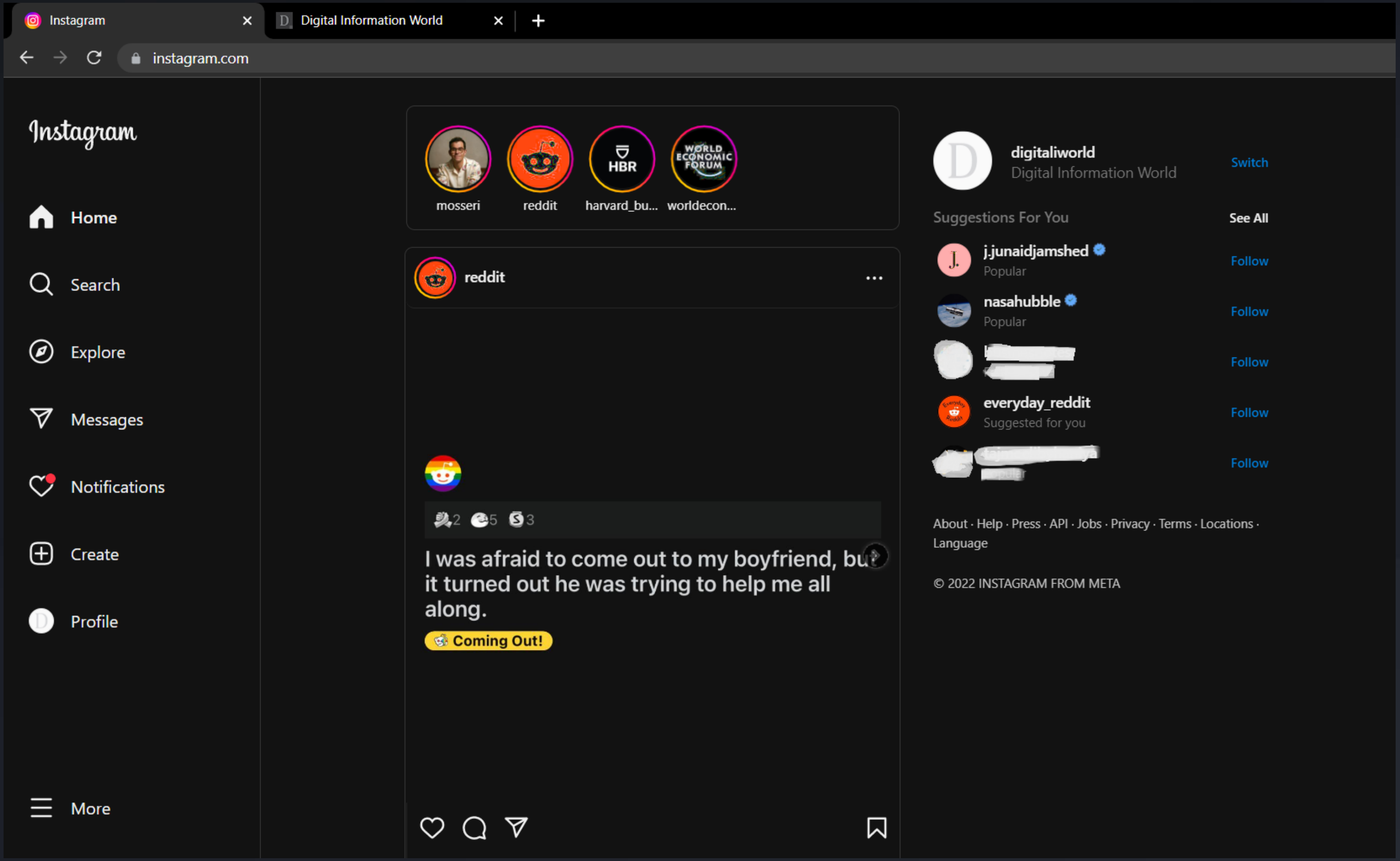
The tree. SPA



```
<!DOCTYPE html>
<html lang="en">
  <head> ••• </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="overlays"></div>
    <div id="root" style="height: 100%;"> ••• </div> == $0
    <!--
          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>
```

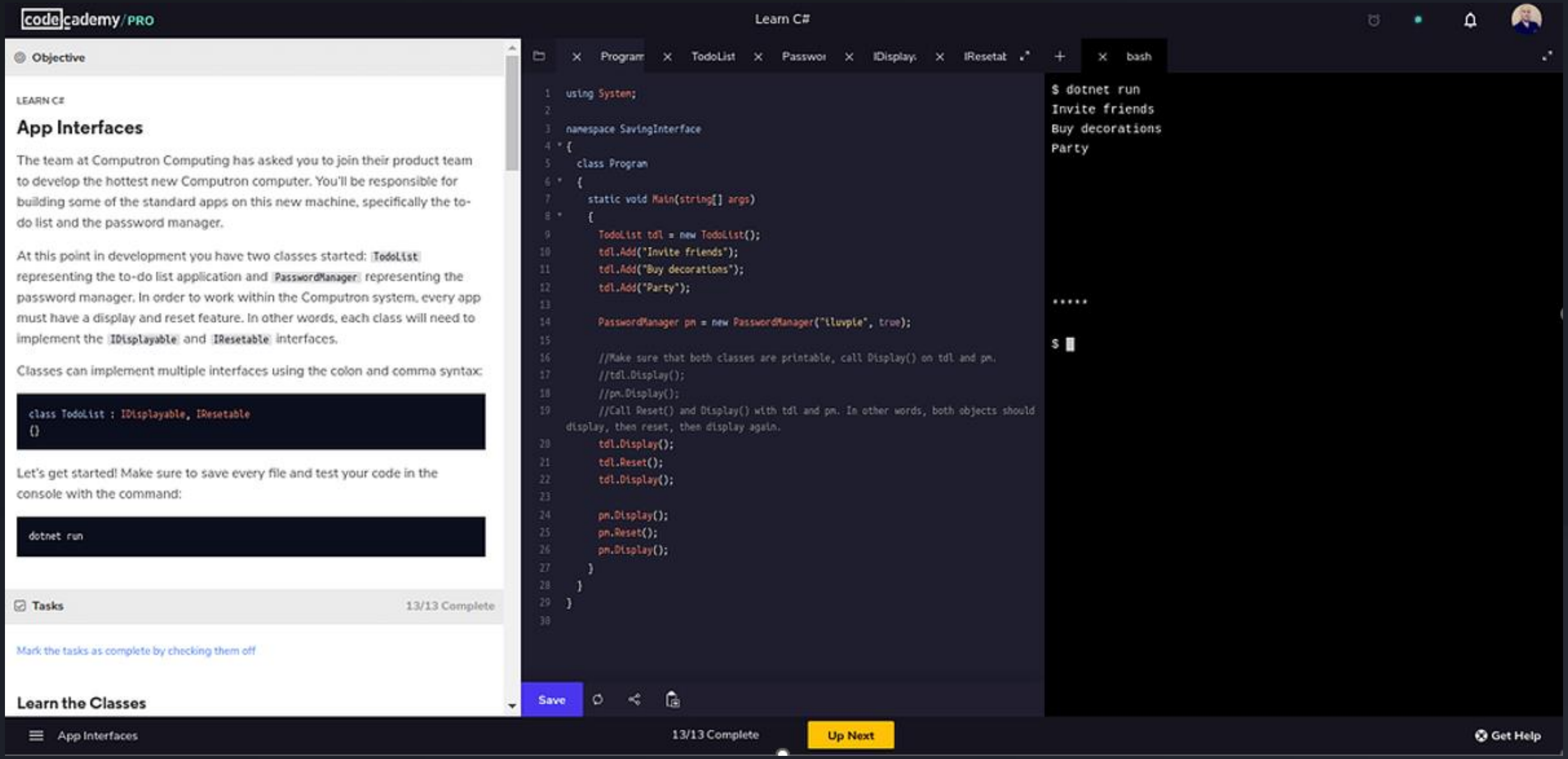
```
const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <React.StrictMode>
    <Provider store={store}>
      <App />
    </Provider>
  </React.StrictMode>
```

Uses ReactJS: Instagram



I was afraid to come out to my boyfriend, but it turned out he was trying to help me all along.

Uses ReactJS: Codecademy



LEARN C#

App Interfaces

The team at Computron Computing has asked you to join their product team to develop the hottest new Computron computer. You'll be responsible for building some of the standard apps on this new machine, specifically the to-do list and the password manager.

At this point in development you have two classes started: `TodoList` representing the to-do list application and `PasswordManager` representing the password manager. In order to work within the Computron system, every app must have a display and reset feature. In other words, each class will need to implement the `IDisplayable` and `IResetable` interfaces.

Classes can implement multiple interfaces using the colon and comma syntax:

```
class TodoList : IDisplayable, IResetable
{}
```

Let's get started! Make sure to save every file and test your code in the console with the command:

```
dotnet run
```

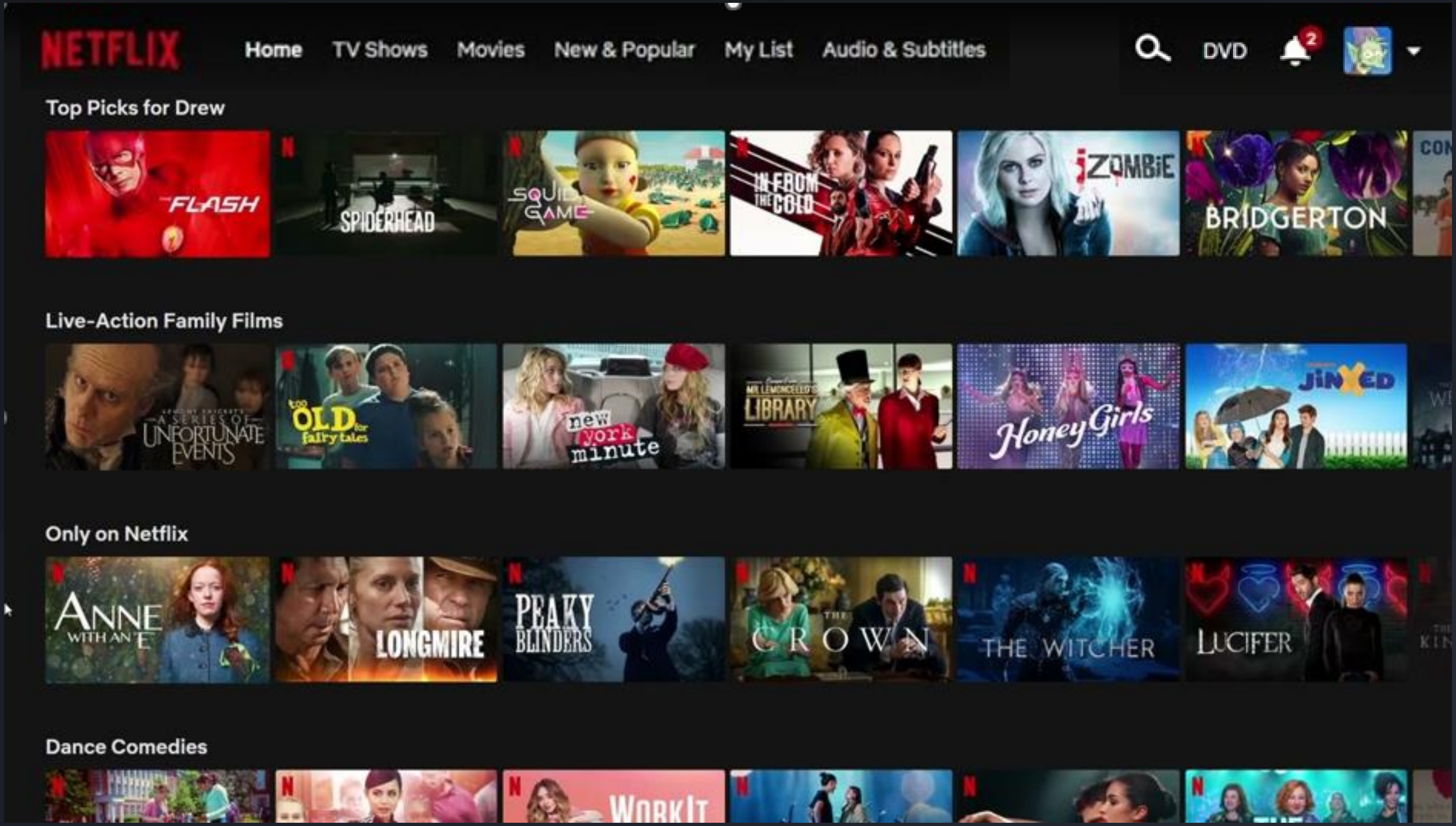
Mark the tasks as complete by checking them off

Learn the Classes

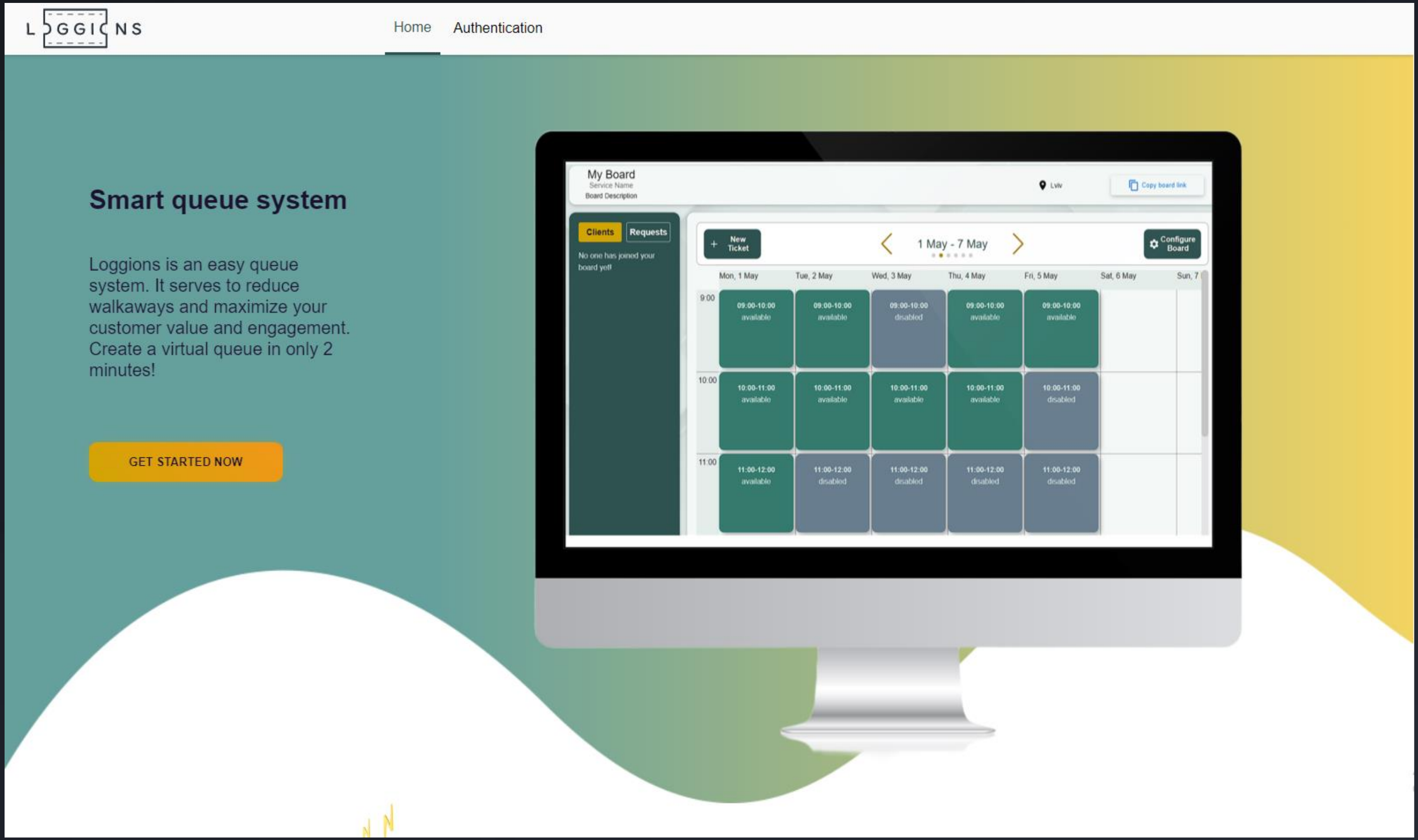
```
using System;

namespace SavingInterface
{
  class Program
  {
    static void Main(string[] args)
    {
      TodoList tdl = new TodoList();
      tdl.Add("Invite friends");
      tdl.Add("Buy decorations");
      tdl.Add("Party");

      PasswordManager pm = new PasswordManager("iluvpie", true);

      //Make sure that both classes are printable, call Display() on tdl and pm.
      //tdl.Display();
      //pm.Display();
      //Call Reset() and Display() with tdl and pm. In other words, both objects should display, then reset, then display again.
      tdl.Display();
      tdl.Reset();
      tdl.Display();

      pm.Display();
      pm.Reset();
      pm.Display();
    }
  }
}
```

```
$ dotnet run
Invite friends
Buy decorations
Party




*****

$ 
```

Uses ReactJS: Netflix



Uses ReactJS: Loggions



Home Authentication

Smart queue system

Loggions is an easy queue system. It serves to reduce walkaways and maximize your customer value and engagement. Create a virtual queue in only 2 minutes!

GET STARTED NOW

My Board
Service Name
Board Description

Lviv

Copy board link

Clients Requests

No one has joined your board yet!

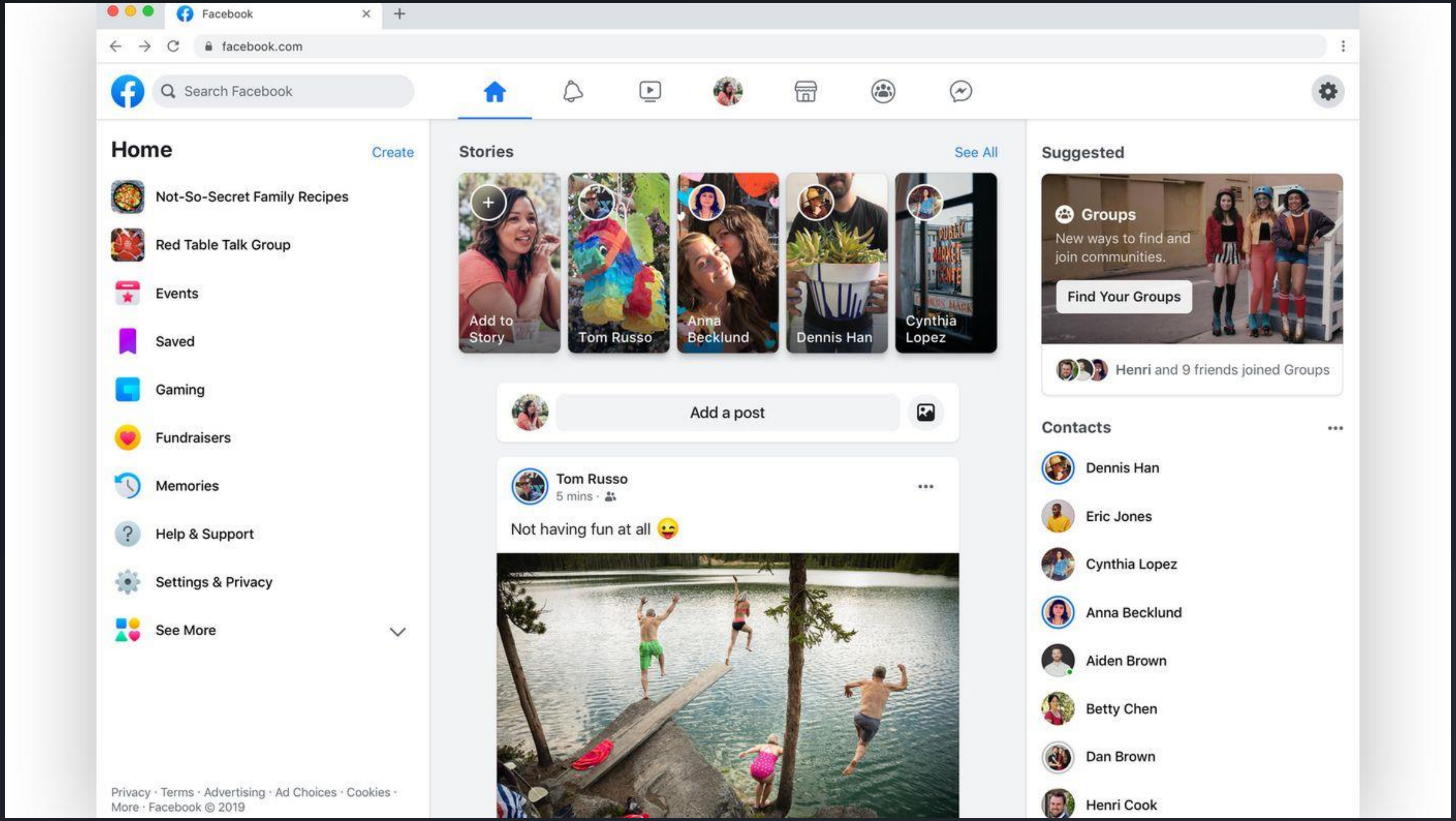
New Ticket

1 May - 7 May

Configure Board

	Mon, 1 May	Tue, 2 May	Wed, 3 May	Thu, 4 May	Fri, 5 May	Sat, 6 May	Sun, 7
9:00	09:00-10:00 available	09:00-10:00 available	09:00-10:00 disabled	09:00-10:00 available	09:00-10:00 available		
10:00	10:00-11:00 available	10:00-11:00 available	10:00-11:00 available	10:00-11:00 available	10:00-11:00 disabled		
11:00	11:00-12:00 available	11:00-12:00 disabled	11:00-12:00 disabled	11:00-12:00 disabled	11:00-12:00 disabled		

Uses ReactJS: Facebook



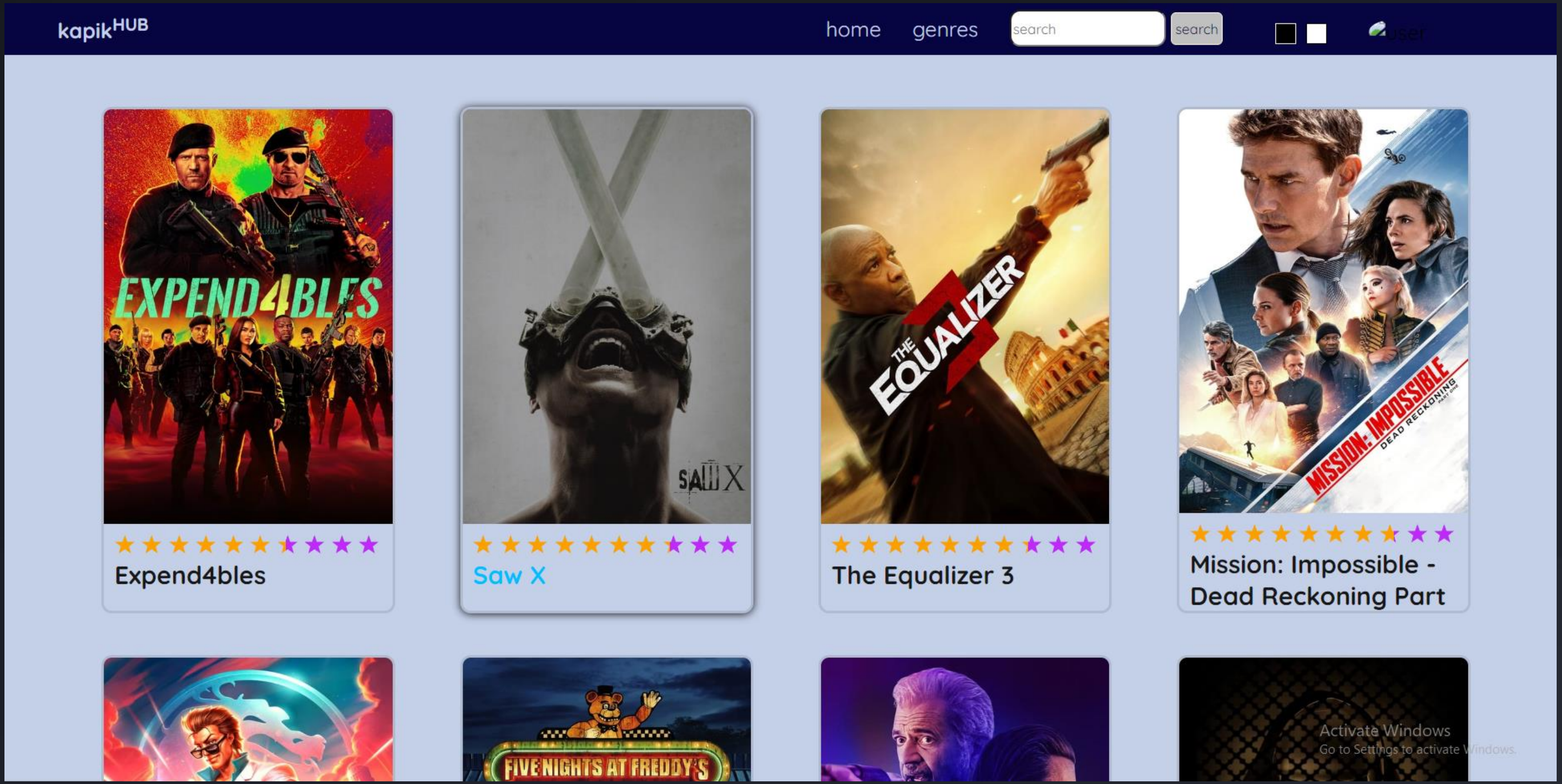
Tom Russo
5 mins ·

Not having fun at all 😜

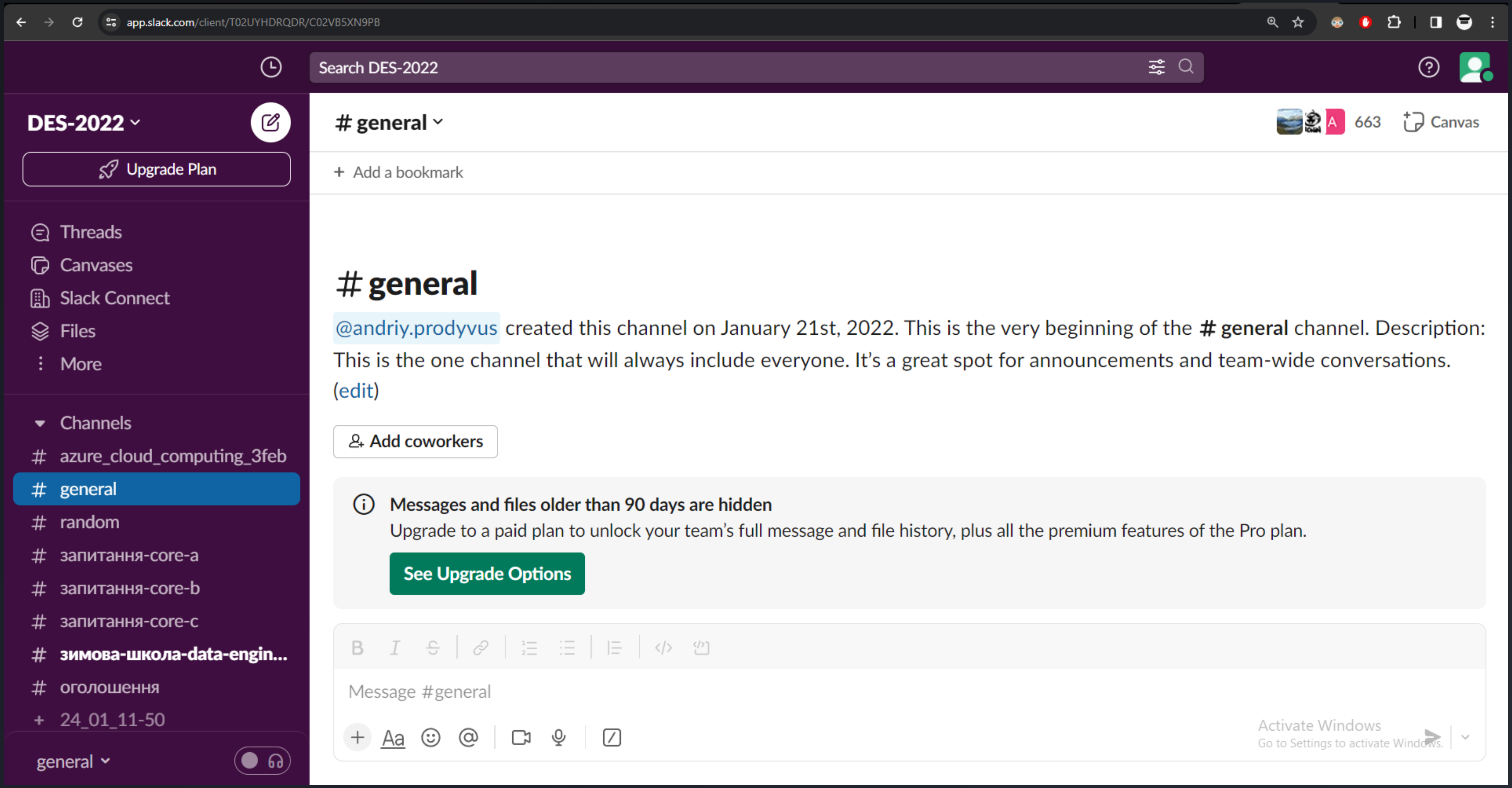
New ways to find and join communities.

Henri and 9 friends joined Groups

Uses ReactJS: Kapikhub



Uses ReactJS: Slack (browser version)



app.slack.com/client/T02UYHDRQDR/C02VB5XN9PB

Search DES-2022

DES-2022

Upgrade Plan

Threads
Canvases
Slack Connect
Files
More

Channels
# azure_cloud_computing_3feb
# general
# random
# запитання-core-a
# запитання-core-b
# запитання-core-c
# зимова-школа-data-engin...
# оголошення
+ 24_01_11-50

general

general

663 Canvas

+ Add a bookmark

general

@andriy.prodyvus created this channel on January 21st, 2022. This is the very beginning of the # general channel. Description: This is the one channel that will always include everyone. It’s a great spot for announcements and team-wide conversations. (edit)

Add coworkers

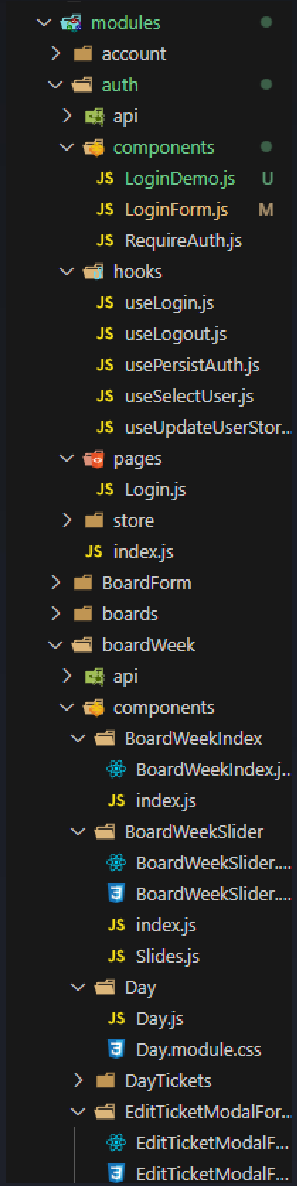
Messages and files older than 90 days are hidden
Upgrade to a paid plan to unlock your team’s full message and file history, plus all the premium features of the Pro plan.

See Upgrade Options

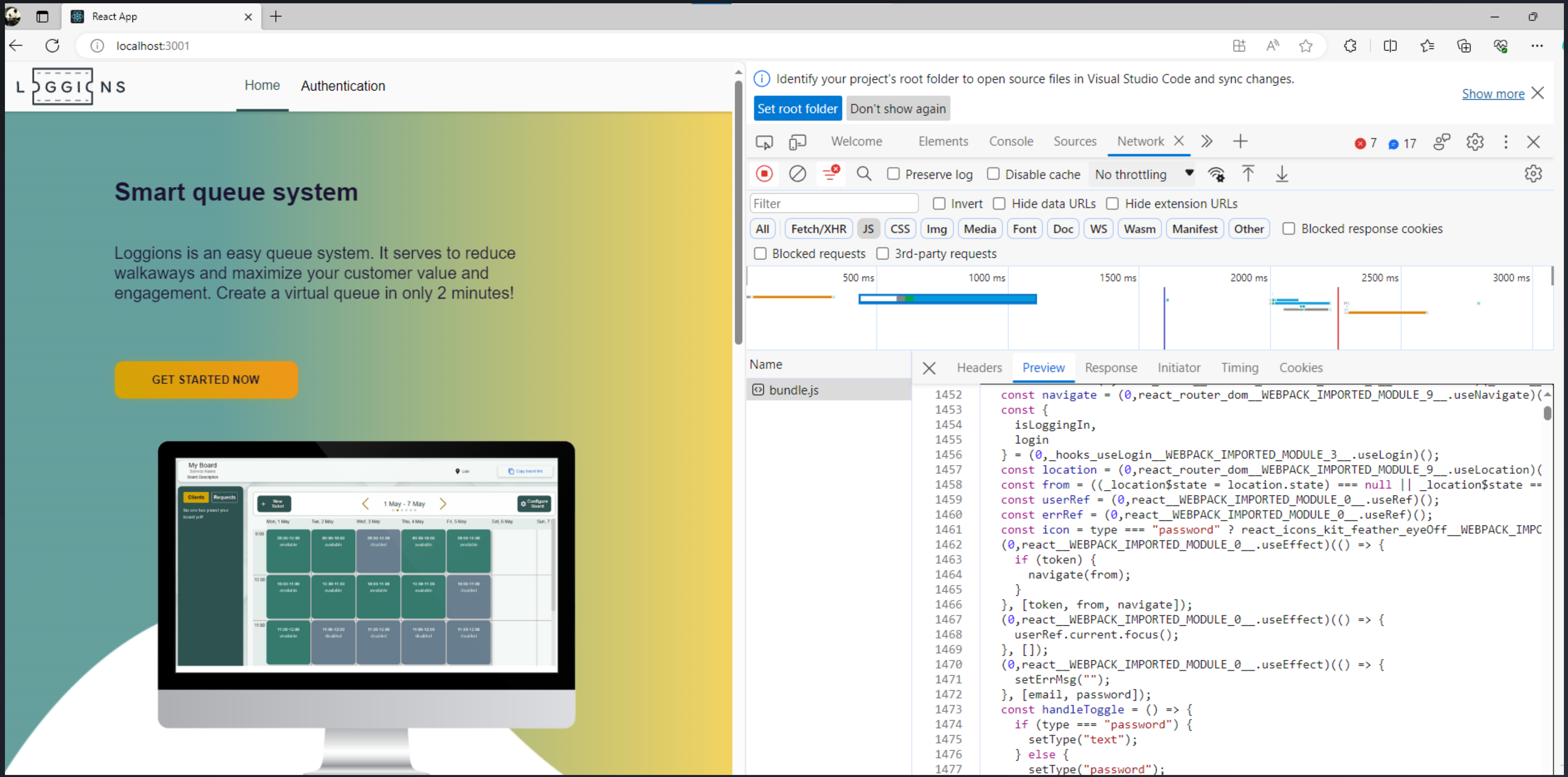
Message #general

Activate Windows
Go to Settings to activate Windows.

SPA and browsers



```
modules
  account
  auth
    api
    components
      LoginDemo.js        U
      LoginForm.js        M
      RequireAuth.js
    hooks
      useLogin.js
      useLogout.js
      usePersistAuth.js
      useSelectUser.js
      useUpdateUserStor...
    pages
      Login.js
    store
    index.js
  BoardForm
  boards
  boardWeek
    api
    components
      BoardWeekIndex
        BoardWeekIndex.j...
        index.js
      BoardWeekSlider
        BoardWeekSlider....
        BoardWeekSlider....
        index.js
        Slides.js
      Day
        Day.js
        Day.module.css
      DayTickets
      EditTicketModalFor...
        EditTicketModalF...
        EditTicketModalF...
```



React App

localhost:3001

L GGI NS Home Authentication

Smart queue system

Loggions is an easy queue system. It serves to reduce walkaways and maximize your customer value and engagement. Create a virtual queue in only 2 minutes!

GET STARTED NOW

Identify your project's root folder to open source files in Visual Studio Code and sync changes.

Set root folder Don't show again Show more

Welcome Elements Console Sources Network

Preserve log Disable cache No throttling

Filter Invert Hide data URLs Hide extension URLs

All Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other Blocked response cookies

Blocked requests 3rd-party requests

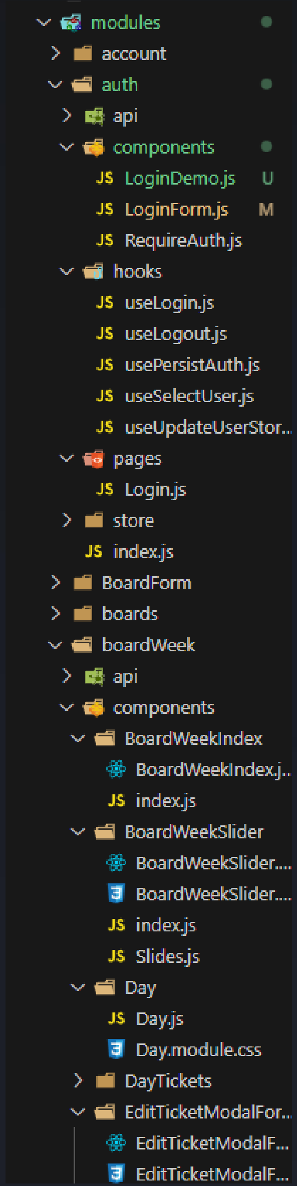
500 ms 1000 ms 1500 ms 2000 ms 2500 ms 3000 ms

Name bundle.js

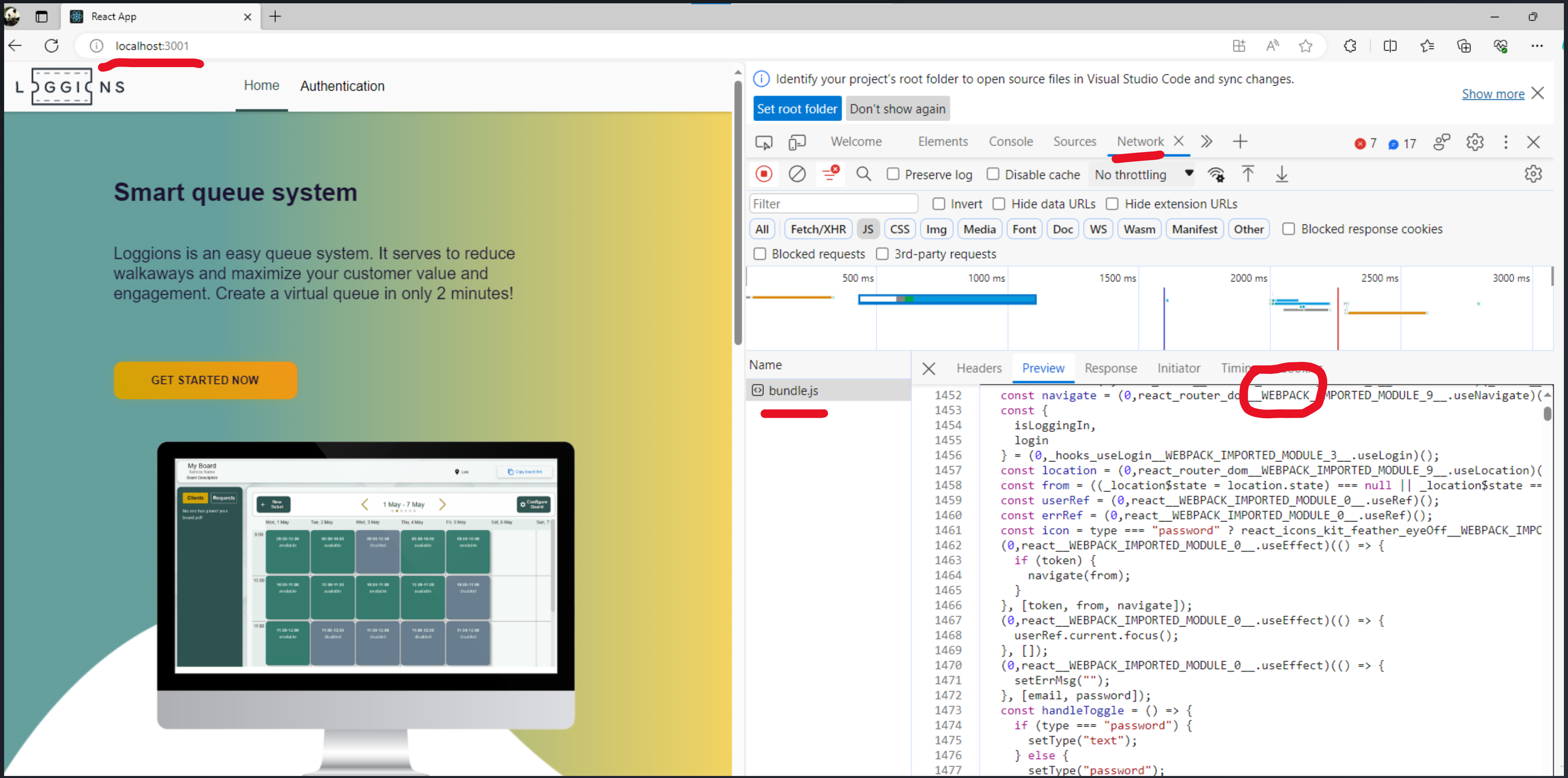
Headers Preview Response Initiator Timing Cookies

```
const navigate = (0,react_router_dom__WEBPACK_IMPORTED_MODULE_9__.useNavigate)(
const {
  isLoggingIn,
  login
} = (0,_hooks_useLogin__WEBPACK_IMPORTED_MODULE_3__.useLogin)();
const location = (0,react_router_dom__WEBPACK_IMPORTED_MODULE_9__.useLocation)(
const from = ((_location$state = location.state) === null || _location$state ==
const userRef = (0,react__WEBPACK_IMPORTED_MODULE_0__.useRef)();
const errRef = (0,react__WEBPACK_IMPORTED_MODULE_0__.useRef)();
const icon = type === "password" ? react_icons_kit_feather_eyeOff__WEBPACK_IMPC
(0,react__WEBPACK_IMPORTED_MODULE_0__.useEffect)(() => {
  if (token) {
    navigate(from);
  }
}, [token, from, navigate]);
(0,react__WEBPACK_IMPORTED_MODULE_0__.useEffect)(() => {
  userRef.current.focus();
}, []);
(0,react__WEBPACK_IMPORTED_MODULE_0__.useEffect)(() => {
  setErrMsg("");
}, [email, password]);
const handleToggle = () => {
  if (type === "password") {
    setType("text");
  } else {
    setType("password");
```

SPA and browsers



```
modules
  account
  auth
    api
    components
      LoginDemo.js      U
      LoginForm.js      M
      RequireAuth.js
    hooks
      useLogin.js
      useLogout.js
      usePersistAuth.js
      useSelectUser.js
      useUpdateUserStor...
    pages
      Login.js
    store
    index.js
  BoardForm
  boards
  boardWeek
    api
    components
      BoardWeekIndex
        BoardWeekIndex.j...
        index.js
      BoardWeekSlider
        BoardWeekSlider....
        BoardWeekSlider....
        index.js
        Slides.js
      Day
        Day.js
        Day.module.css
      DayTickets
      EditTicketModalFor...
        EditTicketModalF...
        EditTicketModalF...
```



Smart queue system

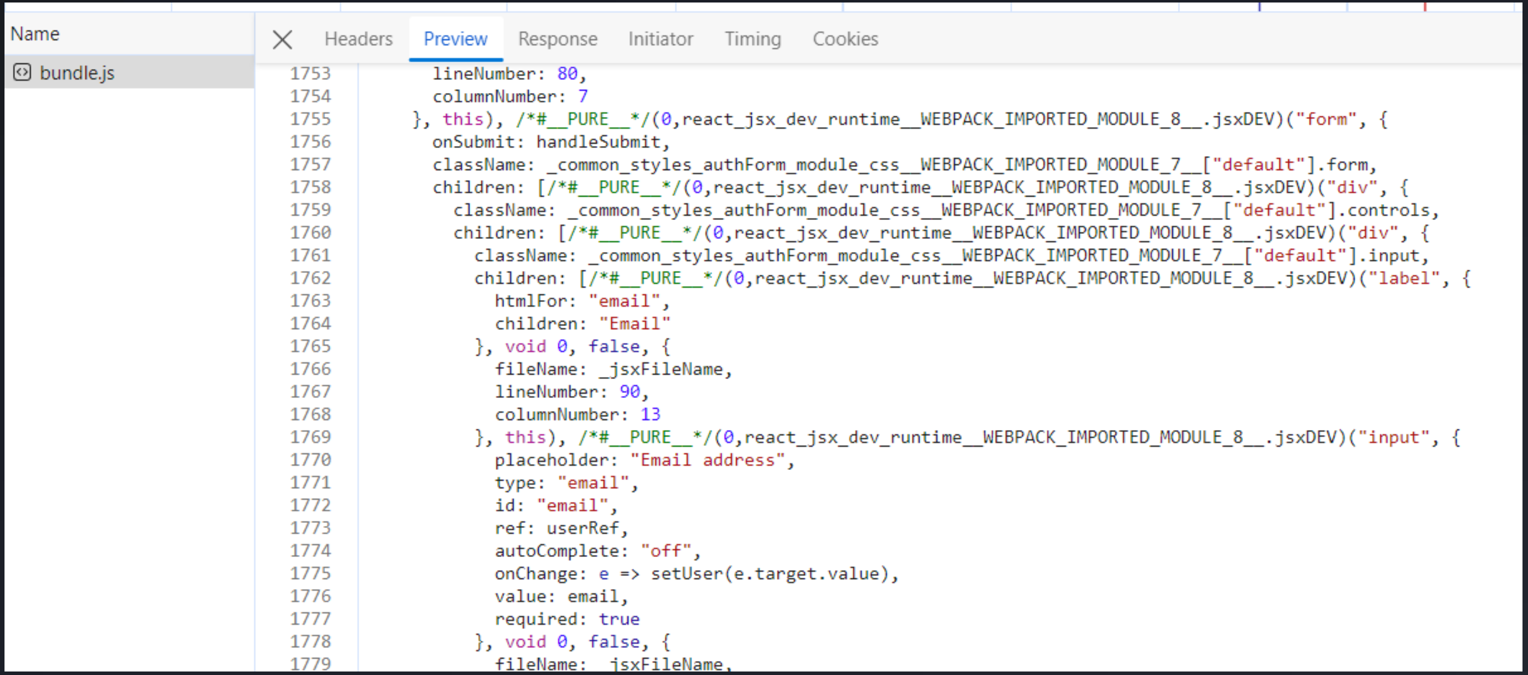
Loggions is an easy queue system. It serves to reduce walkaways and maximize your customer value and engagement. Create a virtual queue in only 2 minutes!

Identify your project's root folder to open source files in Visual Studio Code and sync changes.

```
const navigate = (0,react_router_dom__WEBPACK_IMPORTED_MODULE_9__.useNavigate)(
const {
  isLoggingIn,
  login
} = (0,_hooks_useLogin__WEBPACK_IMPORTED_MODULE_3__.useLogin)();
const location = (0,react_router_dom__WEBPACK_IMPORTED_MODULE_9__.useLocation)(
const from = ((_location$state = location.state) === null || _location$state ==
const userRef = (0,react__WEBPACK_IMPORTED_MODULE_0__.useRef)();
const errRef = (0,react__WEBPACK_IMPORTED_MODULE_0__.useRef)();
const icon = type === "password" ? react_icons_kit_feather_eyeOff__WEBPACK_IMPC
(0,react__WEBPACK_IMPORTED_MODULE_0__.useEffect)(() => {
  if (token) {
    navigate(from);
  }
}, [token, from, navigate]);
(0,react__WEBPACK_IMPORTED_MODULE_0__.useEffect)(() => {
  userRef.current.focus();
}, []);
(0,react__WEBPACK_IMPORTED_MODULE_0__.useEffect)(() => {
  setErrMsg("");
}, [email, password]);
const handleToggle = () => {
  if (type === "password") {
    setType("text");
  } else {
    setType("password");
```

Virtual DOM

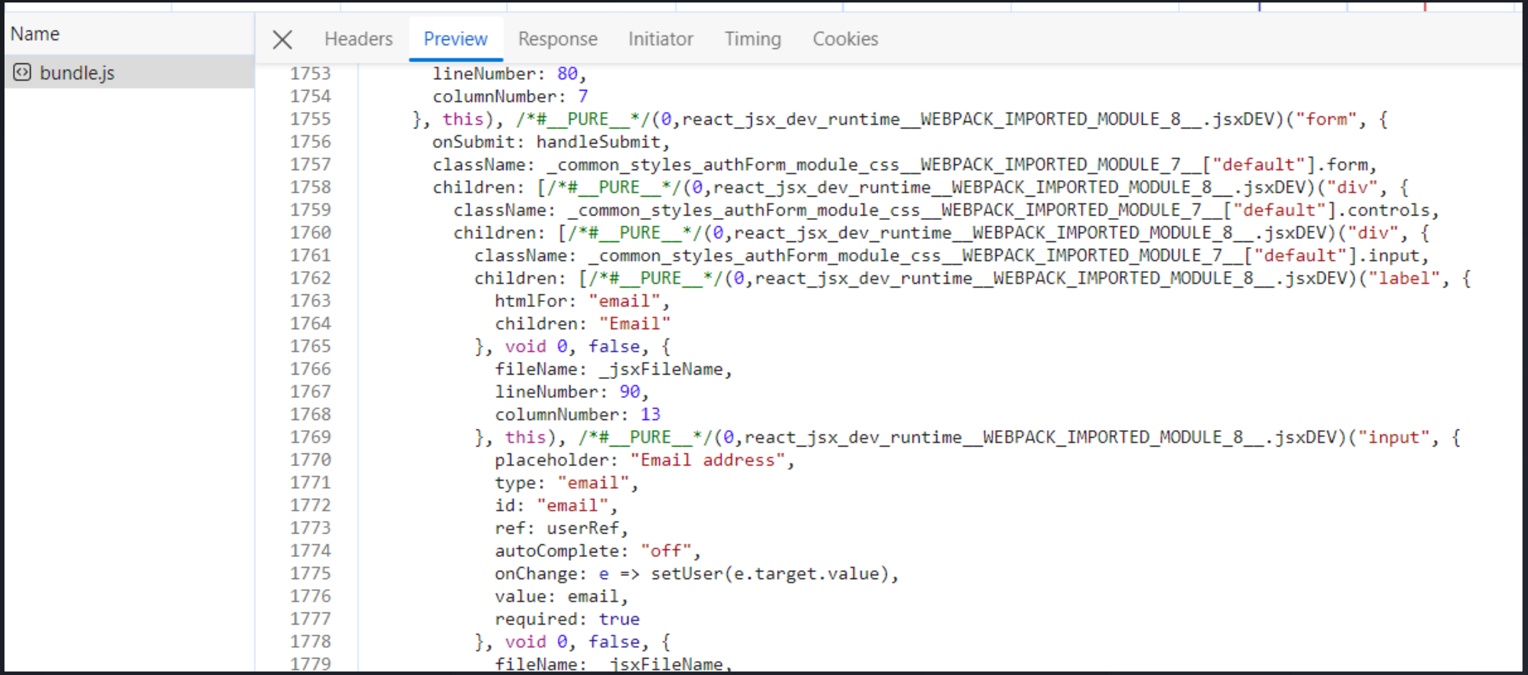
The Virtual DOM is a lightweight in-memory representation of the actual browser DOM.



```
      lineNumber: 80,
      columnNumber: 7
    }, this), /*#__PURE__*/(0,react_jsx_dev_runtime__WEBPACK_IMPORTED_MODULE_8__.jsxDEV)("form", {
      onSubmit: handleSubmit,
      className: _common_styles_authForm_module_css__WEBPACK_IMPORTED_MODULE_7__["default"].form,
      children: [/*#__PURE__*/(0,react_jsx_dev_runtime__WEBPACK_IMPORTED_MODULE_8__.jsxDEV)("div", {
        className: _common_styles_authForm_module_css__WEBPACK_IMPORTED_MODULE_7__["default"].controls,
        children: [/*#__PURE__*/(0,react_jsx_dev_runtime__WEBPACK_IMPORTED_MODULE_8__.jsxDEV)("div", {
          className: _common_styles_authForm_module_css__WEBPACK_IMPORTED_MODULE_7__["default"].input,
          children: [/*#__PURE__*/(0,react_jsx_dev_runtime__WEBPACK_IMPORTED_MODULE_8__.jsxDEV)("label", {
            htmlFor: "email",
            children: "Email"
          }, void 0, false, {
            fileName: _jsxFileName,
            lineNumber: 90,
            columnNumber: 13
          }, this), /*#__PURE__*/(0,react_jsx_dev_runtime__WEBPACK_IMPORTED_MODULE_8__.jsxDEV)("input", {
            placeholder: "Email address",
            type: "email",
            id: "email",
            ref: userRef,
            autoComplete: "off",
            onChange: e => setUser(e.target.value),
            value: email,
            required: true
          }, void 0, false, {
            fileName: _jsxFileName,
```

Virtual DOM

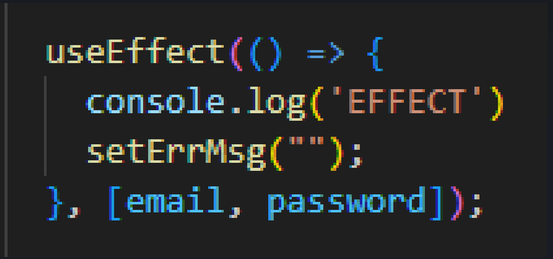
The Virtual DOM is a lightweight in-memory representation of the actual browser DOM.



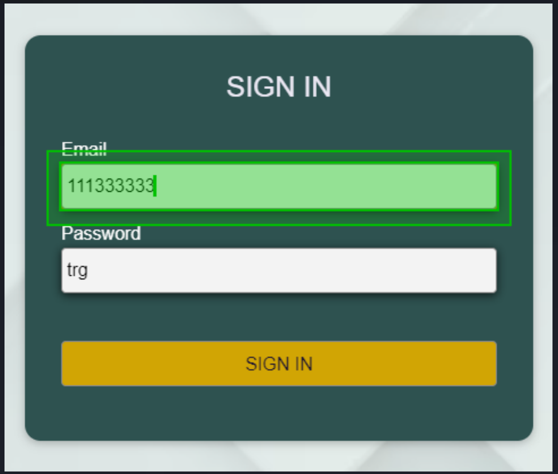
```
      lineNumber: 80,
      columnNumber: 7
    }, this), /*#__PURE__*/(0,react_jsx_dev_runtime__WEBPACK_IMPORTED_MODULE_8__.jsxDEV)("form", {
      onSubmit: handleSubmit,
      className: _common_styles_authForm_module_css__WEBPACK_IMPORTED_MODULE_7__["default"].form,
      children: [/*#__PURE__*/(0,react_jsx_dev_runtime__WEBPACK_IMPORTED_MODULE_8__.jsxDEV)("div", {
        className: _common_styles_authForm_module_css__WEBPACK_IMPORTED_MODULE_7__["default"].controls,
        children: [/*#__PURE__*/(0,react_jsx_dev_runtime__WEBPACK_IMPORTED_MODULE_8__.jsxDEV)("div", {
          className: _common_styles_authForm_module_css__WEBPACK_IMPORTED_MODULE_7__["default"].input,
          children: [/*#__PURE__*/(0,react_jsx_dev_runtime__WEBPACK_IMPORTED_MODULE_8__.jsxDEV)("label", {
            htmlFor: "email",
            children: "Email"
          }, void 0, false, {
            fileName: _jsxFileName,
            lineNumber: 90,
            columnNumber: 13
          }, this), /*#__PURE__*/(0,react_jsx_dev_runtime__WEBPACK_IMPORTED_MODULE_8__.jsxDEV)("input", {
            placeholder: "Email address",
            type: "email",
            id: "email",
            ref: userRef,
            autoComplete: "off",
            onChange: e => setUser(e.target.value),
            value: email,
            required: true
          }, void 0, false, {
            fileName: _jsxFileName,
```

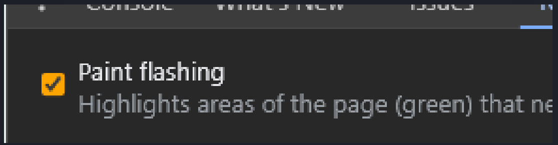
Compiled JSX will affect only the virtual DOM, which will cause the Reconciliation.

Virtual DOM

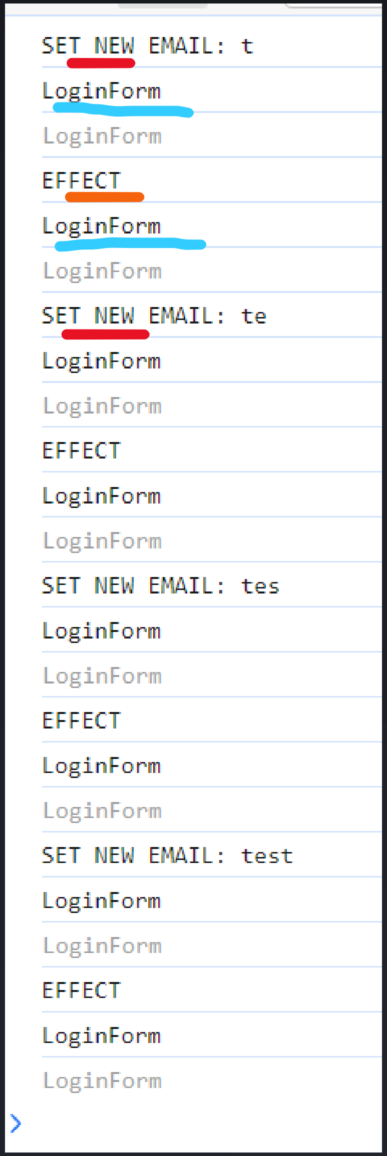


```
useEffect(() => {
  console.log('EFFECT')
  setErrMsg("");
}, [email, password]);
```

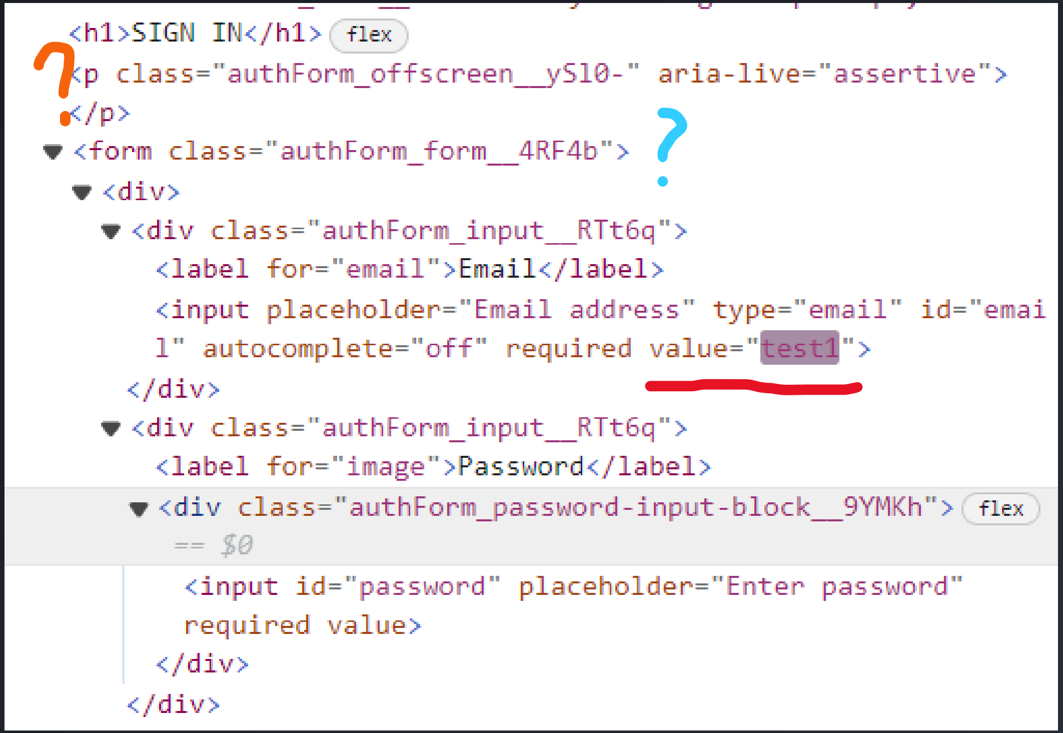




☑ Paint flashing
Highlights areas of the page (green) that ne

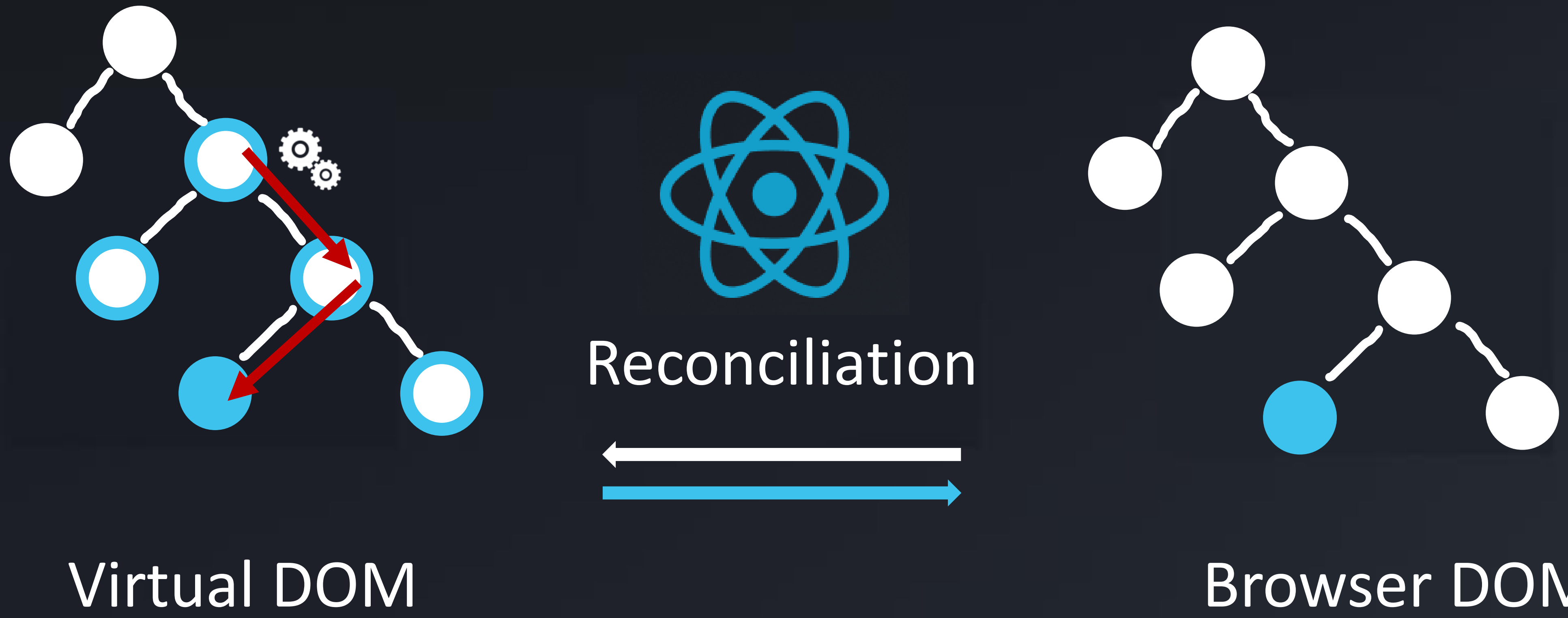


```
SET NEW EMAIL: t
LoginForm
LoginForm
EFFECT
LoginForm
LoginForm
SET NEW EMAIL: te
LoginForm
LoginForm
EFFECT
LoginForm
LoginForm
SET NEW EMAIL: tes
LoginForm
LoginForm
EFFECT
LoginForm
LoginForm
SET NEW EMAIL: test
LoginForm
LoginForm
EFFECT
LoginForm
LoginForm
>
```



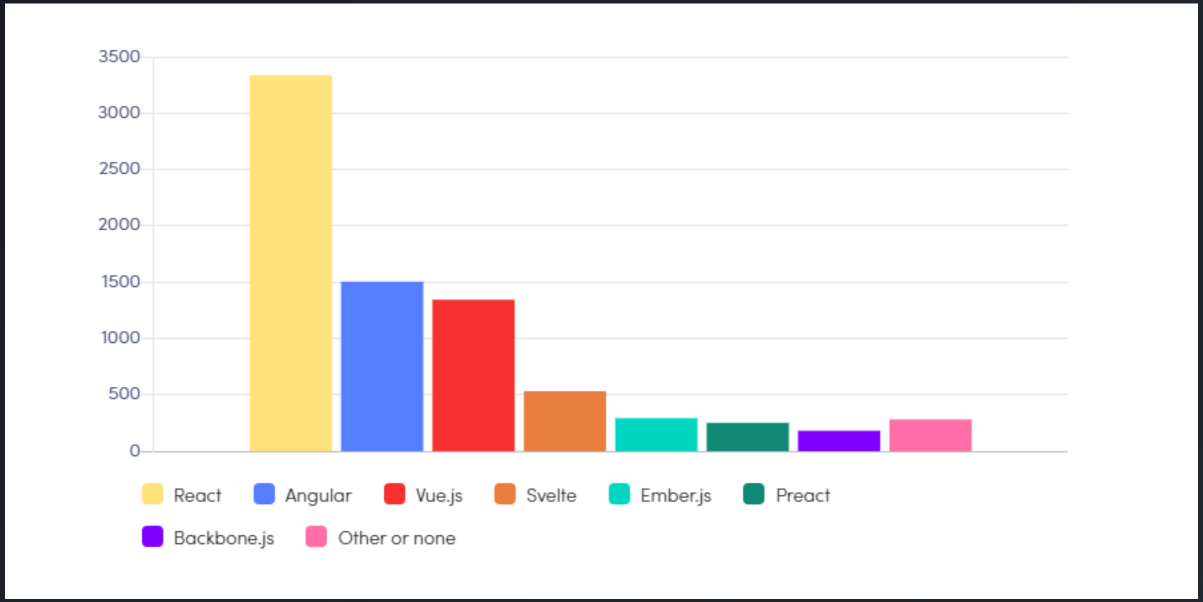
```
<h1>SIGN IN</h1> flex
<p class="authForm_offscreen__yS10-" aria-live="assertive">
</p>
▼<form class="authForm_form__4RF4b">
  ▼<div>
    ▼<div class="authForm_input__RTt6q">
       <label for="email">Email</label>
       <input placeholder="Email address" type="email" id="emai
       l" autocomplete="off" required value="test1">
     </div>
    ▼<div class="authForm_input__RTt6q">
       <label for="image">Password</label>
      ▼<div class="authForm_password-input-block__9YMKh"> flex
        == $0
         <input id="password" placeholder="Enter password"
         required value>
       </div>
     </div>
```

Virtual DOM



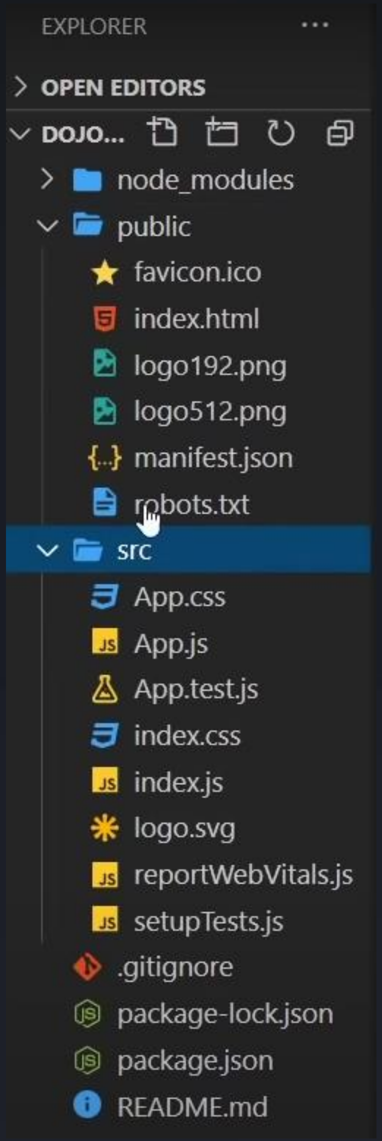
Uses ReactJS: Instagram

VS VS



The Most Popular JavaScript Frameworks in 2023

React and its friends: Create React App


EXPLORER

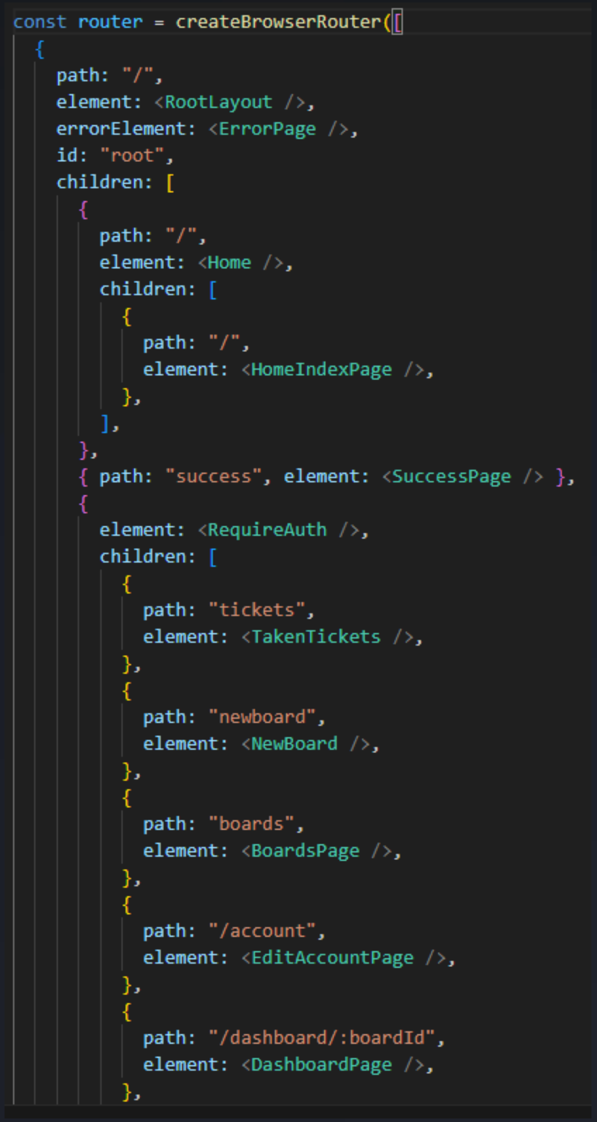
OPEN EDITORS

DOJO...

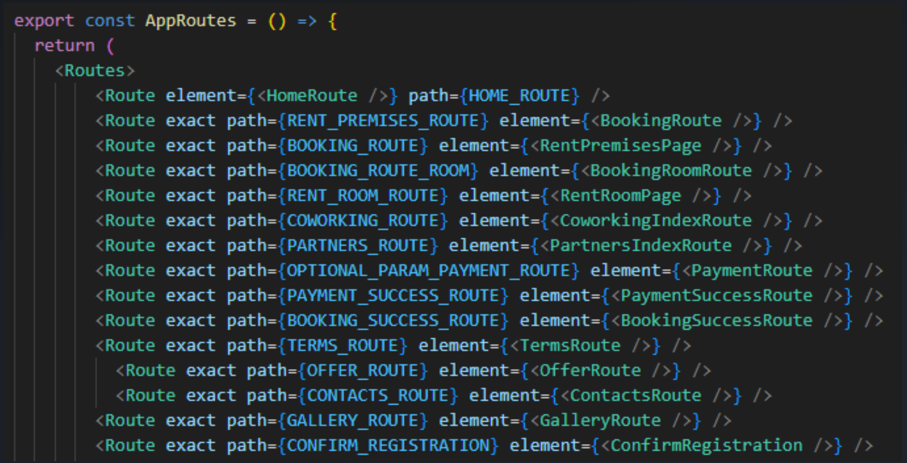
- node_modules
- public
  - favicon.ico
  - index.html
  - logo192.png
  - logo512.png
  - manifest.json
  - robots.txt
- src
  - App.css
  - App.js
  - App.test.js
  - index.css
  - index.js
  - logo.svg
  - reportWebVitals.js
  - setupTests.js
- .gitignore
- package-lock.json
- package.json
- README.md

Create React App is the most common command line tool for creating a new React application

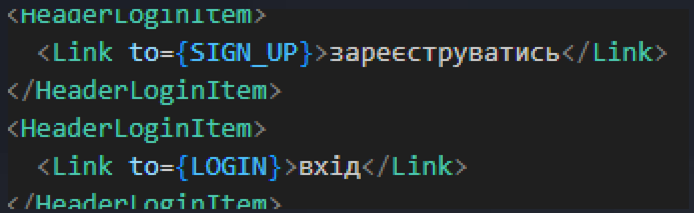
React and its friends: React Router



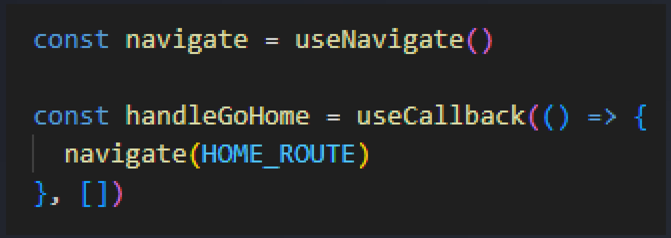
```
const router = createBrowserRouter([
  {
    path: "/",
    element: <RootLayout />,
    errorElement: <ErrorPage />,
    id: "root",
    children: [
      {
        path: "/",
        element: <Home />,
        children: [
          {
            path: "/",
            element: <HomeIndexPage />,
          },
        ],
      },
      { path: "success", element: <SuccessPage /> },
      {
        element: <RequireAuth />,
        children: [
          {
            path: "tickets",
            element: <TakenTickets />,
          },
          {
            path: "newboard",
            element: <NewBoard />,
          },
          {
            path: "boards",
            element: <BoardsPage />,
          },
          {
            path: "/account",
            element: <EditAccountPage />,
          },
          {
            path: "/dashboard/:boardId",
            element: <DashboardPage />,
          },
```



```
export const AppRoutes = () => {
  return (
    <Routes>
        <Route element={<HomeRoute />} path={HOME_ROUTE} />
        <Route exact path={RENT_PREMISES_ROUTE} element={<BookingRoute />} />
        <Route exact path={BOOKING_ROUTE} element={<RentPremisesPage />} />
        <Route exact path={BOOKING_ROUTE_ROOM} element={<BookingRoomRoute />} />
        <Route exact path={RENT_ROOM_ROUTE} element={<RentRoomPage />} />
        <Route exact path={COWORKING_ROUTE} element={<CoworkingIndexRoute />} />
        <Route exact path={PARTNERS_ROUTE} element={<PartnersIndexRoute />} />
        <Route exact path={OPTIONAL_PARAM_PAYMENT_ROUTE} element={<PaymentRoute />} />
        <Route exact path={PAYMENT_SUCCESS_ROUTE} element={<PaymentSuccessRoute />} />
        <Route exact path={BOOKING_SUCCESS_ROUTE} element={<BookingSuccessRoute />} />
        <Route exact path={TERMS_ROUTE} element={<TermsRoute />} />
          <Route exact path={OFFER_ROUTE} element={<OfferRoute />} />
          <Route exact path={CONTACTS_ROUTE} element={<ContactsRoute />} />
        <Route exact path={GALLERY_ROUTE} element={<GalleryRoute />} />
        <Route exact path={CONFIRM_REGISTRATION} element={<ConfirmRegistration />} />
```

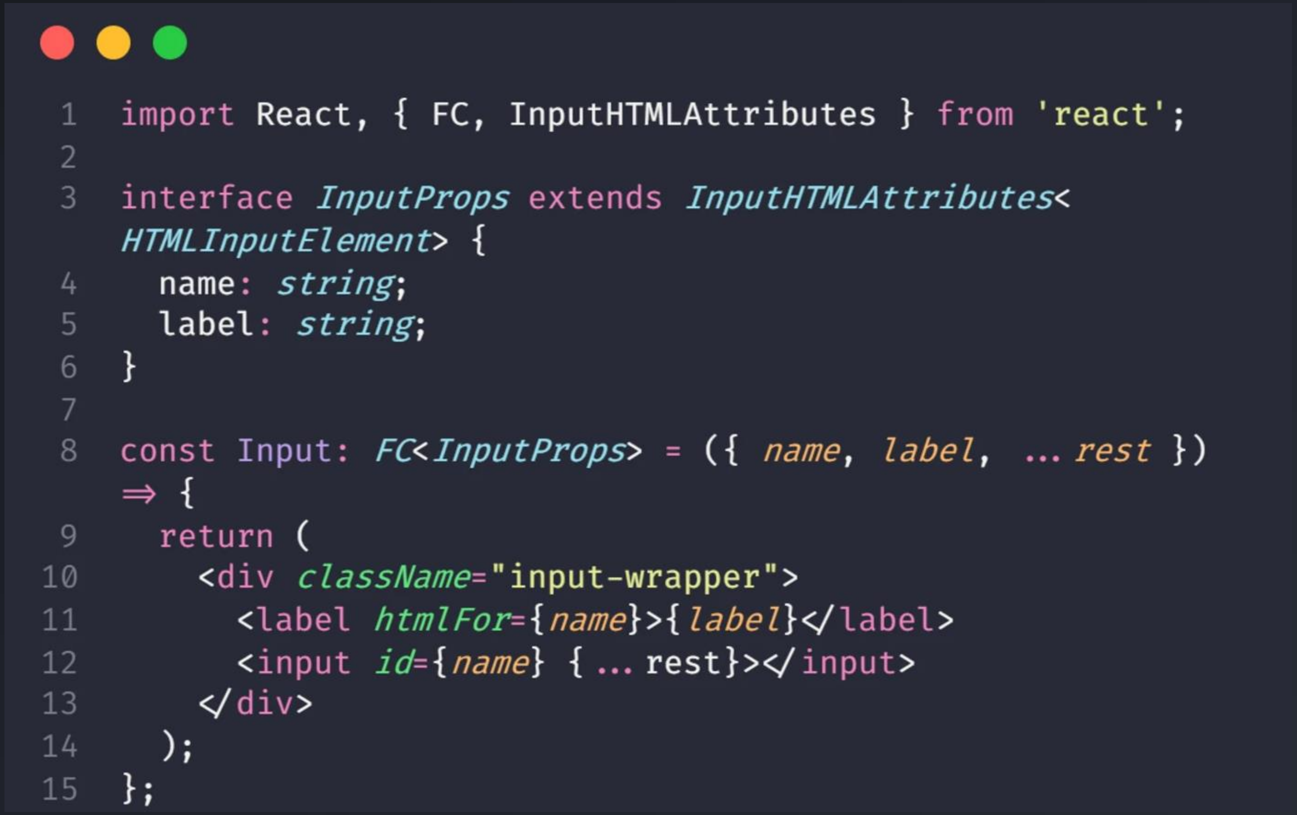


```
<HeaderLoginItem>
  <Link to={SIGN_UP}>зареєструватись</Link>
</HeaderLoginItem>
<HeaderLoginItem>
  <Link to={LOGIN}>вхід</Link>
</HeaderLoginItem>
```



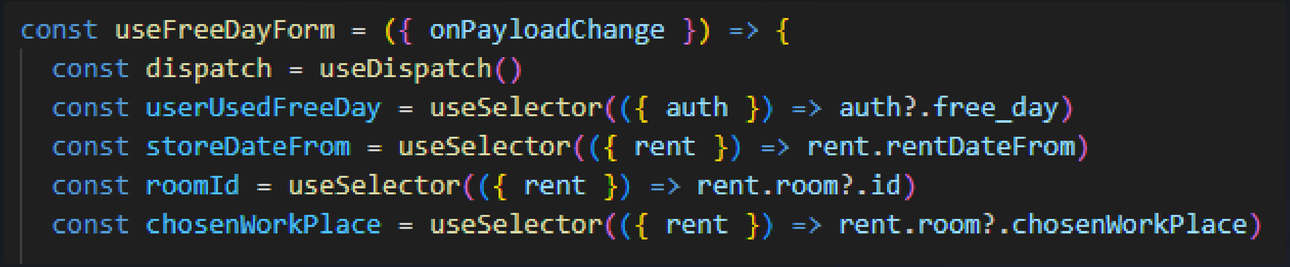
```
const navigate = useNavigate()

const handleGoHome = useCallback(() => {
  navigate(HOME_ROUTE)
}, [])
```

React and its friends: Typescript

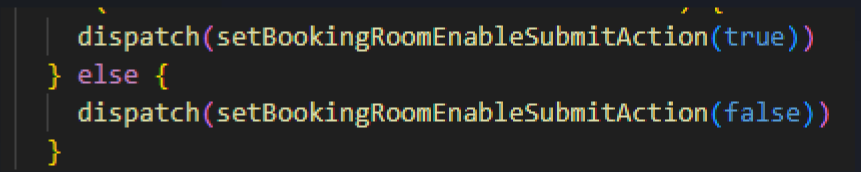


```tsx
import React, { FC, InputHTMLAttributes } from 'react';

interface InputProps extends InputHTMLAttributes<
HTMLInputElement> {
  name: string;
  label: string;
}

const Input: FC<InputProps> = ({ name, label, ...rest })
=> {
  return (
    <div className="input-wrapper">
      <label htmlFor={name}>{label}</label>
      <input id={name} {...rest}></input>
    </div>
  );
};
```

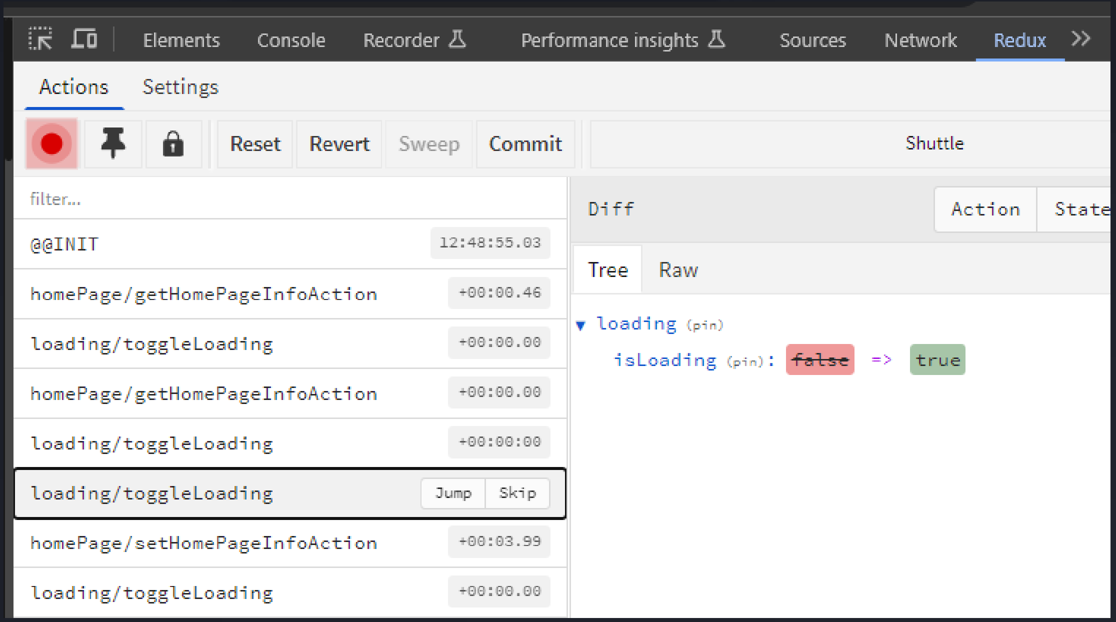
React and its friends: Redux



```
const useFreeDayForm = ({ onPayloadChange }) => {
  const dispatch = useDispatch()
  const userUsedFreeDay = useSelector(({ auth }) => auth?.free_day)
  const storeDateFrom = useSelector(({ rent }) => rent.rentDateFrom)
  const roomId = useSelector(({ rent }) => rent.room?.id)
  const chosenWorkPlace = useSelector(({ rent }) => rent.room?.chosenWorkPlace)
```

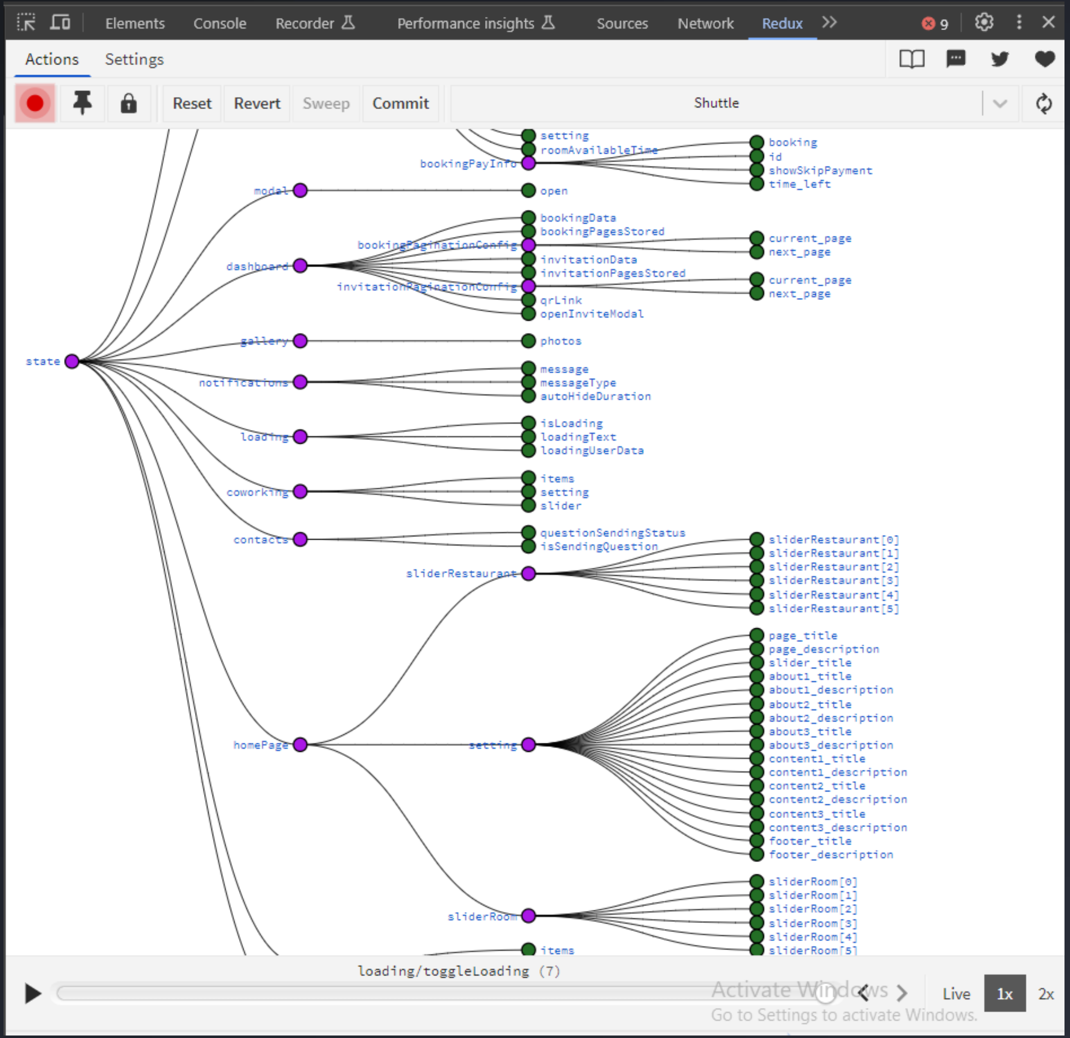


```
    dispatch(setBookingRoomEnableSubmitAction(true))
  } else {
    dispatch(setBookingRoomEnableSubmitAction(false))
  }
```

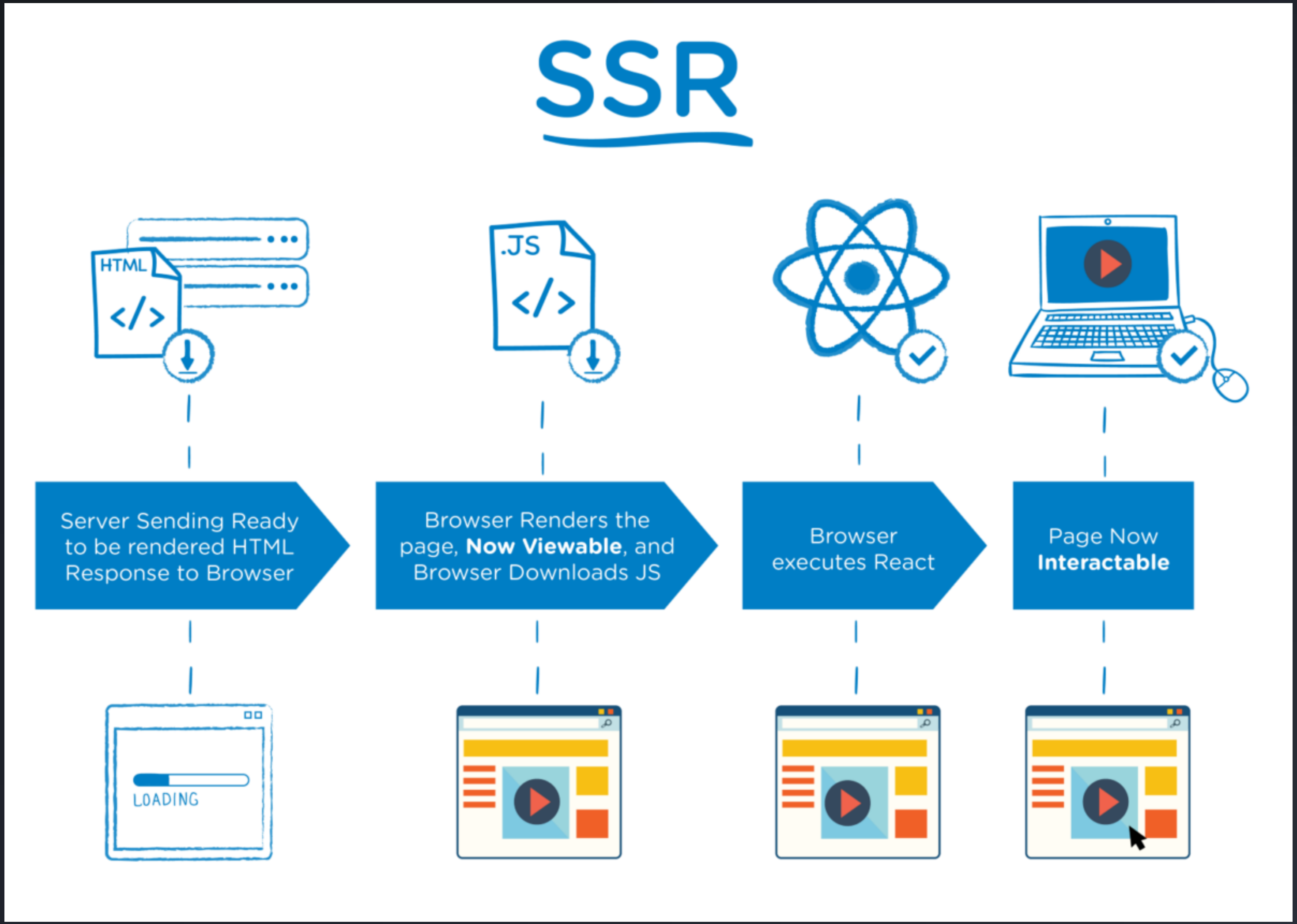


```
@@INIT                              12:48:55.03
homePage/getHomePageInfoAction      +00:00.46
loading/toggleLoading               +00:00.00
homePage/getHomePageInfoAction      +00:00.00
loading/toggleLoading               +00:00.00
loading/toggleLoading               Jump  Skip
homePage/setHomePageInfoAction      +00:03.99
loading/toggleLoading               +00:00.00
```

```
▾ loading (pin)
    isLoading (pin): ~~false~~ => true
```



React and its friends: NextJS


SSR
HTML
.JS
Server Sending Ready to be rendered HTML Response to Browser
Browser Renders the page, **Now Viewable**, and Browser Downloads JS
Browser executes React
Page Now **Interactable**
LOADING

Summarize & questions

- What happens with the component, when its state changes?
- What is Virtual DOM and Reconciliation?
- Where to put AJAX code in functional component?

- What is the only required tool to implement SPA using ReactJS?
  a) Redux
  b) React Router
  c) NextJS
  d) Client-side Java Script

What’s next?

Learn about hooks:

- useEffect
- useState
- useRef
- useMemo
- useCallback
- useContext
- useReducer
- useImperativeHandle
- useLayoutEffect

Presentation resources and recommended content

Learn about tools:

- Redux
- React router
- NextJS
- Css in React Components
- Axios
- Querying libraries