



---

[Course](#) > [React R...](#) > [Redirec...](#) > [Redirec...](#)

---

**Audit Access Expires Jun 20, 2020**

You lose all access to this course, including your progress, on Jun 20, 2020.

## Redirects

The screenshot displays a CodeSandbox editor for a React application. The code in `index.js` defines several routes and a redirect logic:

```
1 import React from "react";
2 import { render } from "react-dom";
3 import { BrowserRouter, Route, Link, Redirect, Switch } from "react-router-dom";
4
5 const Home = () => <div>Home</div>;
6 const About = () => <div>About</div>;
7 const Error = () => <div>Error: Invalid User</div>;
8
9 const Users = ({ match }) => {
10   var validUsers = ["abc", "def"];
11   if (!validUsers.includes(match.params.id)) return <Redirect to="/error" />;
12   else return <div>Welcome: {match.params.id}</div>;
13 };
14 const DefaultRoute = () => <div>Default Route</div>;
15
16 const App = () => (
17   <BrowserRouter>
18     <div>
19       <ul>
20         <li>
21           <Link to="/">Home</Link>
22         </li>
23         <li>
24           <Link to="/about">About</Link>
25         </li>
26       </ul>
27     </div>
28   </BrowserRouter>
29 );
30
31 render(<App />, document.getElementById("root"));
```

The preview on the right shows a list of links: [Home](#), [About](#), [Users: abc](#), and [Users: inv](#). Below the links, the text "Home" is displayed.

The video player at the bottom shows a timestamp of 0:03 / 1:47 and a speed of 1.50x.

## Video

[Download video file](#)

## Transcripts

[Download SubRip \(.srt\) file](#)

[Download Text \(.txt\) file](#)

## Redirects

### Redirect

The Redirect component will navigate the page to a specified location when rendered.

We can import the Redirect component with:

```
import { Redirect } from 'react-router-dom'
```

We can reference the Redirect component using the `<Redirect>` tag.

```
<Redirect to="/login"/>
```

## Redirect from Routes

Redirect components are often used inside Route components. When the Route is matched, there is often logic that triggers the Redirect component. If you directly render a Redirect component outside of a Route component, it will just navigate the page away immediately.

```
var validUser = false

const ValidateUser = () => {
  if(validUser){
    return <div>Welcome User</div>
  }
  else{
    return <Redirect to="/error"/>
  }
}

const Login = () => {
  validUser = true
  return <div>Logged In</div>
}

const Error = () => {
  return <div>Please Log In</div>
}

<Route exact path="/user" render={ValidateUser}/>
<Route exact path="/error" render={Error}/>
<Route exact path="/login" render={Login}/>
```

## Redirect from Switches

Redirects can also be used inside Switches. To use a Redirect within a Switch component, add a `from` property to the Redirect component. The Switch component will render the first component that matches, whether it is a Route or a Redirect. The `from` property can be only used with Redirects that are within Switch components.

```
<Switch>
  <Route path='/user' component={User}/>
  <Redirect exact from="/" to='/user' />
  <Redirect from='/oldPageURL' to ='/newPageURL' />
  <Route component={DefaultPage}/>

</Switch>
```

## push

The Redirect component has a `push` attribute that will push the redirected URL onto the history stack instead of replacing the current one.

```
<Redirect push to="/login" />
```

## exact

Just like Route components, the Redirect component has an `exact` attribute. See the `exact` section in the Routers and Routes page for more details.

## strict

Just like Route components, the Redirect component has a `strict` attribute. See the `strict` section in the Routers and Routes page for more details.

Try out this live example on CodeSandbox: <https://codesandbox.io/s/ool93klrl9>