

EdX and its Members use cookies and other tracking technologies for performance, analytics, and marketing purposes. By using this website, you accept this use. Learn more about these technologies in the [Privacy Policy](#).

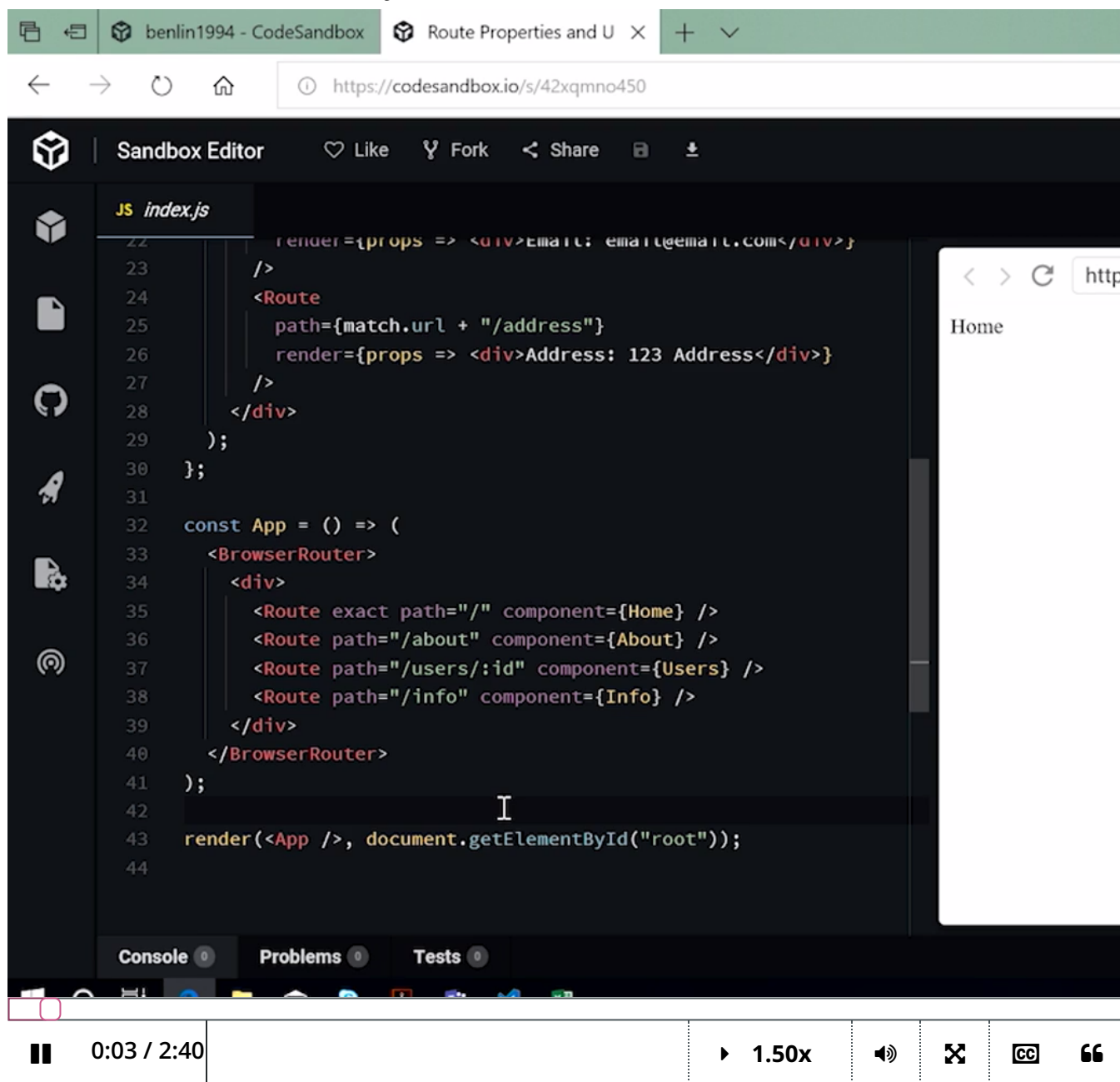


[Course](#) > [React R...](#) > [Router...](#) > [Route ...](#)

Audit Access Expires Jun 20, 2020

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

Route Properties and URL Parameters



The screenshot shows a CodeSandbox editor with the following code in `index.js`:

```
22 render={props => <div>Email: email@email.com</div>}
23 />
24 <Route
25   path={match.url + "/address"}
26   render={props => <div>Address: 123 Address</div>}
27 />
28 </div>
29 );
30 };
31
32 const App = () => (
33   <BrowserRouter>
34     <div>
35       <Route exact path="/" component={Home} />
36       <Route path="/about" component={About} />
37       <Route path="/users/:id" component={Users} />
38       <Route path="/info" component={Info} />
39     </div>
40   </BrowserRouter>
41 );
42
43 render(<App />, document.getElementById("root"));
44
```

The browser preview on the right shows the URL `http://localhost:3000/` and the text `Home`.

Video

[Download video file](#)

Transcripts

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

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

Route Properties and URL Parameters

Route properties

All three render methods will have route properties passed down from the Route component. Here are the route properties that are passed down as props:

- match
- location
- history

You can access them through props, just like any other React property:

```
const About = (props) => {
  console.log(props.match)
  // Object {path: "/about", url: "/about", isExact: true, params: O
  console.log(props.location)
  // Object {pathname: "/about", search: "", hash: "", state: undefi
  console.log(props.history)
  // Object {length: 1, action: "POP", location: Object}
  return (
    <div>
      About
    </div>
  )
}

const App = () => (
  <div>
    <BrowserRouter>
      <Route path="/about" component = {About}/>
    </BrowserRouter>
  </div>
);
```

You can also access them through ES6 destructuring:

```
const App = () => (  
  <div>  
    <BrowserRouter>  
      <Route path="/about" render={ ({match,location,history}) => {  
        console.log(match)  
        // Object {path: "/about", url: "/about", isExact: true, par  
        console.log(location)  
        // Object {pathname: "/about", search: "", hash: "", state:   
        console.log(history)  
        // Object {length: 1, action: "POP", location: Object}  
        return (  
          <div>About</div>  
        )  
      } } />  
    </BrowserRouter>  
  </div>  
) ;
```

match

A `match` object is passed down to all components rendered from `Route` components as a property. The `match` object is null when there is no match, but contains useful information when a match occurs. The `match` object contains the following information about a matched path that caused a component to be rendered:

- `params` (object) - contains key/value pairs of the dynamic segments that are used in the path pattern (e.g. `{userId = 123}` when path = `"/user/:userId"` and the URL is `"/user/123"`)
- `isExact` (boolean) - true if the URL is an exact match with the path property
- `path` (string) - the `path` property of the route component. Useful for building nested routes
- `url` (string) - the matched portion of the URL. Useful for building nested Links

location

A `location` object is passed down to all components rendered from `Route` components as a property. The `location` object contains the following information about the path that was rendered:

- `pathname` (string) - the full path of the URL

- search (string) - the URL query string
- hash (string) - the URL hash fragment
- state (object) - the current state, if a state was provided to `history.push(path, [state])`, otherwise undefined

history

A `history` object is passed down to all components rendered from `Route` components as a property. The `history` object contains the following information about the path that was rendered:

- length (number) - The number of entries in the history stack
- action (string) - (PUSH, POP, or REPLACE)
- location (object) - see `location`

It also has the following methods:

- `push(path, [state])` - (function) Pushes a new entry onto the history
- `replace(path, [state])` - (function) Replaces the current entry on the history stack
- `go(n)` - (function) Moves the pointer in the history stack by `n` entries
- `goBack()` - (function) Equivalent to `go(-1)`
- `goForward()` - (function) Equivalent to `go(1)`
- `block(prompt)` - (function) Prevents navigation (see the history docs)

URL Parameters

URL parameters are segments of your URL path that can vary and be parsed from the URL. To define a segment of your path to be a URL param, add a colon in front of the segment name:

```
<Route path="/userId/:id" render={(props) => <div>Home</div>}/>
```

The “:id” portion of the URL path can now vary and the route will still match. For example “/userId/123” and “/userId/abc”, will both cause a match.

You can access the URL param value by using the `match.params` object that is passed down to the rendered component:

```
const User = ({match}) => {  
  return (  
    <div>  
      UserId: {match.params.id} //this will show the id entered in the  
    </div>  
  )}  
  
const App = () => (  
  <div>  
    <BrowserRouter>  
      <Route path="/userId/:id" component = {User}/>  
    </BrowserRouter>  
  </div>  
);
```

Nested Routes

It is pretty straight forward to nest Route components within each other. The `match.url` property is useful if you want to add path segments onto the original matched url. For example, if you want to add additional Route components within a component rendered from an original Route component do the following:

```
const Info = ({match}) => {
  return (
    <div>
      <Route path="{match.url} + '/phone'" render={props} => <div>P
      <Route path="{match.url} + '/email'" render={props} => <div>E
      <Route path="{match.url} + '/address'" render={props} => <div>
    </div>
  )}

const App = () => (
  <div>
    <BrowserRouter>
      <Route path="/info" component = {Info}/>
    </BrowserRouter>
  </div>
);
```

Check out this live example on CodeSandbox: <https://codesandbox.io/s/42xqmno450>

© All Rights Reserved