
COMP 431/531: Web Development

Lecture 12: React Routing

Mack Joyner (mjoyner@rice.edu)

<https://www.clear.rice.edu/comp431>



Announcements & Reminders

HW #4 (Draft Front-end) due Tue. Oct. 21st at 11:59pm

classroom hw4 repo: https://classroom.github.com/a/hfLD_40C



Deploying HW #4 Front-End to Surge

- Deploying app using Surge:

>> ng build (or npm run build)

Fix TypeScript build errors

- Project name folder appears in **dist**

>> *cd dist/*

- Check dir for index.html (avoid page not found error for surge url)

>> *surge*

- Client side only app (SPA) for React, may need to add in package.json:

"build": "tsc -b && vite build && cp build/index.html build/200.html"

- Data persistence not a requirement for this assignment

- **page refresh or navigating away from page restores original data**



HW #4 Demo



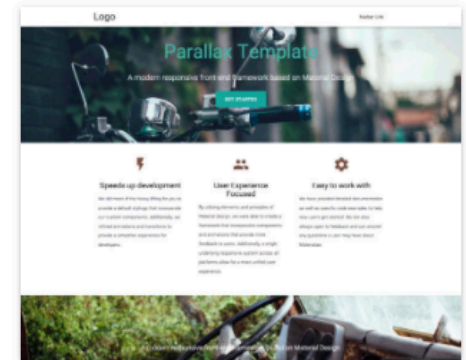
Libraries

- bootstrap
- material
- font awesome
- material-ui/core
- react-icons
- forms-react (sign-in, registration)
- react-define (define actions)
- axios (http requests)
- tailwind, shadcn

Starter Template

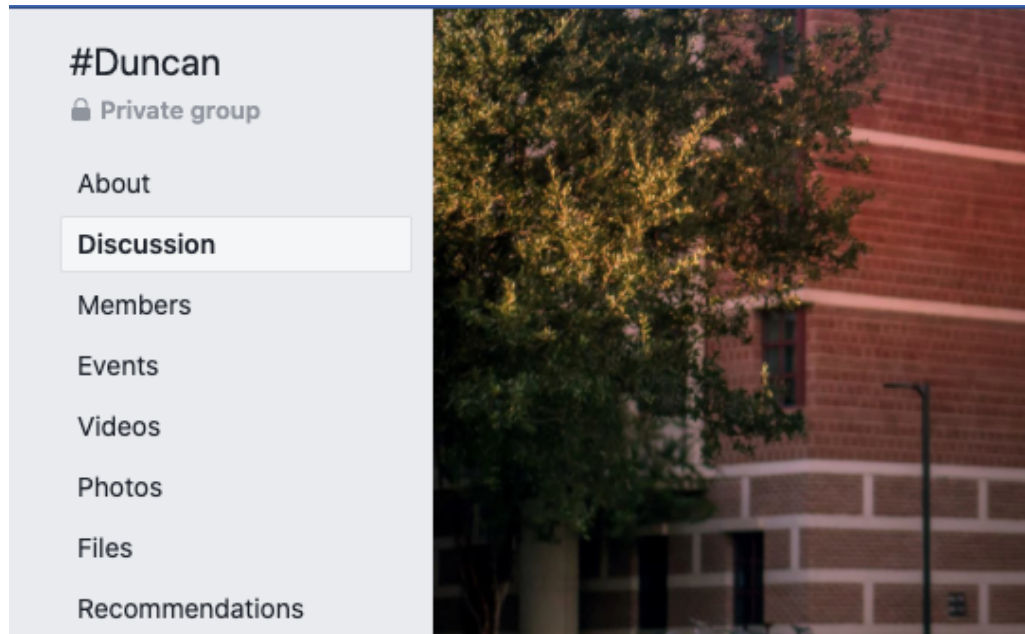


Parallax Template



React Routing

Navigate between different pages of application with authenticated users



React Routing

- `>> npm install —save react-router-dom`
- BrowserRouter for dynamic web app
- Import:
 - BrowserRouter as Router
 - Routes to bundle together multiple routes.
 - Route employs path for navigation

```
12  createRoot(document.getElementById('root')).render(  
13    <StrictMode>  
14      <Provider store={store}>  
15        <Router>  
16          <Routes>  
17            <Route exact path="/" element={<App />}/>  
18          </Routes>  
19        </Router>  
20      </Provider>  
21    </StrictMode>  
22  )  
23
```

No longer use `exact` when matching paths in Routes



React Route with Path

- Route employs **path** for navigation
- Router renders element for first path match

```
13  createRoot(document.getElementById('root')).render(  
14      <StrictMode>  
15          <Provider store={store}>  
16              <Router>  
17                  <Routes>  
18                      <Route path="/" element={<App />}/>  
19                      <Route path="/players" element={<Players />}/>  
20                  </Routes>  
21              </Router>  
22          </Provider>  
23      </StrictMode>  
24  )
```

Alt: element={App}



React Route with Path

- Alternatively use `render={() => (...)}` instead of `element={Game}`

- May use path parameters (“user/:userid”)

```
<Router>
  <Routes>
    <Route exact path="/">
      <Login />
    </Route>
    <Route path="/home" render={(props) => <Main {...props}/>}/>
    <Route path="/profile" render = {(props) => <Profile {...props}/>}/>
  </Routes>
</Router>
```

- Component `match` prop has params, url



Links

- Link enables navigation with hyperlinks
- Specify place to navigate (to=)
- Link can accept identifier `/players/{id}` where `id` is a defined variable
- May also find alternative react hook `useNavigate()` to be useful

```
15 class Game extends React.Component {
16   render() {
17     return (
18       <div className="game">
19         <div className="game-board">
20           <Board/>
21         </div>
22         <div className="game-info">
23           <div>{/* status */</div>
24           <ol>{/* TODO */</ol>
25         </div>
26         <div>
27           <Link to={"/players"}>Players</Link>
28         </div>
29       </div>
30     );
31   }
32 }
```



Router Hook: useHistory

- Access router for navigation
- Support navigating forward/back

```
import {useHistory} from "react-router-dom";

const history = useHistory();

function handleSubmit(event) {
  event.preventDefault();
  history.push({
    pathname: "/home"
  });
}

<form onSubmit={handleSubmit}>
```

Halt normal behavior



Component Lifecycle Methods

- `componentWillMount`
 - invoked once, on client and server, before rendering occurs
- `componentDidMount`
 - invoked once, on client, after rendering occurs
- `componentWillUpdate`
 - preprocessing, before render called
- `componentDidUpdate`
 - postprocessing, after render called
- `componentWillUnmount`
 - invoked once, before unmounting component



React Hook: useEffect

- Synchronize component with external system
- Setup runs when component connects to DOM
- Dependencies list values referenced in setup

`useEffect(setup, dependencies?)`

```
const [serverUrl, setServerUrl] = useState('https://localhost:1234');
```

```
useEffect(() => {  
  const connection = createConnection(serverUrl, roomId);  
  connection.connect();  
  return () => {  
    connection.disconnect();  
  };  
}, [serverUrl, roomId]);
```

Rerun when serverUrl or roomId changes

<https://react.dev/reference/react/useEffect>



AJAX Requests in React

- Components lifecycle method: requires a component
- `useEffect` (hook) doesn't require component class

```
> fetch
< f fetch() { [native code] }
> fetch('https://jsonplaceholder.typicode.com/posts').then(r => r.json())
< ▼ Promise {<pending>} ⓘ
  ▶ __proto__: Promise
  [[PromiseStatus]]: "resolved"
  ▼ [[PromiseValue]]: Array(100)
    ▶ 0: {userId: 1, id: 1, title: "sunt aut facere repellat provident", body: "quia et est Lorem ipsum dolor sit amet consectetur adipiscing elit dolor"}
    ▶ 1: {userId: 1, id: 2, title: "qui est esse", body: "est rerum tempore vero, et est ex animam"}
    ▶ 2: {userId: 1, id: 3, title: "ea molestias quasi exercitationem repellat qui ipsa sit", body: "et est ex animam"}
    ▶ 3: {userId: 1, id: 4, title: "eum et est occaecati", body: "ullam et saepe de qui tempora ut commodi"}
    ▶ 4: {userId: 1, id: 5, title: "nesciunt quas odio", body: "repudiandae veniam quaerat sunt sed"}
    ▶ 5: {userId: 1, id: 6, title: "dolorem eum magni eos aperiam quia", body: "ut enim aut vero"}
    ▶ 6: {userId: 1, id: 7, title: "magnam facilis autem", body: "dolore placeat quibusdam"}
    ▶ 7: {userId: 1, id: 8, title: "dolorem dolore est ipsam", body: "dignissimos ad eum"}
    ▶ 8: {userId: 1, id: 9, title: "nesciunt iure omnis dolorem tempora et accusantium", body: "consectetur ut"}
    ▶ 9: {userId: 1, id: 10, title: "optio molestias id quia eum", body: "quo et aut ut"}
    ▶ 10: {userId: 2, id: 11, title: "et ea vero quia laudantium autem", body: "consectetur ut"}
    ▶ 11: {userId: 2, id: 12, title: "in quibusdam tempore odit est dolorem", body: "consectetur ut"}
    ▶ 12: {userId: 2, id: 13, title: "dolorum ut in voluptas mollitia et saepe quia voluptate", body: "consectetur ut"}
    ▶ 13: {userId: 2, id: 14, title: "voluptatem eligendi optio", body: "fuga et voluptatem"}
    ▶ 14: {userId: 2, id: 15, title: "eveniet quod temporibus", body: "reprehenderit"}
    ▶ 15: {userId: 2, id: 16, title: "sint suscipit perspiciatis velit dolorum rerum ipsa", body: "consectetur ut"}
    ▶ 16: {userId: 2, id: 17, title: "fugit voluptas sed molestias voluptatem pro", body: "consectetur ut"}
    ▶ 17: {userId: 2, id: 18, title: "voluptate et itaque vero tempora molestiae", body: "consectetur ut"}
    ▶ 18: {userId: 2, id: 19, title: "adipisci placeat illum aut reiciendis qui", body: "consectetur ut"}
    ▶ 19: {userId: 2, id: 20, title: "doloribus ad provident suscipit at", body: "consectetur ut"}
    ▶ 20: {userId: 3, id: 21, title: "asperiores ea ipsam voluptatibus modi minima", body: "consectetur ut"}
    ...
```

Inline function callbacks with arrow notation

Same as:
`.then(function(r) {
 return r.json()
})`
`r.json()` returns a *Promise*, the next `then()` is called when `json()` resolves.



Async/Await

- **Async** functions always return Promise
 - **Await** can only be inside async
-

```
async function addNewArticles() {  
  let newArts = await fetch(...)  
  newArts = await newArts.json()  
  newArts = newArts.articles  
  articles.push(...newArts)  
}
```

Expression after **await** is like using
Promise then



Repeating Elements in Virtual DOM

```
<li><span className="fancy">thing1</span></li>
<li>thing2</li>
{ Array(5).fill(1).map((x, i) => <li key={i}>thing{3 + i}</li>) }
```

React uses the **key** to identify separate repeated elements.

- thing1
- thing2
- thing3
- thing4
- thing5
- thing6
- thing7



React Developer Tools (Redux)

Display when visiting players page after clicking on Players link in Board

Players

1. Bret
2. Antonette
3. Samantha
4. Karianne
5. Kamren
6. Leopoldo_Corkery
7. Elwyn.Skiles
8. Maxime_Nienow
9. Delphine
10. Moriah.Stanton

[Home](#)

The screenshot shows the React Developer Tools interface with the 'Components' panel active. The component tree on the left shows the following structure:

- Provider
 - ReactRedux.Provider
 - BrowserRouter
 - Router
 - Navigation.Provider
 - Location.Provider
 - Routes
 - RenderedRoute
 - Route.Provider
 - Players** (highlighted)
 - Link

The right panel shows the props and hooks for the selected 'Players' component:

- props
 - new entry: ""
- hooks
 - Selector2: ["Bret", "Antonette", "Sama..."]
 - Dispatch2:
 - Effect: f () {}
- rendered by
 - createRoot()
 - react-dom@18.3.1

