

React-Router

Goal

The goal of this lab is to build a Single Page Application using React and [React-Router](#).

The Mission

In this lab you will convert a simple React movie editor to a proper Single Page Application. The original movie editor uses one single URL and internal state to manage what is displayed to the user. While this works it has several drawbacks. One big drawback is that when the user refreshed the browser all state is lost and the main page is displayed. Another drawback is that users can't bookmark individual pages. A third drawback is that the user can't use the browsers back and forward buttons.

The starter application in this exercise contains a working movie editor. This movie editor doesn't use the browsers navigation capabilities yet. In this lab you will use React-Router to enable normal navigation inside of the application. This design change will remove all the mentioned issues.

You can start the application by running the start.bat file in the root folder. When you do a small Node.js server will start. Next the default browser will start to display the index.html page. Please note that the first time you start this will take some time as number of NPM packages will download. This requires an internet connection where the NPM registry is not blocked by a firewall.

Note: The **FINISHED** folder contains a finished version of this lab as a reference.

Type it out by hand?

Typing it drills it into your brain much better than simply copying and pasting it. You're forming new neuron pathways. Those pathways are going to help you in the future. Help them out now.

Assignment 1

Start the application in the **BEGIN** folder and familiarize yourself with the functionality. The application should be familiar to you as this is the result of the lab with the introduction module. Note that when you switch between the movie list and editing the URL in the browser is not updated. Refresh the browser when viewing the edit page. Observe that the movie list is shown instead of the last edit page. Click on the edit button of a movie and once the editor is visible click on the browsers back button. Observe that you navigate away from the page to whatever page was active before.

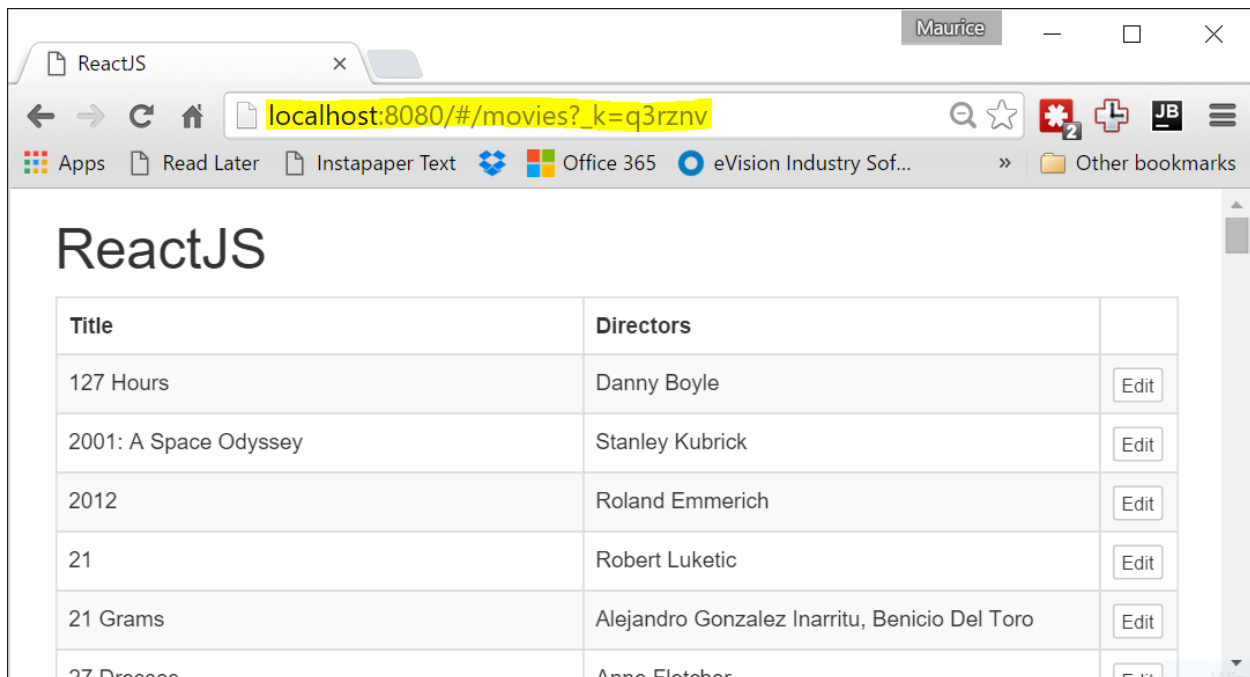
Add the React-Router NPM package to the project. This is done using the command: `npm install react-router-dom --save`. Make sure to include the `--save` option so it is save in the `PACKAGE.JSON` and reinstalled when needed.

Open `APP.JS` and import `BrowserRouter`, `Switch`, `Route` and `Redirect` from `react-router-dom`. Using React-Router you can define the following routes:

1. A route with path `"/movies"` that will render the `MovieList` component.
2. A second nested route with the path `"/movie/:id"` that will render the `MovieEdit` component.
3. Add a redirect from anything else to `"/movies"` for when users enters an invalid URL.

Update the `App` component to render the child components as defined by the routing. Also remove the state and related functions from the `App` component. These are no longer needed as the browsers URL provides this state.

Make sure the application works and shows the `/movies` URL with the movies list as below. Please note that navigation using the edit button doesn't work yet. We will be fixing that in the next task. For now you can type in the URL <http://localhost:3000/movie/278> manually. This will take you to the edit page.



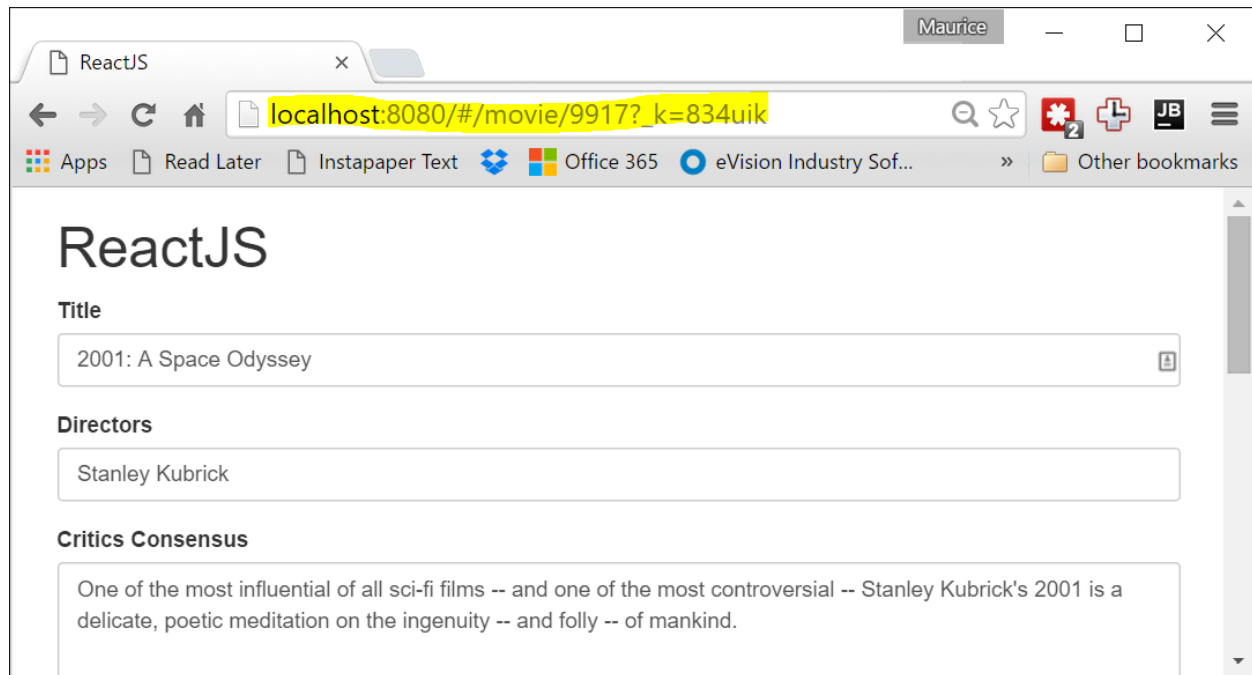
Title	Directors	
127 Hours	Danny Boyle	Edit
2001: A Space Odyssey	Stanley Kubrick	Edit
2012	Roland Emmerich	Edit
21	Robert Luketic	Edit
21 Grams	Alejandro Gonzalez Inarritu, Benicio Del Toro	Edit
27 Dresses	Anne Fletcher	Edit

Assignment 2

Open `MOVIEROW.JS` and locate the `MovieRow` component. Update the `MovieRow` component changing the button in the third column. Instead of a html button element this should become a React-Router `Link` component. The target should be the edit URL for the movie at `"/movie/:id"` where the `:id` should be the actual ID from the movie selected.

Locate the `MovieList` component in the Open `MOVIELIST.JS` and remove the `toEditMode` property. This property was passed from the `App` to the `MovieRow` for navigation but is no longer needed.

Make sure you can now navigate from the movie list to the movie edit page. Note that the edit page has not been updated yet and can't display the selected movie yet. We will fix that issue in the next task. You should be able to use the browsers back button to return to the movie list.



Assignment 3

Open the **MOVIEEDIT.JS** and locate the **MovieEdit** component. The **MovieEdit** component tries to load the movie data from the server. This happens in the **componentDidMount()** function. Before the movie id was passed in from the **App** component. As that is no longer the case we need to retrieve it from the routing **match** parameter. These route parameters are passed as **this.props.match.params**. This object contain a property for each route parameter defined. In this case there is one called **id** containing the movie id from the URL. Make sure to use the same route parameter in the **save()** function.

Update the **Save** and **Cancel** buttons to use navigation. Change the **Cancel** button to use a **Link** component for navigation. The target URL should be **"/movies"**. In the **save()** function we also need to update the navigation to get back to the list after a successful save. In this case we can't use a **Link** component as we need conditional control. This can be done using the **this.props.history.push()** function.

Make sure the application is functional and all navigation works as expected.