Lab 6
React Router

CSCI2720 Building Web Applications

Agenda

- Tooling ecosystem
- Accessing command line
- Basics of React Router
- URL parameters
- No-match route
- Functional vs class components

Tooling Ecosystem

- We don't want to work with bare HTML/CSS/JS
- More and more tools are there to help us for…
  - *Safety net*
    - E.g., checking errors in code
  - *Transformation*
    - E.g., transpiling JSX to plain JS
  - *Post-development*
    - E.g., testing and deployment

Tooling Ecosystem

- To ensure easy incorporation of multiple tools, the command line interface (CLI) is the best choice
  - Consistency with clear syntax
  - Easily scriptable for automation
- Many tools provide CLI
  - npm, React CLI, Netlify CLI, …

Accessing CLI on your computer

- *(Note: For simplicity, our lab will demo using npm using cloud services, but you can try it on your own computer too.)*
- On Linux and macOS, the ***Terminal*** is ready for use
  - Just search for it on your computer
- On Windows, the ***Command Prompt*** (cmd) is a bit primitive
  - Just run ***cmd*** from the start menu: it is okay to use cmd in this stage
  - More preferred: *Powershell*
  - Advanced users: *Windows Subsystem for Linux* (WSL)
    - *See: https://docs.microsoft.com/en-us/windows/wsl/install*
- There are other possibilities, e.g., virtual machines, cloud services
  - We will try AWS later

Basic commands

- While Linux and macOS are similar, Windows have a different set of commands
- A quick table for reference

	Linux/macOS	Windows
List dir contents	`ls -l`	`dir`
Change dir	`cd` *dir*	`cd` *dir*
Make new dir	`mkdir` *dir*	`mkdir` *dir*
Copy file	`cp` *fileA fileB*	`copy` *fileA fileB*
Move file	`mv` *fileA fileB*	`move` *fileA fileB*
Delete file	`rm` *file*	`del` *file*
Delete dir	`rm -rf` *dir*	`rmdir` *dir*
Show file contents	`cat` *file*	`type` *file*

  - Some more details: *https://www.geeksforgeeks.org/linux-vs-windows-commands/*
- A lot more to learn. Good luck!
  - *Look for online tutorials!*

npm and npx

- npm is Node's package manager
  - It can be downloaded together with Node.js at *https://nodejs.org*
- Other than the Node.js platform, it is also good for basic web development
- More and more other tools are now provided as npm packgaes
- ***npm*** allows easy management of packages
- ***npx*** lets you execute without installing
  - Newest version guaranteed

Installing React-router

- Home
- About

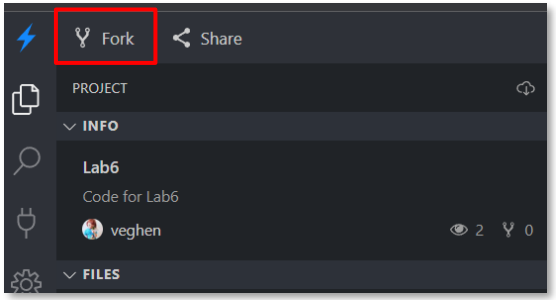
About

- Start with this link with ***Google Chrome***

  https://stackblitz.com/edit/node-xrctfw

  A React app is already created for you.

  - ***Fork*** the project to edit and save a copy

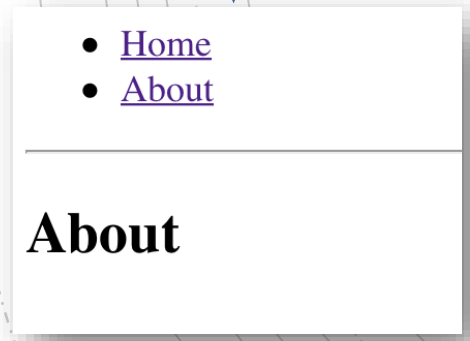


  - In the **Terminal**, enter the following commands
    - `cd demo-app/  # enter app folder`
    - `npm install react-router-dom`
      `# install react-router-dom`
    - `npm start  # start server`

Basic Components

- Look at the given file `src/index.js`
  - You will make your edits here
- The router
  - **`<BrowserRouter>`**
    - For modern browsers, supporting HTML5 History API with states, e.g., the *Back* button
- Route matchers
  - **`<Routes>`** looks at children `<Route>` elements for the first match, and ignore others
  - **`<Route>`** matches URL against the **`path="…"`** attribute/props
- Route changers
  - **`<Link>`** allows specifying the **`to`** attribute

In a nutshell…



- The given files has these features:
  - A list of links, and the linked component are displayed inside the component **App**
    - If the link ***Home*** is visited (URL becomes `/`), the **Home** component is shown
    - If the link ***About*** is visited (URL becomes `/about`), the **About** component is shown instead

Task 1:

URL parameters

- A variable could be matched inside the URL

1. Set up three more **<Link>**, pointing to **/file/fileA**, **/file/fileB**, and **/file/fileC**
   - You can decide what *labels* they should take
2. Under **<Routes>**, add one more **<Route>**

```
<Route path="/file/:id" element={<File/>} />
```

3. Add this line in the top of the js file

```
import { useParams, useLocation } from
'react-router-dom';
```

4. Set up a new component **File**

```
function File() {
  let { id } = useParams();
  return (
    <div>
      <h3>ID: {id}</h3>
    </div>
  );
}
```

   - Where does **id** come from?

Task 1:

URL parameters

- Using the parameter `:id`, the string could be automatically captured for use with the `useParams()` hook
- This is especially useful for ***pattern matching*** in URL
- *See: https://reactrouter.com/en/main/start/concepts#matching*
- *Read more here: https://medium.com/better-programming/using-url-parameters-and-query-strings-with-react-router-fffdcea7a8e9*

Task 2:

No-match route

- FileC
- Wrong Link

No match for /wrong

- Traditionally, a web server would return status **404** with an error page to a URL not found on the server
- We can also do it here

1. Add a wrong ***to*** URL with **`Link`**
2. Add a new **`Route`** at the end (*why?*) of the list

```
<Route path="*" element={<NoMatch/>} />
```

3. Set up a new **`NoMatch`** component

```
function NoMatch() {
  let location = useLocation();
  return (
    <div>
      <h3>
  No match for <code>{location.pathname}</code>
      </h3>
    </div>
  );
}
```

- The **`useLocation()`** hook tells us what URL was bringing to this page

Functional vs class components

- Our "original" components were written in classes
- New components in the lab today are written in functions
- What is better?
  - ***Classes***: more traditional way to understand objects, clear use of props/states
  - ***Functions***: cleaner code, shifting to the use of hooks
- *Lots of tutorials* on both of the two
- Either is fine, or even a mix of both
- Learn more about hooks of React Router:
  - See under "Hooks" on *https://reactrouter.com/en/main/start/tutorial*

Submission

- No submission is needed for labs
- What you have done could be useful for your further exploration or the upcoming assignment
- **Please keep your own files safely**