Introduction

Welcome to Module 7! This week, we will learn about building user interfaces for web services using React. React is a free and open-source front-end JavaScript library for building user interfaces based on UI components. React is maintained by Meta (formerly Facebook) and a community of individual developers. React can be used for single-page, mobile or server-rendered applications. React is only concerned with state management and rendering that state to the DOM. React does not attempt to provide a complete application library. React is specifically for building UIs. This allows React to consume data from different middleware providers.

Topics and Learning Objectives

Topics

- React user interfaces
- User Interfaces with JavaScript and React
- React used in creating single page web applications
- React components
- Node.js, a development environment for JavaScript web applications

Learning Objectives

By the end of the module, you should be able to:

1. Explain the difference between a server rendered and a client rendered application.
2. Examine use cases for a Single Page Application (SPA).
3. Describe JavaScript, React.js, and the related ecosystem.
4. Create a front end application with JavaScript (React.js).
5. Build React components.
6. Run a JavaScript web application on a local Linux server.

Readings & Resources

Reading

Required

- React Documentation (2022)

- React Documentation (2022)
  - Create a new React app
  - Thinking In React
- Facebook. (2022). Create React app – Getting started.
- Github. (2022). Create React app.

React

React Intro

- React is declarative
  - Developers design views for each state of an application and React updates and renders components when data changes.
  - Declarative views make your code more predictable and easier to debug.
- React is component-based
  - React code is made of entities called components.
  - Components are reusable and must be formed in the SRC folder following Pascal Case as its naming convention (capitalize camelCase).
  - Components can be rendered to a particular element in DOM. When rendering components, values can be passed between components through “props”.
- React is learned once and written anywhere (anywhere is any web project)
  - React renders a client. React can also render on server using Node and power mobile apps using React Native.

Advantages of React

- In React you can create components that you need. These components may be reused.
- React is very popular. It is used by many companies such as Apple, Netflix, etc.
- React can be used to develop rich UIs for desktop and mobile applications.
- React is easy to debug and test. The code is written in Javascript rather than HTML.

Disadvantages of React

- React code is written in JSX. This may cause some confusion, since most frameworks prefer keeping HTML separate from JavaScript code.
- File size for ReactJS can be large.

React (ReactJS) vs ReactNative

- React, formerly named ReactJS, is a library for creating UIs in web pages.
- ReactNative enables native Android and iOS development with React and also native platform capabilities provided by the OS.
- One of the largest uses of ReactNative is for developing virtual reality applications for Oculus, now Meta VR visors.
- ReactNative is available at on the ReactNative website.

React Components

- In React, logic is coupled with UI logic. This includes event handling, how state changes or evolves over time and how data is prepared for display.
- Instead of artificially separating technologies through markup and placing logic in separate files, React separates responsibilities with loosely coupled units called “components”.
- React components implement a renderer() method that takes input and return what to display.
- Components may be coded in JSX.
- Class-based components are declared using ES6 classes and functional components are declared with a function that returns JSX code.

JSX

- JSX is a syntax extension to JavaScript.
- JSX is used with React to describe what the UI should look like.
- JSX looks like a template language. Unlike template languages, JSX is powered by JavaScript. This means that JSX is compiled to JavaScript.
- JSX produces React “elements”. React elements can be rendered to DOM.
- React does not require JSX, but it is useful as a visual aid when working with UI inside JavaScript source code.
- JSX also allows React to show useful error and warning messages.

DOM and Virtual DOM

- VirtualDOM is a lightweight copy of the actual DOM. This copy exists in memory.

- React creates an in-memory data-structure cache (VirtualDOM), computes the resulting differences (between UI and state data) and then updates the browser's displayed DOM. This process is called reconciliation.
- Reconciliation allows the developer to code as if the entire page is rendered on each change, while React libraries only render subcomponents that change.
- The selective rendering provides a performance increase. It saves effort of recalculating CSS styles, layout for the page and rendering for the entire page.

HTML and Beyond

- React is an architecture that can be extended from the client to the server.
- The basic architecture of React applies beyond rendering HTML in the browser.
- React can be combined with server technologies to render content retrieved over the Internet on a client's computer. For example, Netflix and PayPal use universal loading to render identical HTML on both the server and client.

React Hooks

- Hooks are functions that let developers "hook into" React state and lifecycle features from function components.
- Note that hooks do not work inside classes.
- Hooks allow you to use React without classes.
- React provides a few built in hooks:
    - usesState
    - usesContext
    - useReducer
    - useMemo
    - useEffect

Rules for Hooks

- Hooks have rules with regards to their usage.
    - Hooks should only be called at the top level. Do not call hooks inside loops or if statements.
    - Hooks should be called from React function components and custom hooks. Do not call hooks from normal functions or class components.
- The above rules are recommended and can not be enforced at runtime. Tools such as linters can be configured to detect mistakes at development time.

Video

Please watch the following video on Creating a React App. This video exhibits the usage of Node.js tools to create a React App template. Once the React App is created you can run it using a Node.js server.

Creating a React App
TMU Video | Duration: 07:56

Coding a React Application

- The code below is your first ReactJS application. It can execute inside a web browser. Furthermore, the code must be place in a file named index.js.

```
import React from 'react';
import ReactDOM from 'react-dom';
var name = "Learner";
var element = <h1>Hello, { name }.Welcome to ReactJS.< /h1>;
ReactDOM.render(
     element,
```

```
        document.getElementById("root")
);
```

- How can the above code execute?
- The above code requires an HTML page to update. Below is an HTML page. Place the code in a file named index.html. Note the div with the id of “root”. This id is also used in index.js.

```
<!DOCTYPE html>
<html>
  <head>
     <meta charset="UTF-8" />
     <title>ReactJS Demo</title>
  </head>
  <body>
     <div id = "root"></div>
   </body>
</html>
```

Obtain React from CDN

- To use ReactJS in the web page from the previous page we must first obtain a ReactJS library. It can be obtained from a CDN.
- CDN links can be obtained from https://reactjs.org/docs/cdn-links.html .
- The following can be used for development:

```
<script crossorigin
src="https://unpkg.com/react@18/umd/react.development.js">
</script>
<script crossorigin src="https://unpkg.com/react-
dom@18/umd/react-dom.development.js"></script>
```

Node.js

- Node.js is an open-source, cross-platform JavaScript runtime environment that executes JavaScript code outside a web browser.

- Node.js is used for creating React applications.
- Node.js provides tools such as a Node server and NPM.
- Node is a simple server that can serve our applications.
- NPN is a package manager used to install libraries such as React
- When developing React applications we should have Node.js installed.

Install React Libraries

- Create a directory/folder named “FirstReact”. Change your working directory/folder to “FirstReact”.
- Inside FirstReact create a Node application.
- Run the following commands from terminal

```
a. npm install create-react-app
b. npx create-react-app firstreactapp
```

- The above commands install required React libraries that will enable development.

Compile JSX into JavaScript

- JSX code is compiled by configuring your Node application. There is a file named “package.json”. This file was created when you created your Node application.

```
"scripts": {
     "start": "react-scripts start"
  }
```

- Your “package.json” file should look like the code below. You may have some differences in version numbers.

```
{
  "name": "FirstReact",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
```

```
      "start": "react-scripts start"
   },
   "author": "",
   "license": "",
   "devDependencies": {
      "react": "^16.9.0",
      "react-dom": "^16.9.0",
      "react-scripts": "^3.1.1"
   }
}
```

Coding a React Application, cont.

Running Your React Application

- To run FirstReact application you have to start the Node server.
  - a. npm run start
- Once the Node server has started you can visit the page that it server. This may be done with a web browser. Simply visit the URL http://localhost:3000/.
- What do you see on the web page? Note this content comes from index.js.
- In index.js, if you change the name variable with your name, what do you see after visiting http://localhost:3000/?

Using JSX in Our React Application

- JSX can be used in a React application.
- Create a file named “test.jsx”. The code is exhibited below.

```
import React from 'react';
import ReactDOM from 'react-dom’;

var name = "Learner";

var element = <h1>Hello, { name }.Welcome to ReactJS.< /h1>;

export default element
```

- The above code represents a JSX file named test.jsx. How can we use this file?
- Modify index.js to use test.jsx. The new index.js file is shown below.

```
import React from 'react';
import ReactDOM from 'react-dom';
import element from './test.jsx';

ReactDOM.render(
     element,
     document.getElementById("root")
);
```

Run the React Application

- Run FirstReact application and visit http://localhost:3000/. What do you observe?
- How is test.jsx executed? Is it invoked directly or indirectly?

Components as Functions

- We can convert the code in test.jsx into a functional component. Note we are using props to pass values to a functional component.

```
import React from 'react';
import ReactDOM from 'react-dom';

function ComponentName(props) {
     return <h1>Hello, { props.value }.Welcome to ReactJS.<
/h1>;
}

var name = "Learner";
const ComponentName_comp = <ComponentName value={name} />
export default ComponentName_comp
```

Components as Classes

- We can convert test.jsx into a class component. Note we are using props to pass values to the constructor of the class component.

```
class ComponentName extends React.Component {
  constructor(props) {
     super(props);
  }
  render() {
     return <h1>Hello, { this.props.value }.Welcome to ReactJS.<
/h1>;
  }
}

const ComponentName_comp = <ComponentName value={name} />
export default ComponentName_comp
```

Run the React Application

- Run FirstReact application and visit http://localhost:3000/. What do you observe?
- How is test.jsx executed? Is it invoked directly or indirectly?
- Which do you like, functional components or class components? State why?

Video

Please watch the following video on React App Deployment. This video exhibits the usage of deploying a React App to a Python middleware. You will compile your react app and generate JavaScript that can be interpreted by a web browser. This way you can use your Python middleware without the use of a Node.js server.

React App Deployment
TMU Video

Summary

React enables you to create single page user interfaces for web applications. This is achieved by turning your web applications into components. This means that react components create html, javascript, css fragments that populate a web page. React is a simpler library when compared to competing frameworks such as Angular. React only focuses on user interface components of web pages. React focuses only enabling you to create components and it excels in this task. It is this fact that over 80% of web applications use React as the framework powering their user interface.

Assessments

Each week, you'll find here reminders of assignments or labs that you should be working on. For Week 7:

- Lab 7: Creating a React Web Application using Node.js (5% of the final grade) is due in 7 days, i.e., the end of day Friday of Week 7.

Reminder

You have a weekly Zoom session. The date/timing for your Zoom session can be found in the Course Schedule in the Course Outline. The link to the Zoom session will be provided separately to you by your instructor. The Zoom session will consist of 20–30 minutes of lecture, followed by

questions and answers (Q&A), followed by a lab period. You may ask questions about the previous or current week's content or labs.

Please bring any questions to the weekly Zoom session. However, if you have any questions during the week outside of the Zoom session, you may post them in the Discussion Board. Your instructor may respond in the Discussion board or during the weekly Zoom session.

Click-n-reveal: **Where is the Discussion board?**

Click on the "Communication" area at the top main course menu, and select Discussions from the dropdown menu.

References

None for this week.