SUMMARY OF COURSERA COURSE

FRONT-END WEB DEVELOPMENT WITH REACT

RATINGS: /5

WEEK 1 – INTRODUCTION TO REACT

UNIT 1: Welcome to Front-End Web Development with React

- You should have good working knowledge of HTML, CSS and JS
- He views Web Design and Web Development/Building/Deployment as two different entities
- JS framework approach to web development
- React library and its ecosystem; Reactstrap, Redux, Fetch
- "Don't rush the course, make sure you understand what is going on"

UNIT 2: How to use the Learning Resources

- Modules have a common overarching theme, mapped to a calendar week
- "One fourth from the teacher, one fourth from own intelligence, one fourth from classmates, and one fourth only with time"

UNIT 3: Additional Resources

- React site: https://reactjs.org/

UNIT 4: Full Stack Web Development – The Big Picture – Objectives and Outcomes

- Understand what is meant by full stack in the context of web development
- Distinguish between front-end, back-end, and full stack web development
- Understand the position of this course in the context of this specialization

UNIT 5: What is Full Stack Web Development?

- Front-end = Client-side: HTML, CSS and JS
- Back-end = Server-side: various technologies
- Three tier architecture

UNIT 6: Additional Resources

- Downloaded "FSWD-BigPicture.pdf"

UNIT 7: Setting up your Development Environment – Git and Node – Objectives and Outcomes

- Set up a Git repository and perform basic Git operations
- Set up and use online Git repositories
- Use Node-based modules to perform basic operations

UNIT 8: Setting up your Development Environment

UNIT 9: Setting up Git - Exercise

- Version control: software tool that enable the management of changes to source code

UNIT 10: Setting up Git – Instructions

UNIT 11: Basic Git Commands – Exrcise

- “git checkout”: Checkout the file from an older commit

UNIT 12: Basic Git Commands – Instructions

UNIT 13: Online Git Repositories

- Local repo can only be linked to one remote repo
- “git clone”

UNIT 14: Online Git Repositories – Instructions

UNIT 15: Node.js and NPM

- NPM: manages ecosystem of node modules/ packages

UNIT 16: Setting up Node.js and NPM - Exercise

UNIT 17: Setting up Node.js and NPM – Instructions

UNIT 18: Basics of Node.js and NPM

- “package.json”: serves as a documentation
- “npm init”

UNIT 19: Basics of Node.js and NPM

UNIT 20: Additional Resources

- Downloaded “NodeJS.pdf”

UNIT 21: Introduction to React – Objectives and Outcomes

- Get a basic overview of JS frameworks and libraries
- Understand the architecture of a React application
- Scaffold out a starter React application using “create-react-app”, the command line tool

UNIT 22: Front-end JS Frameworks and Libraries Overview

- Why JS Libraries/ Frameworks: it helps with the complexity of manipulating and data updates, Well defined application architectures
- Software Library: collection of implementations of behavior with a well-defined interface by which the behavior is invoked

- Software Framework: abstraction in which software provides generic functionality that can be selectively changed by additional user-written code
- "Library vs Frameworks"
- Hollywood Principle: "Don't call us, we will call you" (the frameworks calls upon your code for custimization)
- "Imperative vs Declarative Programming"
- Single Page Application, MVC, MVVM

UNIT 23: Introduction to React

- React: a JS library for building user interfaces, it uses a declarative approach, component-based, technology stack agnostic (plays well with various tech stacks)
- Designed by Jordan Walke, Open Source in 2013
- "One way data flow"

UNIT 24: Getting Started with React – Exercise

- "yarn" similar to npm, but it is popular for React
- Yarn documentation: https://yarnpkg.com/getting-started/install
- "npm install –g create-react-app" > "npx create-react-app confusion"
- "yarn start"

UNIT 25: Getting Started with React – Instructions

UNIT 26: React App Overview

- React Element: the smallest building blocks of React apps, plain JS objects that are cheap to create, "const element = <h1 className = ***>Welcome to React</h1>"
- Components are made of elements, "class App extends Compnent {…}"
- Use "className" instead of "class"
- "App.js" > "Index.js" > "Index.html"
- "ReactDOM": used to render components

UNIT 27: Introduction to JSX

- JSX: syntactic extension to JS, shorthand notation to represent function calls that evaluate to JS objects
- The template and JS code are combined in React (rendering logic and UI logic are combined)
- Embedded expressions in JSX

UNIT 28: Configuring your React Application - Exercise

- Configure our React application to use Bootstrap using "Reactstrap and Bootstrap"
- "yarn add Bootstrap@4.0.0"
- "yarn add reactstrap@5.0.0 react-popper@0.9.2"
- Import bootstrap into the "index.js" file

UNIT 29: Configuring your React Application – Instructions

UNIT 30: Additional Resources

- Downloaded “JavaScript-Frameworks.pdf”, “Intro-React.pdf”, “React-App-Overview.pdf”, “Intro-JSX.pdf”
- Reactstrap Documentations: https://reactstrap.github.io/

UNIT 31: React Components: Objectives and Outcomes

- Create a React component
- Construct the React component code and the view for your component using JSX and JavaScript

UNIT 32: React Components

- React component: returns a set of React elements that should appear on the screen
- Components enable you to split your UI into independent, reusable pieces
- Components also accept inputs
- User defined component names must always start with a capital letter
- Tags starting with lowercase letters are treated as DOM tags

UNIT 33: React Components Part 1 – Exercise

- Download “images.zip”, unzip and place in public > assets
- You can define React component with regular JS Function or with ES6 classes

UNIT 34: React Components Part 1 – Instructions

UNIT 35: React Components: State and Props

- Each component can store its own local info in its “state”
- Only class components can have local state
- State declared within the constructor
- “this.setState({})”
- Never directly manipulate state values
- Handling events is similar to the way you handle events on DOM elements

UNIT 36: React Components Part 2 – Exercise

- Information to a child component from a parent component

UNIT 37: React Components Part 2 – Instruction

UNIT 38: React Components: Lifecycle Methods Part 1

- Every “React class component” has a Lifecycle
- Mounting > Updating > Unmounting
- Several lifecycle methods available in each stage

- “reactdevtools” extension for chrome, useful for react application

UNIT 39: Additional Resources

- Downloaded “Component-Part1.pdf”, “Component-Part2.pdf”, “Lifecycle-Methods.pdf”

UNIT 40: Assignment 1 Requirements – React Components

WEEK 2 – REACT ROUTER AND SINGLE PAGE APPLICATIONS

UNIT 1: React Component Types – Objectives and Outcomes

- Identify the salient features and uses for the various types of components
- Create presentational, container and functional components in your React application

UNIT 2: Presentational and Container Components

- Classification on Components based on how they are used
- Presentational Components: mainly concerned with rendering the “view”
- Container Components: responsible for making things work, make use of presentational components, maintain state and communicate with data sources

UNIT 3: Presentational and Container Components – Exercise

- “filter” array method

UNIT 4: Presentational and Container Components – Instructions

UNIT 5: React Components: Lifecycle Methods Part 2

- Lifecycle methods

UNIT 6: Functional Components

- Class components, we can define local state with class components, it allows lifecycle hooks
- Functional Components: simplest way to define React Component, where you don’t need local state and lifecycle hooks, receives props object as a parameter

UNIT 7: Functional Components – Exercise

UNIT 8: Functional Components – Instructions

UNIT 9: Additional Resources

- Downloaded “Component-Types-Part1.pdf”,
- “Component-Types-Part2.pdf”

UNIT 10: React Router – Objectives and Outcomes

- Set up the router module to enable navigation among multiple component views
- Set up the routes to enable the navigation

UNIT 11: React Virtual DOM

- Browser DOM is a browser object
- Virtual DOM is a React Object; a lightweight representation of the Browser DOM
- “Diffing Algorithm”: will detect those nodes that are changed
- “React Fiber”: new reconciliation algorithm in React 16, incremental rendering

UNIT 12: Header and Footer – Exercise

- “React Fragment”

UNIT 13: Header and Footer – Instructions

UNIT 14: React Router

- A typical React application may consist of multiple pages
- React Router: collection of navigational components, enables navigation among views
- “Web App Routing”: install react-router-dom, Router components
- “<Link>” will render as <a> in the HTML, “<NavLink>” attaches the “active” class to the link

UNIT 15: React Router – Exercise

UNIT 16: React Router – Instructions

UNIT 17: Additional Resources

- Downloaded “VirtualDOM.pdf”, “React-Router.pdf”

UNIT 18: Single Page Applications – Objectives and Outcomes

- Design SPA using React
- Use the React Router to construct SPA

UNIT 19: Single Page Applications

- SPA help to save round trip requests and response to the server, you don’t need to reload the entire page, most resources are retrieved with a single page load

UNIT 20: Single Page Applications Part 1 – Exercise

UNIT 21: Single Page Applications Part 1 – Instructions

UNIT 22: React Router: Parameters

- Paths can also carry parameter values, “match Object”

UNIT 23: Single Page Application Part 2 – Exercise

UNIT 24: Single Page Applications Part 2 – Instructions

UNIT 25: Additional Resources

- Downloaded “Single-Page-Apps.pdf”, “React-Router-Parameters.pdf”

UNIT 26: Assignment 2

- Semi completed “AboutUs” Component
- “Media” reactstrap component

WEEK 3 – REACT FORMS, FLOW ARCHITECTURE AND INTRODUCTION TO REDUX

UNIT 1: Controlled Forms – Objectives and Outcomes

- Design a controlled form in your React application

UNIT 2: Controlled Forms

- Forms are a standard ay of seeking user input
- HTML “form” tag and elements
- “Controlled Components”: make the React component control the form that it renders

UNIT 3: Controlled Forms – Exercise

UNIT 4: Controlled Forms – Instructions

UNIT 5: Controlled Form Validation – Exercise

UNIT 6: Controlled Form Validation – Instructions

UNIT 7: Additional Resources

- Downloaded “Controlled-Components-Forms.pdf”

UNIT 8: Uncontrolled Forms – Objectives and Outcomes

- Create uncontrolled forms through uncontrolled components in React
- Handle the form submission in the React application

UNIT 9: Uncontrolled Components

- Ideally you should implement forms within controlled components but sometimes…
- Uncontrolled components make it easier to integrate React with non-React code

UNIT 10: Uncontrolled Forms – Exercise

UNIT 11: Uncontrolled Forms – Instructions

UNIT 12: Additional Resources

- Downloaded “Uncontrolled-Components-Forms.pdf”

UNIT 13: Introduction to Redux – Objectives and Outcomes

- Install and configure Redux in your application
- Enable your React app to make use of Redux

UNIT 14: The Model-View-Controller Framework

- Design patterns: well documented solution to a recurring problem
- Model: manages the behavior and data of the application domain
- View: renders the model into a form suitable for interaction, typically a UI element
- Controller: receives user input and initiates a response by making calls on model objects

UNIT 15; The Flux Architecture

- Flux Architecture: unidirectional data flow, “Action > Dispatcher > Store > View”

UNIT 16: Introduction to Redux

- Redux: realization of the Flux architecture, predictable state container for JS apps
- Principles of Redux: single source of truth, state is read-only (only getters, no setters), changes are made with pure functions

UNIT 17: Introduction to Redux – Exercise

UNIT 18: Introduction to Redux – Instructions

UNIT 19: Additional Resources

- Downloaded “MVC.pdf”, “Flux-Arch.pdf”, “Intro-Redux.pdf”

UNIT 20: React Redux Form – Objectives and Outcomes

- Configure and use react-redux-form to create Controlled forms
- Store the form state in the Redux store

UNIT 21: React Redux Forms

- “react-redux-form”: a versatile, fast and intuitive library for creating complex and performant forms in React and Redux

UNIT 22: React Redux Forms – Exercise

UNIT 23: React Redux Forms – Instructions

UNIT 24: React Redux Form Validation – Exercise

UNIT 25: React Redux Form Validation – Instructions

UNIT 26: Additional Resources

- Downloaded "React-Redux-Form.pdf"

UNIT 27: Assignment 3

- 3 Tasks

WEEK 4 – MORE REDUX AND CLIENT-SERVER COMMUNICATION

UNIT 1: Redux Actions – Objectives and Outcomes

- Define Redux actions
- Create action creator functions that return action objects
- Split the reducer function into multiple simpler functions and combine the reducer functions

UNIT 2: Redux Actions

- Actions: payloads of info that send data from your application to the store
- Action Creators: functions that create actions, return action object
- Reducers should be able to take the previous state and action and return next state

UNIT 3: Combining Reducers – Exercise

UNIT 4: Combining Reducers – Instructions

UNIT 5: Redux Actions – Exercise

UNIT 6: Redux Actions – Instruction

UNIT 7: Additional Resources

- Downloaded "Redux-Actions.pdf"

UNIT 8: Redux Thunk – Objectives and Outcomes

- Use Redux Thunk middleware to return a function instead of an action
- Use a logger middleware to print a log of actions initiated on the Redux store

UNIT 9: Redux Thunk

- Redux Middleware: provides the capability to run code after an action is dispatched, but before it reaches the reducer, it forms a pipeline that wraps around the dispatch()
- Thunk in programming is a subroutine used to inject an additional calculation to a subroutine
- Redux Thunk: middleware that allows you to write action creators that return a function instead of an action

- Redux Saga??

UNIT 10: Redux Thunk – Exercise

- Install “redux-thunk”, “redux-logger”
- Redux logger: it logs that state changes in the console

UNIT 11: Redux Thunk – Instructions

UNIT 12: React-Redux-Form Revisited – Exercise

- React redux form also logs to the console

UNIT 13: React-Redux-Form Revisited – Instructions

UNIT 14: Additional Resources

- Downloaded “Redux-Thunk.pdf”

UNIT 15: Client-Server Communication – Objectives and Outcomes

- Set up a simple server that makes data available for clients
- Access the data from the server using a browser
- Use the json-server as a simple static web server

UNIT 16: Networking Essentials

- Web applications are not stand-alone, many of them have a “Cloud” backend (Amazon, Heroku)
- HTTP, URL, JSON, SOAP, REST, XML, GET, POST, PUT
- You need to write applications recognizing the asynchronous nature of communication
- HTTP: Hypertext Transfer Protocol, a client-server communications protocol
- HTTP Response Codes
- JSON: lightweight data interchange format, language independent, self-describing, structured as a collection of name/value pairs

UNIT 17: Brief Representational State Transfer (REST)

- Web Services: a system designed to support interoperability of systems connected over a network
- SOAP (Simple Object Access Protocol, xml based) and REST (xml or Json based) based web services
- REST API endpoints
- REST: use HTTP methods explicitly, stateless, expose directory structure-like URLs, transfer using xml, JSON or both
- REST: Nouns(resources), Verbs (http methods), Representation (xml, json)
- REST uses URI to identify resources;”http://www.conFusion.food/dishes/***”
- REST Verbs: GET = READ, POST = CREATE, PUT = UPDATE, DELETE = DELETE

UNIT 18: Setting up a Server using json-server – Exercise

- “json-server” node module
- “npm install json-server –g” > “json-server --watch db.json -p 3001 -d 2000”

UNIT 19: Setting up a Server using json-server – Instructions

UNIT 20: Additional Resources

- Downloaded “Networking-Essentials.pdf”, “REST.pdf”

UNIT 21: Fetch – Objectives and Outcomes

- Install Fetch in your React application
- Use Fetch to communicate from your React application with a REST API server

UNIT 22: Promises

- Promise is a mechanism that supports asynchronous computation, proxy for a values not necessarily known when the promise is created
- It represents a values that may be available now/ future/ never
- Promises help to solve the callback hell, can be chained

UNIT 23: Fetch

- Fetch API is a modern replacement for XMLHttpRequest
- Other libraries like “axios”
- Cross-Fetch: provides support for Fetch both in Node application and Browsers, uses polyfill in browsers that do not have fetch support
- Fetch alternatives: Axios (gives a higher level API than Fetch API), Superagent

UNIT 24: Fetch from Server - Exercise

- “json-server –watch db.json –d 2000 –p 3001”
- Leave json-server up and running to enable us fetch resources from it
- “yarn add cross-fetch”

UNIT 25: Fetch from Server – Instructions

UNIT 26: Fetch Handling Errors – Exercise

UNIT 27: Fetch Handling Errors – Instructions

UNIT 28: Fetch Post Comment - Exercise

- “POST” operation using Fetch

UNIT 29: Fetch Post Comment – Instructions

UNIT 30: Additional Resources

- Downloaded “Promises.pdf”, “Fetch.pdf”

UNIT 31: React Animations – Objectives and Outcomes

- Add subtle animations using he react-transition-group
- Add additional component animations using react-animation-components

UNIT 32: React Animations

- Plenty of npm packages that support react animations
- We will explore “react-transition-group”, “react-animation-components”
- React Transition Group: designd with animation in mind
- Transition state: entering, entered, exiting, exited
- “in prop”, “timeout”, “classNames”
- <Transition>
- React Animation Component: a set of react components implemented using react-transition-group

UNIT 33: React Animations – Exercise

- Animations adding a little bit of spice to out UI
- Install react-transition-group

UNIT 34: React Animations – Instructions

UNIT 35: React Animation Components – Exercise

- Install react-animation-components
- Install prop-types
- Try out other options in “react-animation-components”

UNIT 36: React Animation Components – Instructions

UNIT 37: Additional Resources

- Downloaded “Animations.pdf”

UNIT 38: Assignment 4

- 3 Tasks

UNIT 39: Building and Deployment – Objective and Outcomes

- Understand the Webpack way of packaging applications into bundles
- Use react-scripts to build a distribution folder with your React application bundles using Webpack

UNIT 40: Introduction to Webpack

- Webpack is a module bundle for modern JS applications, similar to Grunt and Gulp
- Configuration in webpack.config.js
- Webpack examines your application source code for import statements
- Webpack concepts: Entry, Output, Loaders, Plugins
- Check out Webpack documentation for more in depth knowledge

UNIT 41: Building and Deploying the React Application – Exercise

- “npm run build”
- “never leave comments in the return/ render portion of your code or better wrap them in brackets or /* */”

UNIT 42: Building and Deploying the React Application – Instructions

UNIT 43: Additional Resources

- Downloaded “Webpack.pdf”