EE461L Lab4

Front-end Technologies

Introduction

The purpose of this lab is to help guide you through self-learning different front-end languages. By the end of this lab, we hope that you will have a working knowledge of HTML and CSS, as well as React, a JavaScript framework for developing front-end applications. Our goal is that through this lab, you will be able to start working towards building the front-end (client application) for the semester project.

Please note you have one week from the end of your lab section to finish this assignment. This is the most challenging assignment we have given so far, largely due to the amount of self-learning required to create the deliverables. With that being said, it is quite common to have to learn new languages and techniques on your own in industry. Feel free to use any resources you want to help complete the assignment, and share them on Piazza for your classmates.

Part 0: Dr Samant's Lectures on Front-end and React

Please refer to the following material on Canvas under the Module titled: Module5: Web Development

1. Slides on class lecture on Web Development
2. Video recording of class lecture on Web Development (link)
3. Reference slides on Basic Web Programming and JavaScript
4. Slides on React.js

Part 1: Basic HTML and CSS

Before we can dive into React, we must first understand the grandfather of front-end languages, HTML. For this section, we have attached three references: one for practical knowledge of HTML, one for CSS (used for organized visual formatting of HTML), and a video that should help explain how to combine HTML and CSS to create a basic, static front-end application.

Please read through the deliverable section below, then go back and read/watch the reference material to piece together how to complete the deliverable. Again, you can find your own resources if these are not sufficient. Feel free to share any additional sources on Piazza.

Reference Material:

https://www.w3schools.com/html/default.asp https://www.w3schools.com/css/default.asp

Intro to HTML & CSS - Tutorial

Deliverable:

1. Create a file called foo.html. Create a populated table, paragraph, header, list, and link to a website in the body of foo.html in an organized manner, ideally representing some sort of home page. Include at least one class attribute and one id attribute to use for CSS styling.
2. Create a file called bar.css. Put some CSS definitions in it that refer to the class and id attributes above: https://www.w3schools.com/cssref/default.asp
3. Now find a front-end framework (https://www.w3schools.com/w3css/w3css_templates.asp, Bootstrap, etc.) and write a different html file (name it <EID>.html) that uses this framework to create a basic homepage. In terms of requirements, try to emulate the components we ask for in step 1.

Submit your source code (foo.html, bar.css, and <EID>.html as the root) in a ZIP file: <EID>-Lab4-P1.zip

Part 2: React JavaScript

Now that we have a basic understanding of HTML and CSS, let's move into the 21st century with JavaScript! You probably stumbled upon embedded JavaScript while working on Part 1 (anything with the <script>xyz</script> tags). Combining HTML, CSS, and JavaScript is referred to as DHTML, and was a common way for front-end developers to incorporate OOP-esque practices into front-end code. You could embed scripts to define what certain buttons do, process information (using something like JQuery), create Boolean/conditional logic, loops, etc.

Eventually, the development community realized they could better structure/organize/embed logic into their code by developing OOP-esque JavaScript frameworks to be the core language on the front-end. This means we transitioned from HTML/CSS being embedded with JavaScript (or sometimes Java in the case of JSP files), to embedding HTML/CSS into core JavaScript files. That is the fundamental difference between DHTML/Bootstrap and React/Angular/Vue JavaScript. Most technology companies now use a JavaScript framework for developing any front-end, whether it be a website, mobile application, mobile-friendly website, etc.

The reason we are teaching React is because of the most popular JS frameworks, React is considered to have the least steep learning curve. In order to complete this section, we want you to read the reference material on React, then follow the installation instructions to set React up to run on your local machine (click the "Optional: Instructions for following along locally using your preferred text editor" tab to start the setup, then go from there). From there, we want you to follow along with the tutorial in the deliverable section, and submit another zip file containing that final source code of your Tic-Tac-Toe game.

Reference Material:

https://reactjs.org/docs/hello-world.html

Installation (Please set up React locally, do not use the online text editors provided):

https://reactjs.org/tutorial/tutorial.html#setup-option-2-local-development-environment

Deliverable:

https://reactjs.org/tutorial/tutorial.html

Submit your source code (the entire project repository npx created) in a ZIP file: <EID>-Lab4-P2.zip

Part 3: Build your own React Front-end

Now comes the challenge. Given your newly acquired knowledge on front-end languages, use React to build the homepage of your own personal website (a website showing us some basic things about you and your interests). You can model the look/feel off the React homepage, your vision of your semester project website (e.g. https://powderwireless.net/), or any other website you want. In terms of grading, we are looking for a site that demonstrates a basic to intermediate understanding of React. Your website does NOT need multiple pages or to be deployed to the Cloud/have a backend/have a database. With that said, it should have some dynamic applications, such as a carousel of images (you would need to include this in your submission), buttons that perform interesting actions, drop-down tables, etc.

Reference Material for UI design and Routing:

[React Website Tutorial - Beginner React JS Project Fully Responsive]

[https://material-ui.com/getting-started/installation/] (Popular source of pre-made components for React.js, including pre-made buttons, tables, lists, and cards, with documentation as well.)

[https://reactrouter.com/web/guides/quick-start] (Documentation for react-router)

Deliverable:

- Create a react app using create-react-app
- Create a homepage with at least two routes that route to two different components. The links must be accessible from your homepage
- One component must be implemented as a function component
    - You must use at least two hooks to modify 2 state variables
- Once component must be implemented as a traditional React.js component
    - You must have at least two state variables
- Import and use at least one icon and one component from any popular component library (such as MaterialUI), or spend time using CSS to create good-quality components yourself
- You must pass in props, at least once, to either one of the components you designed or any external components that you imported.

Submit your source code (the entire project repository npx created) in a ZIP file: <EID>-Lab4-P3.zip

Rubric

Please note that 25% of the points for each part are allocated points for following proper coding/commenting practices.

Task	**Points**
Part 1	2
Part 2	2
Part 3	2