ITMD 465/565

Lab React

Introduction

In this lab we will be experimenting with some additional React concepts like lifecycle events, components, state and props, JSX, ES6 Modules, and webpack.

Objective

The objective is to build a small React application from a few components using JSX. You will also be retrieving your data from a public API. This lab files will be submitted as a zip file containing your project folder to the blackboard assignment. Do not include the node_modules folder.

Requirements/Process

- You must submit this Lab as a zip file containing your project folder to the blackboard assignment. Do not include the node_modules folder.
- I have supplied a preconfigured webpack/babel node project called dogimage, just like the one we setup in class. The node_modules folder is not included so you will need to use npm to install the module dependencies.
- Please change the author key in the package.json file to have your name in it.
- Make sure you are looking at your browser's dev tools console for errors if you are having issues.
- You will need to use the following API: https://dog.ceo/dog-api/
- Read the documentation at the Dog API site to find out the AJAX request URLs you will need to use
- Use supplied project zip to start from. It is all setup and configured for you. You just need to install dependencies with npm install.
- The starter project includes a package.json file with all the dependencies you need preconfigured and some npm webpack scripts.
    - npm run start – Starts webpack dev server
    - npm run build – Compiles to dist directory in production mode
    - npm run dev – Compiles to dist directory in development mode
- You are going to make a simple application that uses React and ES6 modules to show you a random image of a dog from an API. You will be able to click a button to load a new random image. Make sure it loads a random image when the app loads and also when the button is clicked.
- You need to use component state to hold the image URL that is currently being viewed.
- Pick 1 breed of dog from the API and code your app so it looks like the sample. Choose your own choice, not my example.

- The state that holds the image URL should be passed to a child component as a property and that component should render the img tag using the URL in the props. Please create a separate component called DogImage that gets the URL prop passed to it. Then have this component render out an img tag with the correct attributes.
- I showed a demo in class, review the class demo if needed.
- 2 Components at minimum: App, DogImage
- App Component
    - Skeleton file App.js is already in the src/components directory
    - Keep your apps state in this component
    - When the app loads this component should do an ajax call to get a random dog image from the dog.ceo API. Do this in componentDidMount.
    - The App component should render the HTML like in the video:
        - <h1> tag with "Breed Dog Image Generator" using your chosen breed.
        - <h2> tag with your name, email address, and class number "Brian Bailey – bbailey4@iit.edu – ITMD-465"
        - <p> tag saying "Please press the button to generate a new random image."
        - Button tag that when clicked shows a new random image of that breed.
        - <hr> take to separate the top controls from the image
        - The DogImage Component to render the image tag
- DogImage Component
    - Create a new component in its own file in the src/components directory
    - This component should render output as an HTML <img> tag
    - This component should take in two things as props. One is the fully qualified URL to an image file for the src attribute and the other should be the text for the alt attribute.
    - Use the prop with the image url for the src attribute and use a prop with the breed name to fill the alt attribute in the img tag.
- 565 Students look in the below section for additional requirements

Graduate Additional Requirements

If you are involved in **any section of 565** you need to complete the additional requirements listed here.

Add a select list to your app. Choose at least 3 breeds to put in the select list. You may hard code the options in the select list. When someone changes the list, you should see the new breed. If someone clicks the button you should see a new image of the same breed. You will need to keep the breed selection in the component state too. This select list should be another Component in your app. Make sure when the select list is changed it updates the app state for what breed is selected and the app shows a new image.

- App Component
    - Everything above needs to be included

    - Need to add the new DogSelect component to the HTML output of the App component next to the Button.
    - Store the selected breed as state in the App component and use that state to substitute the selected breed name in the H1 tag and also use it to control which breed in the API endpoint is used. You should be able to build the correct API URL from the base URL and this state value with string concatenation.
    - You will need another event handler method in this component to handle when the DogSelect component changes. This would be passed to the DogSelect component as a prop to be used inside the DogSelect component as a onChange handler.
    - Make sure the text in the H1 changes when the breed select control changes and also a new random image loads.
    - Make sure you load a new random image when the select control changes. You should not have to click the button to get a new image after the select changes but if you do hit the button it should load a new image of whatever is selected too.
- DogSelect component
    - Create a new component in its own file in the src/components directory
    - This component should render output as an HTML <select> tag
    - Pick at least 3 breed names to put in the select list. You can pick whichever ones you want and you can code them right in the select options.
    - This component should take in as a prop a function to handle changing the selected breed in the App component's state.
    - This function should be used with the <select> tag's onChange attribute.
    - **Not Required**, but if you want to go further you could fill the select options with a call to the API for the list of breeds in the componentDidMount lifecycle method.

README File

No additional readme files other than the ones described in the lab directions. If you can not get this lab working then you need to submit a comprehensive written readme that explains everything you tried and how you attempted to troubleshoot the problems you were having. This should include any console errors you could not solve or problems with dependencies. You need to include screenshots and document the assignment and your process well. If your app does not run you will be graded on how good/complete your readme is. Please submit it as PDF format.

Due Date / Late Policy

This assignment is due **Saturday November 30, 2019 11:59 PM Chicago Time**. **No Extensions or Exceptions. Assignments will not be accepted after this date, you will get a zero. I can not take regular assignments once finals week begins and regular class week ends.**

Submission Guidelines

You must upload your submission, to the blackboard assignment by the due date. The submission must be in the following format and structure. If you do not submit your assignment exactly as specified, you will receive an immediate 5% deduction.

Submission Format Specification:

Zip your project as your myIITusername_lab_canvas.zip and it should contain your entire project folder. Do not include the node_modules folder. Make sure you compile the finished project using the npm script before submitting. I will be grading the compiled version.

Things you may need to use
Lifecycle events
componentDidMount()
componentDidUpdate()
this.state
this.setState
this.props
AJAX requests, fetch API, axios library