3D Visualization of Data
Tech Review

CS461 - Senior Software Engr Project

Instructor: D. Kevin McGrath
Instructor: Kristen Winters

Fall 2018

Eli Laudi

October 11th, 2018

Abstract

This tech review will focus on the web page interactivity that will be provided to the user in our application. The application itself is a web page that serves up a 3D visualization tool that can take in a CSV file and output a 3D rendering of the data in the file. This 3D visualization will live in a user interface that must provide the user with both feedback and interactivity. The specific sections of the UI this paper will focus on is the file explorer module that will be used to upload files and the interface the user will use to modify the graph.

I. File Management

A common user interaction with a web page is uploading a file. Uploading a file will be the first step a user will perform when using our application. Since this is such a commonly implemented service, there have been many tools and methods created to allow simplistic integration of file exploring in a webapp. This next section will explore three of those solutions and examine how we might use them within our application.

A. FileUpload

The first tech solution is the generic input FileUpload property. The html tag of ¡input /¿ has a type that can be set to the string file. This method piggybacks off of the file explorer native to the users operating system. The input will appear as a button that, when clicked, will allow the user to upload one or more files to the web page. This method will load the file into the web page and allow it to be accessed by id via the value passed into the id attribute on the input tag. When referenced, the element grants access to an array of file objects which have various properties that can be access via the dot operation: lastModified, lastModifiedDate, name, webkitRelativePath, size, and type. These properties enable the consumption of the file for the CSV reader to handle.

B. Formidable

The second tech solution is to utilize the Node js framework in building our web page. This particular method would allow the introduction of a module to be installed via NPM (node package manager). This module is called Formidable. Formidable accepts files through the ¡input /¿ tag much like the generic input method. Where the two start to differ is the handling of the file once it is uploaded. With the Formidable module, the file is uploaded, parsed, and placed in a temporary folder. From that point it provides what is called an IncomingForm object, which can be accessed when the form is uploaded. From there, the Node js File System module, or fs, can be included to manipulate the form by moving it, renaming it, or anything else that needs to be done. For this app, the most important feature would be reading the form, which can be simply done with a call to the parse method of the IncomingForm object.

C. FilePond

The third tech solution is to build our front end using React JS. React is a javascript library for building high speed, often single page, web applications. The advantage of React is that through its use, one gets access to many third party libraries that grant functionality that would otherwise take a large amount of work to create. One of these libraries is called FilePond. FilePond allows highly customizable features for file uploading and managing. Its implementation is very simple and can be done in one line. It supports drag and drop from desktop as well as file browsing using the native file explorer. Once uploaded to the webpage, the file can either be sent to a server or accessed from the web app itself. This would allow us to leave an opening for expanding to include back-end storage of previously uploaded datasets.

II. Graph Settings

An important part of the interactivity of this application is the settings that handle the way the graphical data is rendered. The requirements for this section is to allow the user to select which dataset should correspond to which axis, the possible colors for each dataset, the way in which the data should be displayed (bar, linear, point), and other possible graph settings.

A. HTML form Object

The first tech option for managing the settings for the graph is utilizing the HTML DOM form object. The form objects are created via the tag ¡form /¿. Within the form, there are several ¡input /¿ tags that can take on various properties based on the type that they are given. There can be text fields, checkboxes, multiple choice radio options, dropdowns, etc. These input tags can be accessed via the action supplied in the attributes of the form opening tag. A submit button supplied somewhere in the form will trigger the action with whatever values are within each field at the time of the button press. In the case of our application, this action would trigger a re-render of the 3D graph with the settings supplied by the user.

B. React

The second tech option would be utilizing the React library to maintain the state of the graph options at all times. React provides a library that utilizes components to render bits of HTML onto the DOM. Within each of these components, there is a contextual state which can hold information that is pertinent to the component inside of it. In the case of this application, the state could hold all of the options for the graph within it. The component re-renders every time the state is updated, so the graphs rendering can be dependent on the state of the component that holds it. The state would have to be manipulated via user input. For this, there are several react libraries that can be included to provide simple form functionality. One of these libraries is reactstrap, a barebones, yet fairly popular library that provides a stylized form and functionality associated with it.

C. Redux

The third tech option would be keeping all graph options in something called Redux. Redux is a state container for Javascript apps that can be accessed globally. Redux creates a store at the root of the application that can be included in any children of the root. These children can either consume or modify this store via actions. These actions are dispatched to Redux, which then checks them against various cases which are stored as reducers. If an action is dispatched that matches a reducers case, the action is handled by that reducer and put on the Redux state tree. From there, it can be accessed from any part of the app that is linked to that Redux state. This methodology would follow the same principle of rendering the graph based on the contents of the state that the React solution has. Redux can either be integrated with React and reactstrap, or it can be used with raw Javascript and the HTML DOM form object.

III. Conclusion

In conclusion, through the use of existing libraries such as React and Redux, we can ease the development of these features and create a cleaner look to the application. Additionally, the integration of these tools into the application add flexibility to its development enabling features to be added later on if needed.