Homework 3 - Component Libraries

By: Constantin Miranda, Faadhil Moheed, & Kyle Harms

1. Overview

A PWA can be built from the ground up with HTML, CSS, and JavaScript. However, professional programmers rarely build apps from the ground up. Instead they use frameworks and libraries to save costs (time + money) in development and maintenance. Vue.js provides a convenient framework for building and structuring an app, but it doesn't the UI controls commonly used in actual user interfaces. In this lab you'll use the Semantic UI component library/framework to reduce the costs of coding the user interface from scratch with HTML+CSS and instead use the pre-defined controls of Semantic UI.

2. Learning Objectives

- Exposure to a typical component library for coding user interfaces.
- Practice using component layouts or containers.
- Use reference documentation to learn a new library.
- Improve general programming abilities.

3. Deadlines and Credit

Assignment	Deadline	Credit
Part I	Thu 10/17, 2:55pm	~20 points
Part II	Thu 10/17, 2:55pm	~20 points
Part III	Thu 10/17, 2:55pm	~60 points

4. Git Repository & Submission

Clone `git@github.coecis.cornell.edu:info4340-fa2019/YOUR_GITHUB_USERNAME-hw3.git`. Replace **YOUR_GITHUB_USERNAME** in the URL with **your actual GitHub username**. This is usually your NetID.

Submit **all** materials to your GitHub repository for this assignment. To submit you much **commit** *and* **push**.

Part I: Design a Calculator

Design a basic calculator UI using *sketches* (not wireframes, or mock-ups). For your design, you may assume that you are *the user* (target audience). If you'd like to pick a different user, please communicate that in your sketches.

Your calculator is required to have an input controls for each of the following:

- Numerical values: 0-9
- Decimal point
- Addition
- Subtraction
- Multiplication
- Division
- Equals
- Back button and/or clear

Feel free to expand the functionality as you desire (exponents, inverse, square root, etc.). Keep in mind McKay's principles that we looked at in the previous homework. Specifically as you design think about how your interface communicates with the user. Are your buttons clear? Does the placement of your buttons create a positive user experience?

Submit (commit and push) a PDF, JPG, or PNG of your sketches to the **documents** folder in your repository.

Part II: UI Component Libraries

Implement your calculator in Vue.js using the Semantic UI component library.

1. About Semantic UI

There are several UI component libraries for applications written in HTML, CSS, and JavaScript. The most well known is Bootstrap. However, Bootstrap's library is a bit limited for the scope of this class. In terms of popularity, Semantic UI doesn't match Bootstrap, but it's well known and used across the web. It has good component (widget) support and integration with popular frameworks like Vue.

The specific integration we will be using is *Semantic UI Vue*. Semantic UI Vue is highly inspired on Semantic UI React and pretty similar to the original Semantic UI with many of its components but modified for Vue.js projects. You will find Semantic UI Vue's API to be almost the same, but **you should always use the Vue-specific documentation when working on this assignment.**

Reference Documentation:

- Semantic UI: https://semantic-ui.com/introduction/getting-started.html
- Semantic UI Vue: https://semantic-ui-vue.github.io/#/

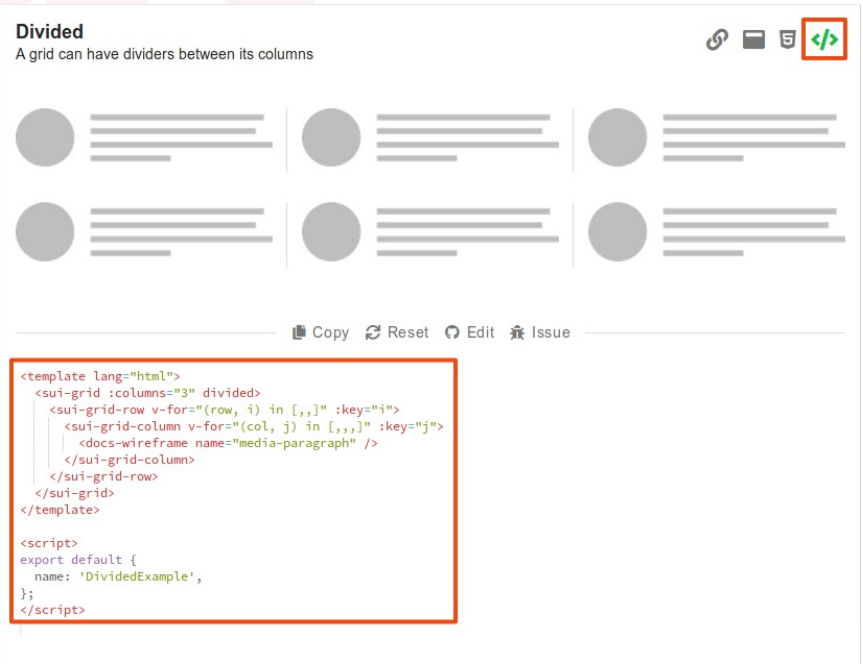
Note: The Semantic UI documentation is more comprehensive than the Vue integration documentation. Always start here first. Then refer to the Semantic UI Vue documentation to learn how to integrate the component into your Vue app.

2. Containers

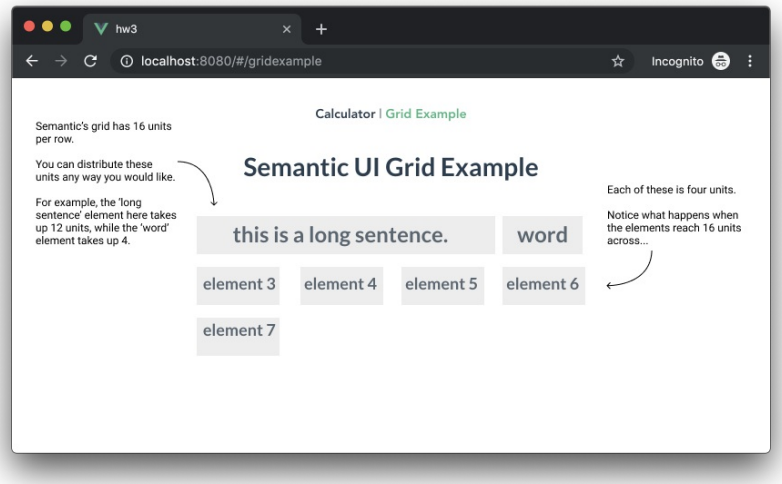
Many component libraries use layout containers to assist designers/programmers in implementing their designs. With PWAs you don't need to use these containers, you can use standard HTML5, and CSS3 technologies like Flexbox. However, for this assignment, I'd like you to gain some experience using these layout containers, since they are a common aspect of most UI component libraries you will likely encounter in the future.

Start by checking out the documentation for Semantic UI's layouts: https://semantic-ui.com/usage/layout.html. Pick a layout, I recommend the Responsive layout. Now view the source of this page. Study the source and the layout (View source or use Developer Tools). Pay particular attention to the section titled 'Device Column Width'. If you aren't sure how it works, copy the source code and play with yourself.

Next, take a look at the Semantic UI Application layout documentation. Notice how you can structure your template portion using `ui grid`, `segment`s, and `column`s. Always study the code examples:



Included in your repository is an example of how to use Semantic UI's grid. When running your development server, you can go to http://localhost:8080/#/gridexample to view it. Study the code and reference the annotated screenshot below to get a better understanding of how to create a responsive layout.



Observe here that the Grid Example uses the method in the Semantic UI documentation using `<div>` instead of `<sui-grid>` and `<sui-grid-row>`. **Try to convert the `<div>`s to use the Semantic UI Vue method.** If you can't get it work, that's okay. The `<div>` method works just fine!

3. Controls

Once you have a grid, you can now place your controls in your app.

Semantic UI has pre-defined UI controls, including:

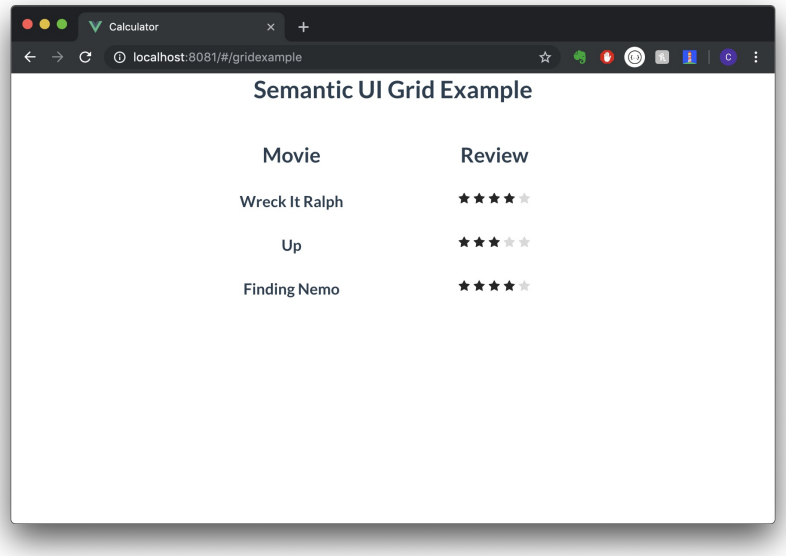
- Input Component (Vue)
- Button Component (Vue)

4. Practice

Let's get some practice with Semantic UI's grid and control components! **Modify GridExample.vue:**

- Modify the columns so that they are equal in width.
- Add a list of 2-3 movies to the left column.
- Add Semantic UI Rating components to the right column. Refer to the Semantic UI Vue documentation for an example on how to integrate the rating component into your Vue app.

Once finished, you should have something that looks like this:



Note: The rating component does not need to be interactive.

Part III: Calculator

Implement your calculator design from Part I. We have included some base code in **Calculator.vue** that will help you get started.

Implement your Part I calculator design in Vue using Semantic UI Vue:

- Calculator must use Semantic UI *Vue* components.

  Exception: Instead of using Semantic UI's grid, you may optionally use Flexbox.

- Implement `append(value)` and `backspace()` functions.
- Your finished app must be fully responsive (mobile, tablet, desktop).
- Calculator must handle both button and key press input gracefully.
- Calculator must be styled.

  Note: Many UI frameworks provide the ability to change grid gutter (the space between columns and rows) size. Semantic UI's documentation states that "Gutters remain a constant size regardless of the width of the grid, or how many columns are in a row." Thus, if you would like to modify button sizes, you will have to override the specific button CSS classes.

Important! In this assignment we used the `eval()` function with user input. Never do this in production code. We used `eval` here to simplify this assignment. In production code this is a **massive** security vulnerability!

1. Implementation Strategy

Place the Semantic UI components first.

Once your grid and buttons are laid out appropriately, you can start to add event listeners so that the proper functions are run depending on which button is pressed. For this you can simply add `@click` attributes to your button controls and create JavaScript functions within the `methods` attribute of your `script` section.

Test everything thoroughly. I should be able to use my num pad on my keyboard and type 1 + 2 and have it report back 3. I should also be able to do the same by clicking the buttons.

Tip: You'll want the callbacks for your buttons to append characters to the `this.formula` variable. `this.formula` is a string, so don't forget about the concatenation operator! You must implement the `append()` and `backspace()` functions in **Calculator.vue**.