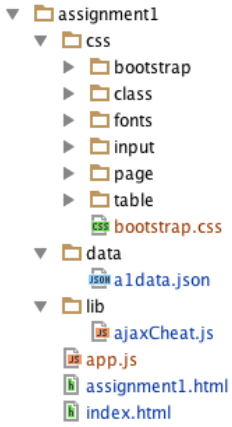
**Pre-Requisites:**

- Web Server (Node server.js script ~ as per week 3 lecture.

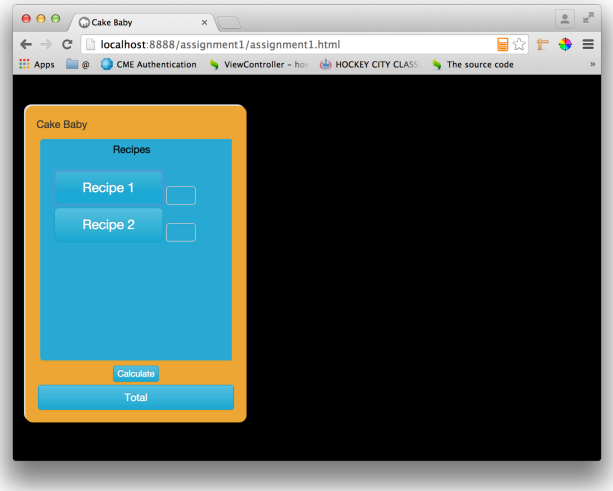
**Installation:**

1) Unzip the assignment to the www directory
2) Startup the server script
3) Navigate to the assignment root directory
   > The index.html should redirect to the assignment1.html page

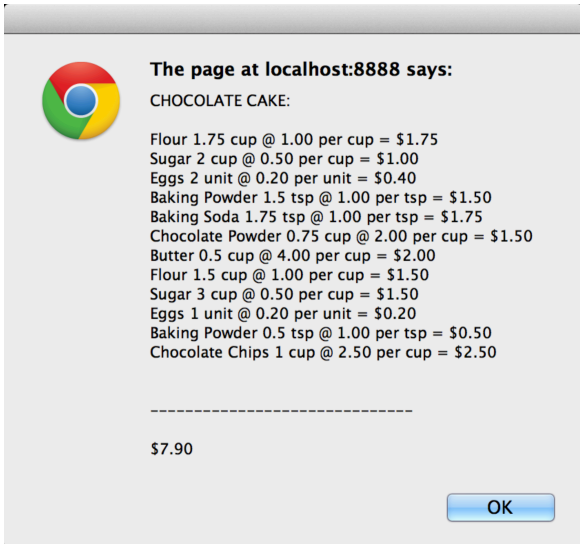
**Files:**




**Startup Wire Frame:**



**Recipe Button Click ~ Wireframe:**

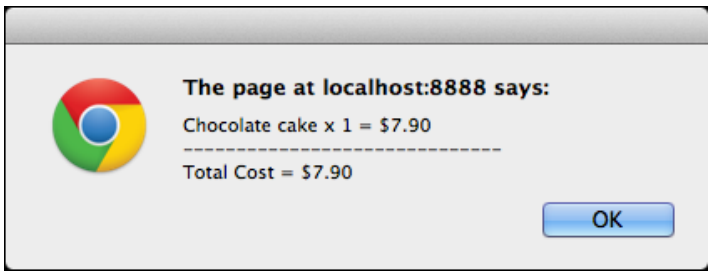



Clicking the Recipe buttons opens an alert dialog displaying the details of the associated recipe model.

**Calculate Button Click ~ Wireframe:**


Clicking the calculate button, results in the value attribute of the Total button being updated based on the aggregation of the recipes cost.

Total Button Click ~ Wireframe:




Clicking the "Totals" button results in a message being displayed the shows the contents of the Aggregate Model. Specifically, it displays the total cost of each recipe with a "serving number" greater than zero. If all recipes having a servings number of zero, then clicking the button does nothing.

**Namespace:**

The application code resides in the app.js script file. This file defines a single namespace called "cakeBaby".

**Models:**

The data supporting the application is broken down into 5 models:

1. **Recipe Model** – This model is used to store and manipulate recipe data. A single recipe model contains multiple ingredient model instances.
2. **Product Model** – This model is used to store and manipulate product data.
3. **Ingredient Model** – This model is used to store and manipulate ingredient data. Each model includes a product model instance.
4. **Aggregate Model** – This model is used to store and manipulate multiple recipe model instances. It is a singleton in that it is only defined once in the application.
5. **Recipe Cache Model** – This model is used to manipulate multiple instances of the same recipe model.

**Design Pattern**

The application uses a prototypal pattern or "Differential Inheritance". Per Douglas Crockford's book, Javascript the Good Parts, "In a purely prototypal pattern, we dispense with classes. We focus instead on the objects … You start by making a useful object. … we can make more instances with the Object create method. … This is called *differential inheritance* (Crockford 50, 51)."

Using this approach, the 5 models (prior section) are consumed like templates. Each time one is needed, a copy is created and reconfigured.

This is an alternate approach to using the “New” operator as discussed in class.

**Outstanding Issues**

Flaky. Depending on the order in which the Number change event fires, the expected behavior may or may not occur. In particular, if update recipe 2, press calculate and then modify recipe 1 and press calculate, the total is updated correctly. If you repeat these steps top down, the total may not be updated correctly. As to what causing this? I believe its because I decided to use a map object instead of an array for my recipe model instance caching. However, I am not certain, and I think I have gone beyond the initial requirements for this project ☺. If this were the real world, this issue would something that I would sort out once the app reaches QA (aka acceptance testing).

**Works Cited**

Crockford, Douglas. "Inheritance." JavaScript: The Good Parts. Sebastopol: O'Reilly Media, 2008. 50, 51. Print.