Tyler Moak

COP3813

Intro to Internet Computing

For the forth project I utilized all my knowledge in JavaScript, jQuery, HTML, CSS/Bootstrap and the MVC type framework to create a todo list.

In this assignment I looked at two different todo lists to get an idea about how to implement it. One todo list I looked at looked easy implement and I used it as a resource in my project. This gave me a starting point from which I can build onto that design. This example also used two resource libraries: director and handlebars.

The director, from my understanding, takes care of url routing. This was too complex for me to understand but the synopsis given by the developer states: “Director is a router. Routing is the process of determining what code to run when a URL is requested.” The second library used was handlebars. Handlebars was necessary, as I found out, to allow easy page building. The description given is that “Handlebars provides the power necessary to let you build semantic templates effectively with no frustration.” This too had too much material for me to research, however, I used my basic understanding of it to finish my site.

The Todomvc jQuery example I used broke up the functions into two variables to which I condensed into one. I also removed the GUID generator function and supplemented another that I found that made more sense. This will be a unique identifier used in storing and fetching data from the localstorage. I removed the bindEvents function in the example as it didn’t help me follow the flow of logic. I substituted this for my own events that would call each function.

To make my project unique, I redesigned the css to fit more into my sites style and I added a feature to favourite specific todo items and a filter to go along with it. Adding that allowed me to get a better understanding of the framework being used including function calling, binding, and overall structure. A user clicks on the favourite star which in turn calls the favourite function in my App.

```
<input id='{{id}}' class="favourite" type="checkbox" {{#if favourite}}checked{{/if}}/>
<label for='{{id}}' class="favourite-label glyphicon glyphicon-star"></label>

$('#todo-list').on('click', '.favourite', function (e) {
        App.favourite($(this));
});

favourite: function (e) {
                e.prop( "checked");
                var val = e.is(":checked");
                this.todos[this.indexFromEl(e)].favourite = val
                this.render();
            }
```

The favourite function changes the value of the check box and then updates the value of that todo's favourite field in the todo list. Finally it renders the change and updates the localstorage. The filter option follows the same idea of the other three filters. Simply by specifying the field I wanted to filter by, the director handled what to show. I don't fully understand this process as it seems to happen behind the scenes in the director library.

The most challenging part of this project was just understanding how the todo mvc example works and functions as well as adding my own elements to the mvc example. It was super confusing at first as to how everything connected. I pulled apart the binding as to better follow how one event gets to the next. Following that, a CRUD function would be called that would perform that operation and save it. As I expected, JSON was used to save data to the localstorage. I've used JSON in the past and it far surpasses other markup languages like XML and is easy to use and understand. The list that stored the TODOs was serialized and stored as

JSON in localstorage and then retrieved and parsed back to a list during the init phase of the App.

What really stumped me was the favourite action for a list item. I really wanted to have the bootstrap glycon-star be clickable to favourite an item. The first thing to get this to work was to place the star in a label and give it an jQuery action. This was easily done, but then I needed to make it so that it would stay "favourited." The html checkbox input was used for this, however, I didn't know how to change the checkbox icon to a star. Instead I found could use for=someid in the label right after the checkbox and hide the input. Now when the label is clicked the checkbox is changed. To turn the star gold I used this css selector:

```
.favourite:checked + label {
                color: gold;
}
```

I used brackets for designing my site. I really like the live preview option it has. Also its minimal interface is just right for me. User extensions were also a nice feature. I'm utilizing a github extension for brackets to have source control. I used Firefox for the majority of my testing as I could see how my webpage can be responsive on different screens.

https://github.com/tastejs/todomvc/blob/gh-pages/examples/jquery/js/app.js

http://stackoverflow.com/questions/105034/create-guid-uuid-in-javascript

http://handlebarsjs.com/

https://github.com/flatiron/director