Weather Route

Weather Route is a web page, written in HTML, CSS, Javascript, Google Maps API and jQuery. There is very little actual HTML on the page, just a few tags. Almost everything you see is generated dynamically.

Formatting, for the most part is handled by map.css in the css subdirectory. There is also some minor use of bootstrap. The code is all contained in map.js in the js subdirectory.

I added a click event listener to the map to handle marker creation, and a click event listener to each marker to handle the infowindow pop up. I added 2 custom properties to the marker objects. One, called numdays, to store the date information, to save the date between calls to reload the route HTML. The second is called json, and is used to store the json weather data object, as part of the marker. That way, I query openweather.org once for each marker, and request 14 days of weather. Updates of weather come from the marker information, and not repetitive calls to openweather.org.

The route information HTML below the map is generated on the fly as the user adds waypoints to the map. The route is generated by the google maps directionsService and rendered by directionsDisplay. I hid the route markers and used generic markers instead, as it was easier to assign my click events to generic markers. I ran into issues trying to assign click events to mid-way point route markers.

I decided to use a single global infowindow, I know I could have assigned one for each marker, but this way, there is only one pop up open at a time.

For the HTML generation, there are functions, that call functions, that call functions. It might seem confusing, but I found it simpler to break the HTML generation into small pieces. So, for example, there is one function that just generates the HTML code for the date option drop down. This was good for code reuse, as well as it was easier to troubleshooting generated HTML.

The weather data is loaded via jQuery and jsonp. I was getting the cross domains error, and couldn't load it any other way. JQuery, makes this very easy. Now, jQuery calls and loads the data asynchronously, so I was having issues with the page trying to load before all the weather data had loaded. I kept getting "undefined" errors. It took a while to figure out what was going on. I had to write a wrapper for the json data that checks for undefined, and returns canned messages like "Weather loading" if the json data hasn't loaded for some reason.