Nicholas Caceres

ITP 301

Project Summary

Car Lords Website

The topic of my website is simply a website revolving around being able to find information about various cars of interest. With the website the user can obtain articles and find nearby dealerships that sell the particular car of interest. In addition when searching for nearby dealerships, using a slider corresponding to radius/distance, the user can filter out more distant dealerships dynamically. In addition upon the initial search, the user can begin a new search for dealerships for a different make and different zip code (initial zip code is set to 91208, so Glendale CA). If the user desires, the search bar is always available and ready to use from any menu. That being said, only one aspect is of concern due to a possible timeout / syntax error with the Edmunds API, namely the search bar. Unfortunately, the Edmunds API not only limits the developer to 25 total calls to it per day, but there seems to be an issue calling its article/editorial JSON data in which either a timeout error occurs or a syntax error is reported in the generated endpoint (the API provides an SDK to circumvent the same-origin policy and employ cross origin resource sharing). In order to deal with these issues for testing purposes, I have inserted sample JSON returns corresponding to the various search parameters a user could come up with and get a response from (search words corresponding to make, model and year of cars interested). I also did the same for dealership search JSON data from the Edmunds API and the vehicle make/model JSON data from the API. It's worth mentioning that since I preloaded several variables with JSON data from the API there are large spaces in the html file where it's purely the variable loading in the raw JSON data. In addition, this data is utilized from the error callback, so it may take up to a minute or two for the data to load and be loaded into the DHTML (depending on what the error is, either the editorial syntax error or the 25 callback 403 forbidden resource limit). In terms of browser issues, I use two jquery plug-ins (Magnific Popover and noUISlider) which should work across browsers except IE8 and lower (IE9+ should be fine).

Design

In terms of design, the html layout features three main sections, a header, a body and a footer. The header and footer remain unchanged across page events, but the body is where most of the activity occurs and DHTML is created. In this body, a set of background images are displayed, constantly fading in, with three drop-down menus in the foreground. In the foreground, the three drop-down menus corresponding to make, model and year which populate with options in that order as the user makes his or her choice. Finally once the year is chosen, the button to begin a search becomes enabled. Upon clicking, a new body div appears with a left side to change make, add a zip code, and slider (from noUISlider) to filter out dealer results by distance. In the right, there is the dealer results based on the make chosen, suggesting to the user (most top result being closest) where to go to find the car they are interested in. Each result displays the name of the dealership, address, distance, and rating from an Edmunds API JSON return. Another way for the user to interact with the webpage is for them to conduct a search. The input is tested against an Edmunds API JSON return of makes, corresponding models and available years for those models using a dynamic regex object. If the input contains all three parameters (make, model, year), two (make and model), or one (make), data is handled accordingly so the input can obtain a JSON return from the Edmunds API with articles about that particular make, model and year. The body will once again swap out but instead it will populate with a div container of search results formatted with a name, description and link. Whenever the body swaps out, a back button is displayed to reach the main menu/body. However, the search bar (and therefore search results) is always available to use as described.

DHTML

The website I feel is quite dynamic and interactive in that it's features all change the majority of the page (the body) in very different ways and formats. In the main body, as the make drop-down menu populates and the user makes a choice, the next drop-down menu populates with models for that make. As the model is chosen, the available years for that model is shown. Finally as the year is chosen, the search button is finally clickable to swap out the body for dealership results. In addition, in the car dealership results body, the ability to modify one's search through zip code, make and radius, once again changes the display, changing displayed results in real time without reloading the page. By using the search bar, the user can have access to a variety of different articles based on their input since the Edmunds API JSON return can be quite different depending on the parameters used. One last small interactive feature is the use of modal popover in the footer. As the user clicks, on items in the footer, modal popovers display with details about the website.

JSON API

The API used was the Edmunds API which is broken into several different API data. I used the Vehicle, Editorial and Dealership data to obtain vehicle, make, model, and year info, car articles info and reviews, and dealership information such as distance from zip code, rating and address. The developer connects to it by using their JS SDK to circumvent the same-origin policy which can create functions to call upon events such as button clicks or option selected changing in a select element (onchange). Each of these functions creates its own endpoint where I passed in parameters that I grabbed from input text elements, drop-down menu (select element) option elements, and slider value. The parameters varied but in general they were strings relating to make, model and year or radius and zip code integers. The API documentation is at ( <http://developer.edmunds.com/api-documentation/overview/>)

Extras

I used two jquery plug-ins as well as regex within my website. The two plug-ins were Magnific Popover and noUISlider. Magnific Popover was used to create modal popovers for the footer list items. It's original intention was to create quick tool tips for dealership results, but I still believe it added value by providing info without an extra html file or taking up extra space, keeping the page small. NoUISlider was used in the dealership results to filter the results found by distance. By grabbing the value of the slider, and comparing it each dealership distance, I would set the result container CSS display to none and the page would update. Finally I used regex objects to handle the search bar as user input would be either a make, model and year, make and model, or simply a make. Using regex objects that were global and case insensitive, constructed from the input, I would use the regex test method to compare against each vehicle API JSON data point comparing make name, model name, and year strings. If a match was found it was saved to a global variable and used to look up articles as a parameter for the Edmunds Editorial Articles API to get article data on the search terms.