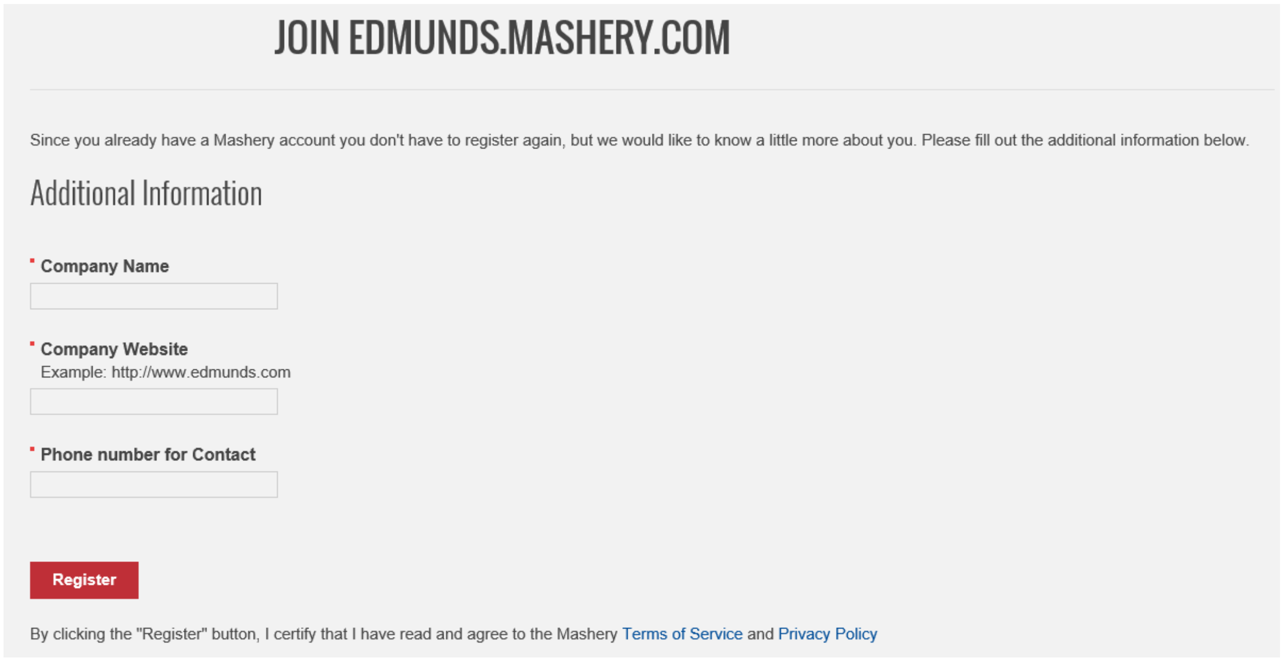
CS290 final project

Tuesday, March 8, 2016 9:37 PM

For the final project of the CSE 290 course I decided to demonstrate the use of the Edmunds API. Edmunds is an engine that offers data on: 1) existing vehicles being produced, 2) the Edmunds editorial, as well as 3) US dealers along with some of their information, which includes 4) the dealer inventory of vehicles. In this assignment I successfully create a server that uses the Edmunds APIs to get, process, and display information pertinent to vehicles and dealers. The first part of the functionality is a command that yields all makes and models in an HTML list and lists the name of the vehicle and its ID as stored in the Edmunds database. The second piece of functionality allows inputting a zip code and results in the listing of all of the dealers pertaining to that specific area, listed with Name, id, and type.

First of all, in order to make the necessary from the server, the developer is required to register with Edmunds in order to get a specific key. This key is unique to every user and is used to track the user activity, as well as enforce the server limits of calls or requests. Signing up is easy and it only requires the Name, Website, and Phone number.



JOIN EDMUNDS.MASHERY.COM

Since you already have a Mashery account you don't have to register again, but we would like to know a little more about you. Please fill out the additional information below.

Additional Information

* Company Name

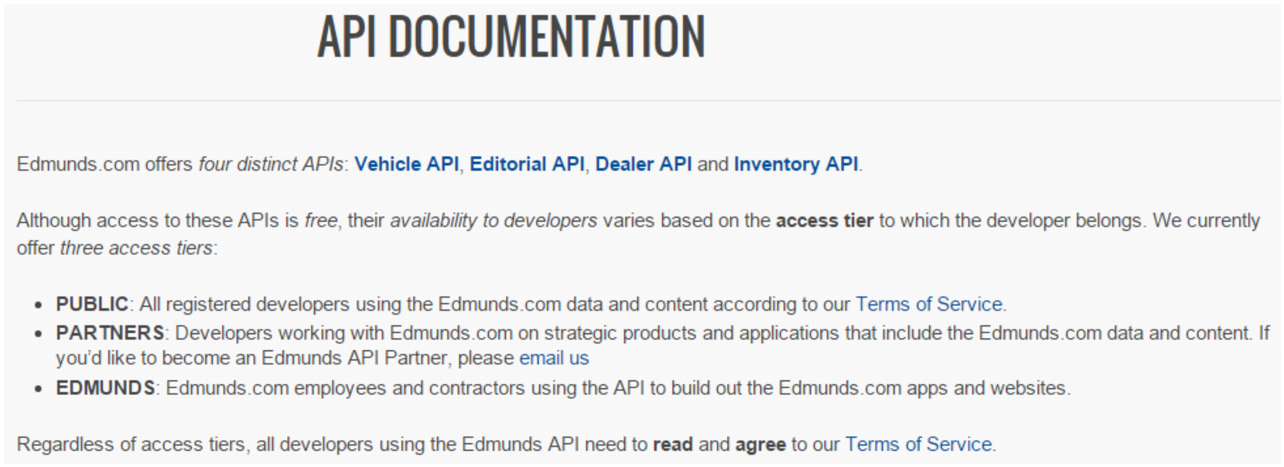
* Company Website
Example: http://www.edmunds.com

* Phone number for Contact

Register

By clicking the "Register" button, I certify that I have read and agree to the Mashery Terms of Service and Privacy Policy

Once signed up, the main page of the API documentation routes to each individual API's page details:



API DOCUMENTATION

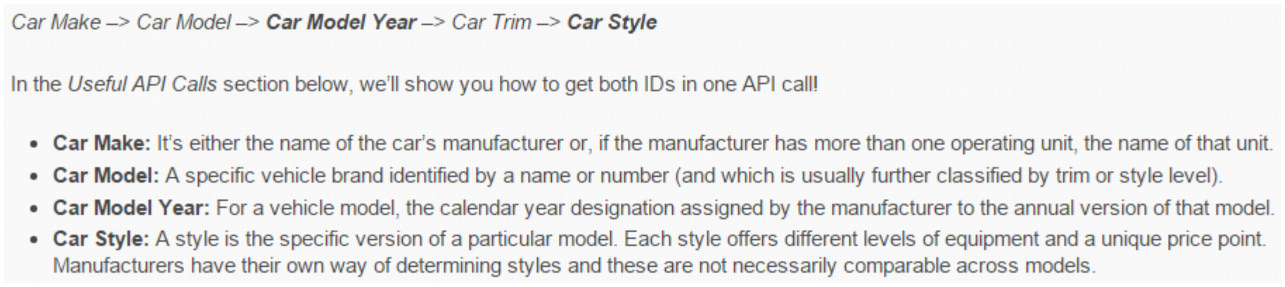
Edmunds.com offers *four distinct APIs*: **Vehicle API**, **Editorial API**, **Dealer API** and **Inventory API**.

Although access to these APIs is *free*, their *availability to developers* varies based on the **access tier** to which the developer belongs. We currently offer *three access tiers*:

- **PUBLIC**: All registered developers using the Edmunds.com data and content according to our Terms of Service.
- **PARTNERS**: Developers working with Edmunds.com on strategic products and applications that include the Edmunds.com data and content. If you'd like to become an Edmunds API Partner, please email us
- **EDMUNDS**: Edmunds.com employees and contractors using the API to build out the Edmunds.com apps and websites.

Regardless of access tiers, all developers using the Edmunds API need to **read** and **agree** to our Terms of Service.

For further information on each specific API, Vehicle API page shows the logic of the structure of the data as pertaining to each car:

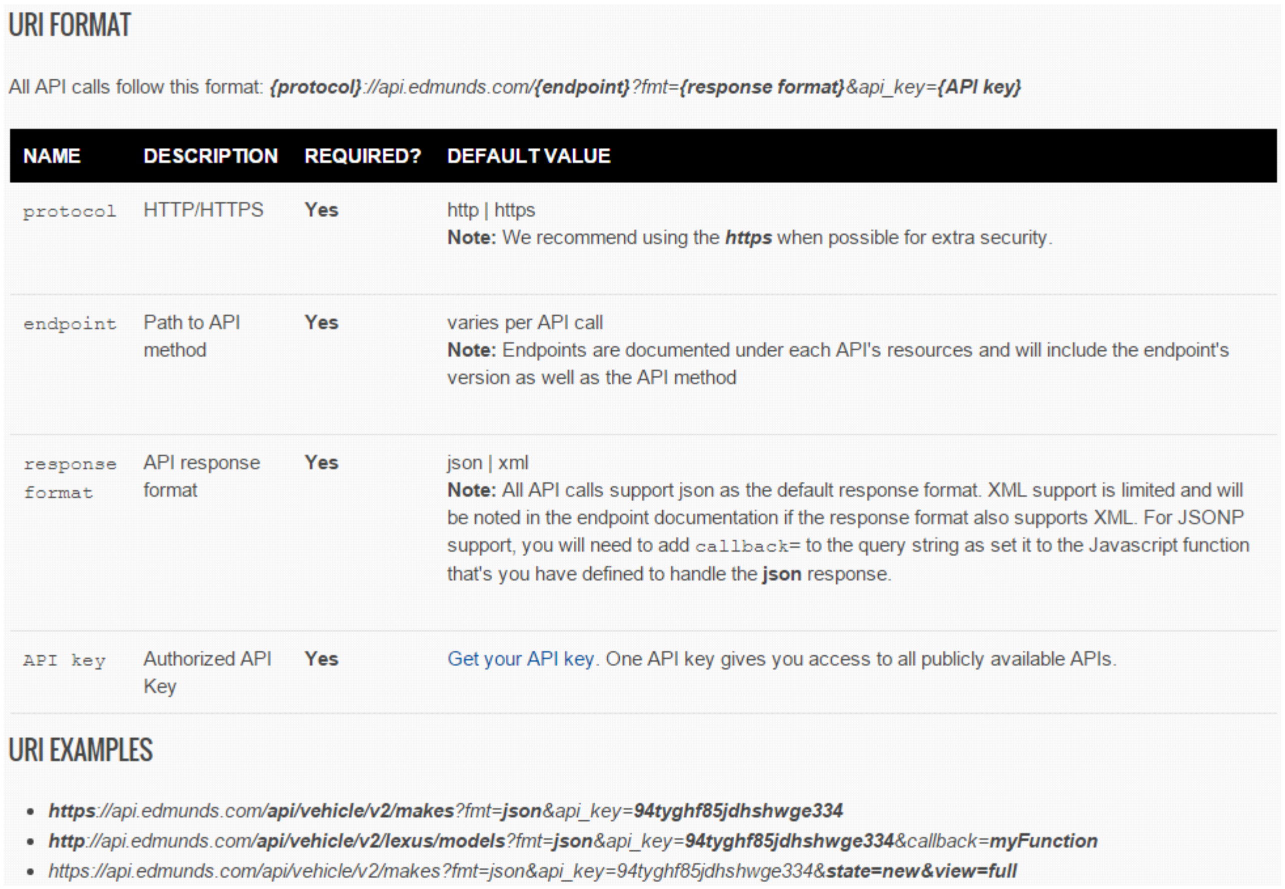


Car Make –> Car Model –> ***Car Model Year*** *–> Car Trim –>* ***Car Style***

In the *Useful API Calls* section below, we'll show you how to get both IDs in one API call!

- **Car Make:** It's either the name of the car's manufacturer or, if the manufacturer has more than one operating unit, the name of that unit.
- **Car Model:** A specific vehicle brand identified by a name or number (and which is usually further classified by trim or style level).
- **Car Model Year:** For a vehicle model, the calendar year designation assigned by the manufacturer to the annual version of that model.
- **Car Style:** A style is the specific version of a particular model. Each style offers different levels of equipment and a unique price point. Manufacturers have their own way of determining styles and these are not necessarily comparable across models.

Also, it provides some useful information on the basic structure of the API calls that it expects, as well as formatting information:



URI FORMAT

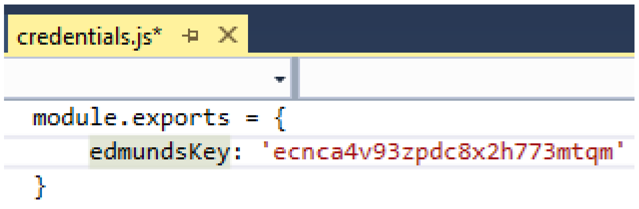
All API calls follow this format: ***{protocol}****://api.edmunds.com/****{endpoint}****?fmt=****{response format}****&api_key=****{API key}***

NAME	DESCRIPTION	REQUIRED?	DEFAULT VALUE
`protocol`	HTTP/HTTPS	**Yes**	http | https **Note:** We recommend using the ***https*** when possible for extra security.
`endpoint`	Path to API method	**Yes**	varies per API call **Note:** Endpoints are documented under each API's resources and will include the endpoint's version as well as the API method
`response format`	API response format	**Yes**	json | xml **Note:** All API calls support json as the default response format. XML support is limited and will be noted in the endpoint documentation if the response format also supports XML. For JSONP support, you will need to add `callback=` to the query string as set it to the Javascript function that's you have defined to handle the **json** response.
`API key`	Authorized API Key	**Yes**	Get your API key. One API key gives you access to all publicly available APIs.

URI EXAMPLES

- ***https****://api.edmunds.com/****api/vehicle/v2/makes****?fmt=****json****&api_key=****94tyghf85jdhshwge334***
- ***http****://api.edmunds.com/****api/vehicle/v2/lexus/models****?fmt=****json****&api_key=****94tyghf85jdhshwge334****&callback=****myFunction***
- *https://api.edmunds.com/api/vehicle/v2/makes?fmt=json&api_key=94tyghf85jdhshwge334****&state=new&view=full***

Once signed up for an account, following the "Get your API key" link allows the user to generate a new and unique API key to be included in every API call. My own key I will be using is '**ecnca4v93zpdc8x2h773mtqm**', and I chose to store it in a separate file named 'credentials.js', included in the 'public' folder of the website, as such:

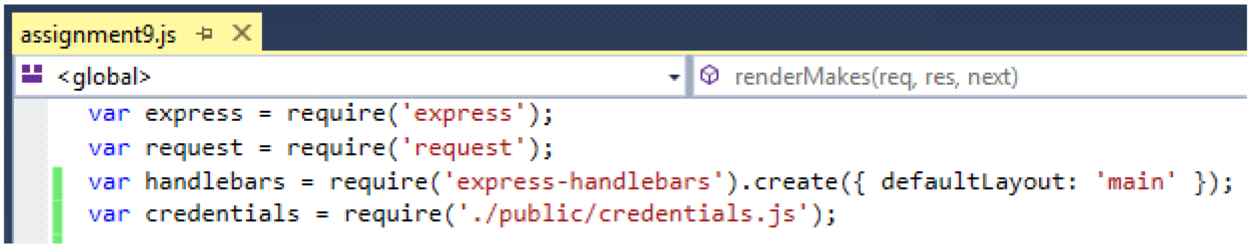


```
module.exports = {
    edmundsKey: 'ecnca4v93zpdc8x2h773mtqm'
}
```

and included in the main javascript file using the command

```
var credentials = require('./public/credentials.js');
```

The node.js programming was centered around the file 'assignment9.js' and included the modules express, express-handlebars, request, and session:



```
var express = require('express');
var request = require('request');
var handlebars = require('express-handlebars').create({ defaultLayout: 'main' });
var credentials = require('./public/credentials.js');
```

Furthermore, another variable 'app' was declared using express, which will be used to make all of the requests of the program:

```
var app = express();
```

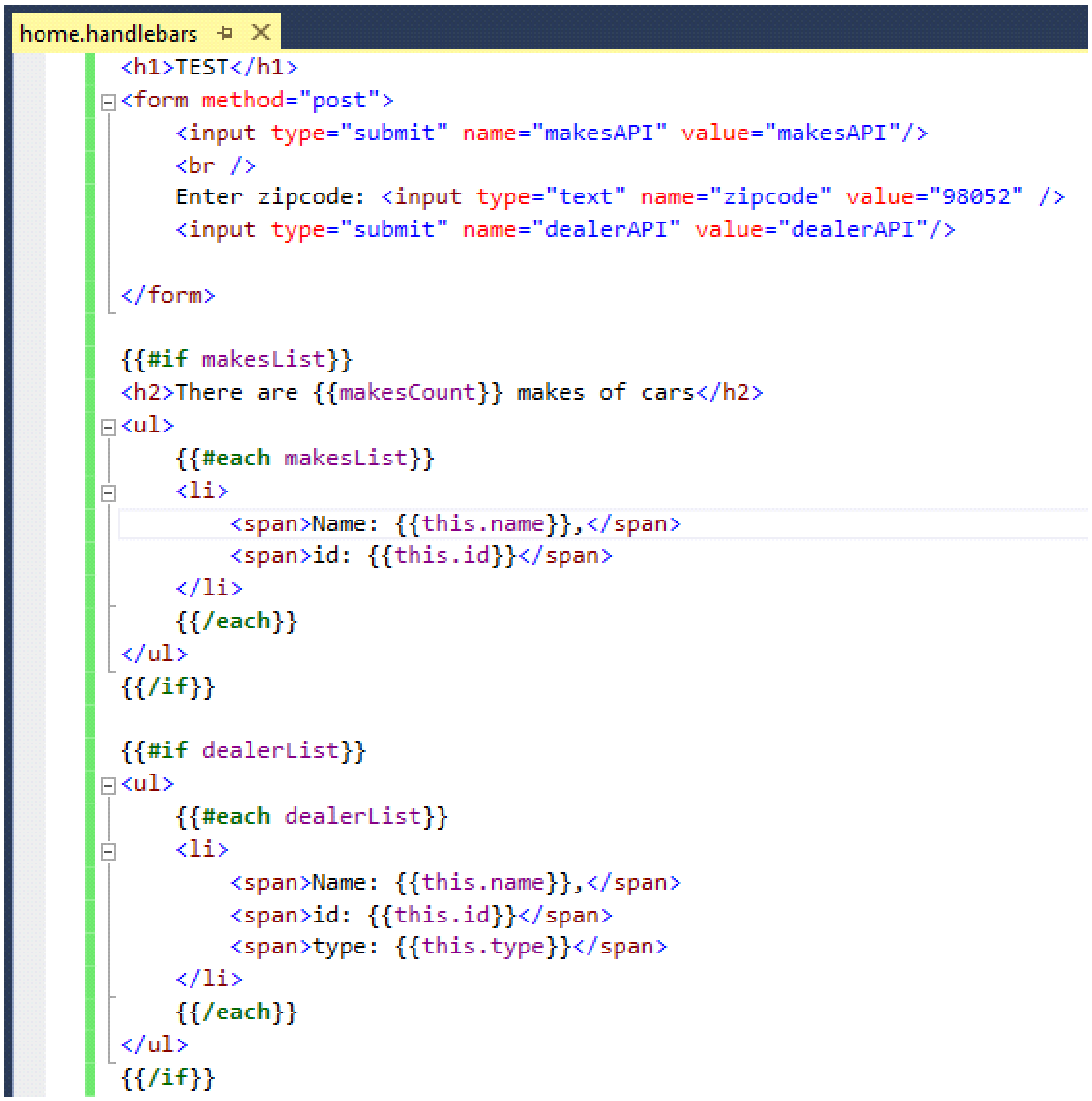
The engine is set to handlebars, and the default port that the application will run locally is 3000:

```
app.engine('handlebars', handlebars.engine);
app.set('view engine', 'handlebars');
app.set('port', 3000);
```

The initial home page will be a static page that enables the website visitor to use the functionality provided. This is accomplished via a simple get request that renders the standard homepage:

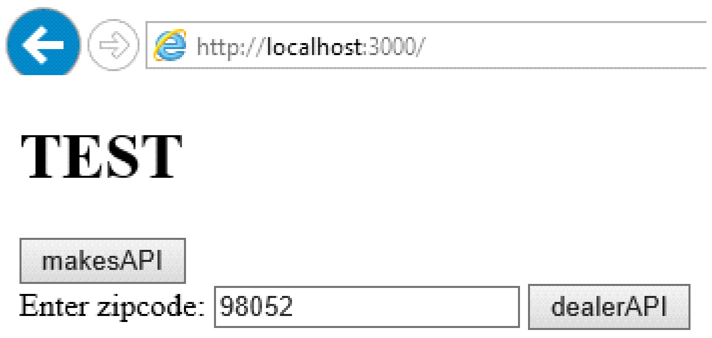
```
app.get('/', function (req, res, next) {
    var context = {};
    res.render('home', context);
});
```

The homepage itself is stored in a file named 'home.handlebars' and contains a simple form with 2 different inputs that each routes to its respective API call. Further down the #if portions of the code are not used initally, since this is a GET request as seen in the function above.



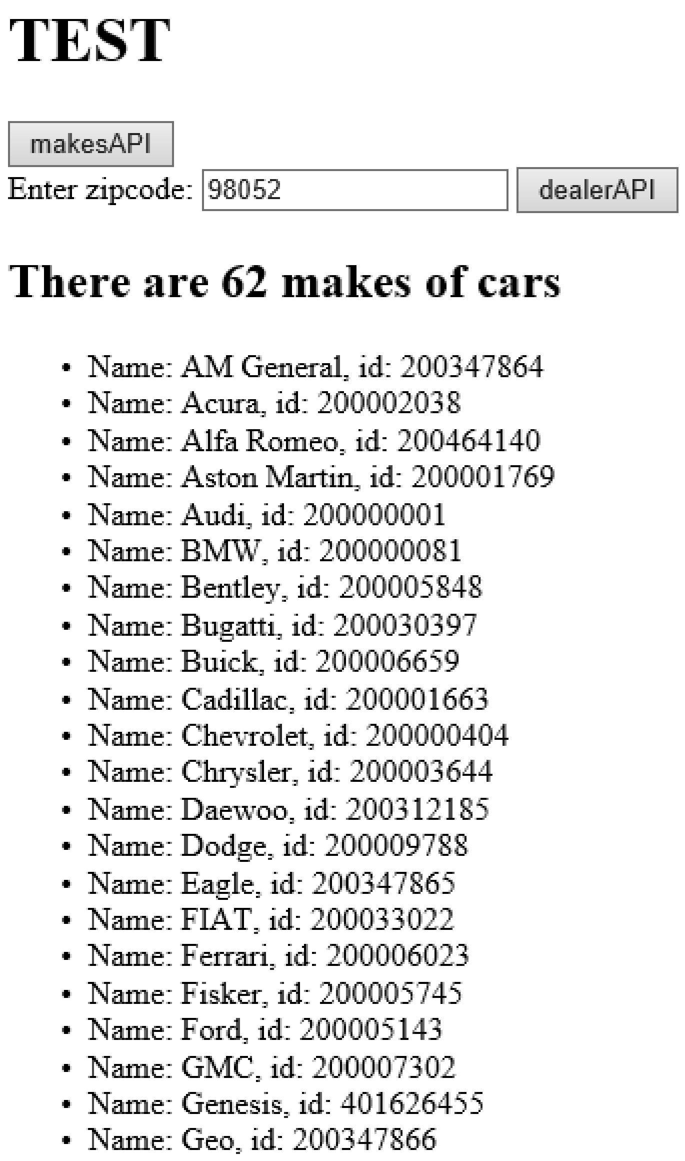
```
<h1>TEST</h1>
<form method="post">
    <input type="submit" name="makesAPI" value="makesAPI"/>
    <br />
    Enter zipcode: <input type="text" name="zipcode" value="98052" />
    <input type="submit" name="dealerAPI" value="dealerAPI"/>

</form>

{{#if makesList}}
<h2>There are {{makesCount}} makes of cars</h2>
<ul>
    {{#each makesList}}
    <li>
        <span>Name: {{this.name}},</span>
        <span>id: {{this.id}}</span>
    </li>
    {{/each}}
</ul>
{{/if}}

{{#if dealerList}}
<ul>
    {{#each dealerList}}
    <li>
        <span>Name: {{this.name}},</span>
        <span>id: {{this.id}}</span>
        <span>type: {{this.type}}</span>
    </li>
    {{/each}}
</ul>
{{/if}}
```

Webpage output:



TEST

Each of the 2 buttons routes to a different API call, and each of the 2 possible results are then displayed on the same page, below the form. Thus, calling the 'makesAPI' button returns the cars stored in the Edmunds database:



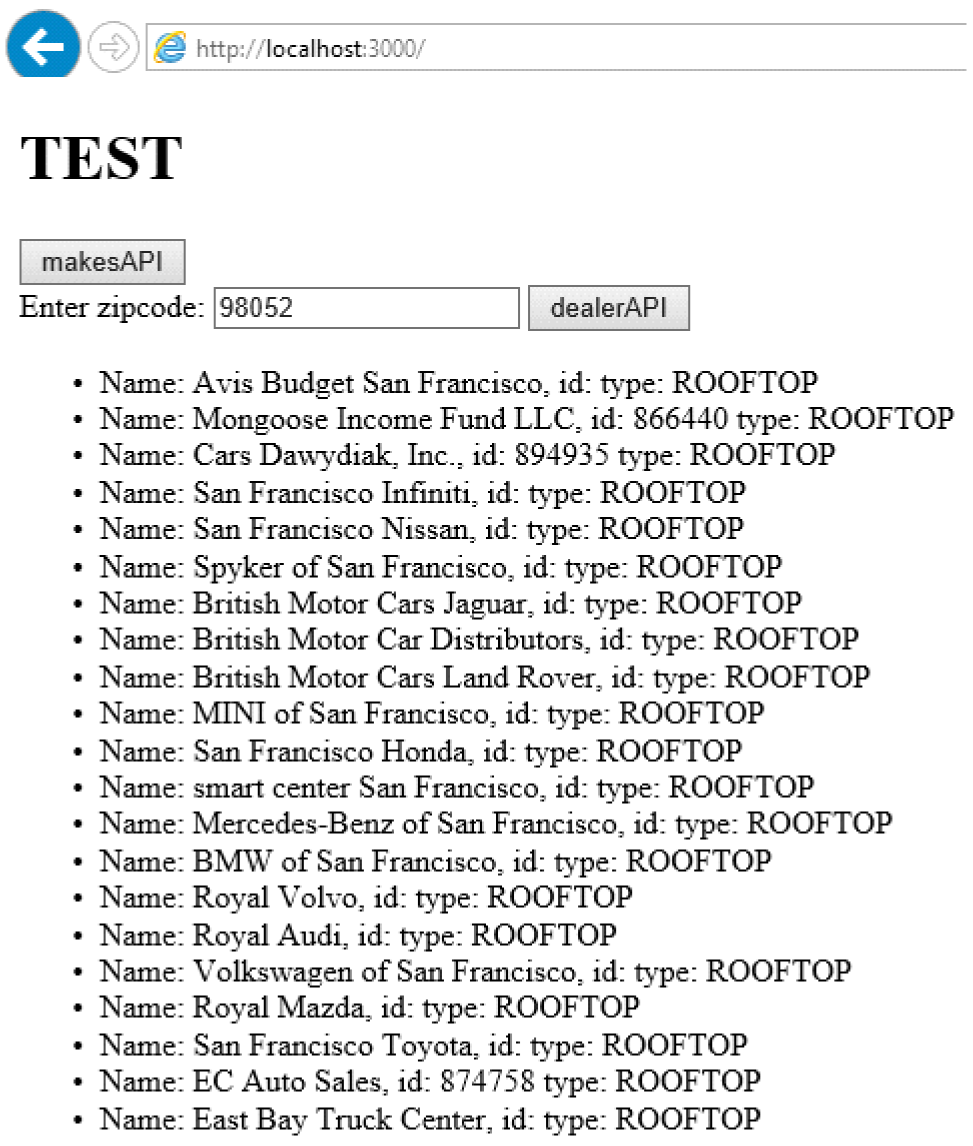
TEST

Enter zipcode: 98052

There are 62 makes of cars

- Name: AM General, id: 200347864
- Name: Acura, id: 200002038
- Name: Alfa Romeo, id: 200464140
- Name: Aston Martin, id: 200001769
- Name: Audi, id: 200000001
- Name: BMW, id: 200000081
- Name: Bentley, id: 200005848
- Name: Bugatti, id: 200030397
- Name: Buick, id: 200006659
- Name: Cadillac, id: 200001663
- Name: Chevrolet, id: 200000404
- Name: Chrysler, id: 200003644
- Name: Daewoo, id: 200312185
- Name: Dodge, id: 200009788
- Name: Eagle, id: 200347865
- Name: FIAT, id: 200033022
- Name: Ferrari, id: 200006023
- Name: Fisker, id: 200005745
- Name: Ford, id: 200005143
- Name: GMC, id: 200007302
- Name: Genesis, id: 401626455
- Name: Geo, id: 200347866

And calling the 'dealerAPI' button returns a list of the dealers specific to the zip code inputted (I set the default to 98052 to display automatically when the page loads, but the user can change the value to any other zip code):


http://localhost:3000/

TEST

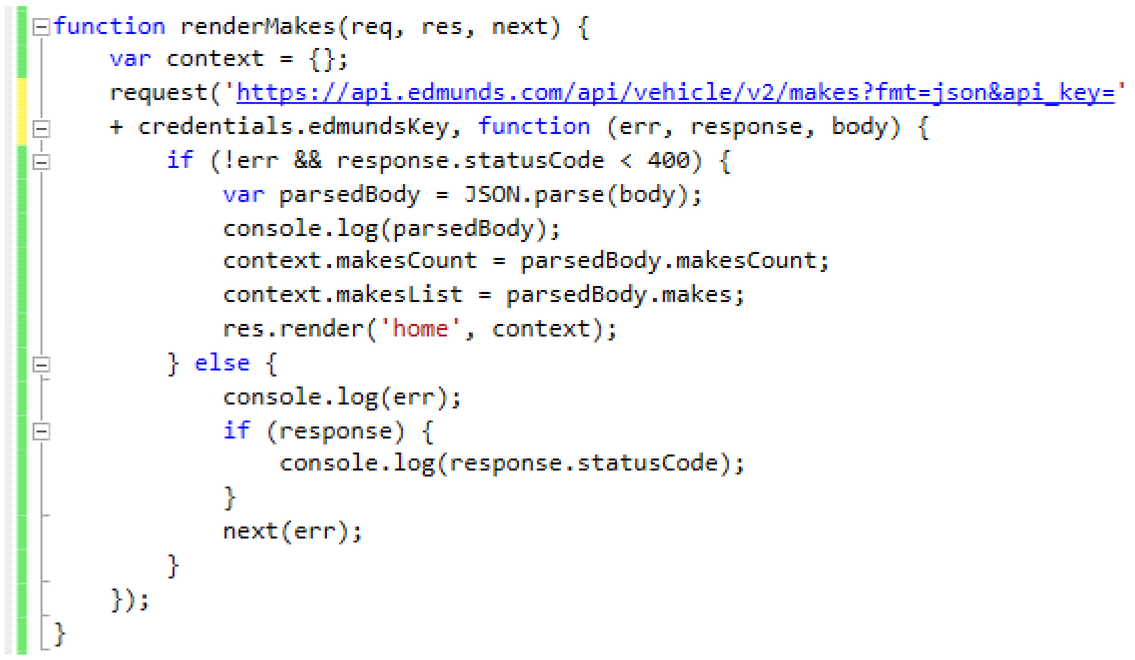
- Name: Avis Budget San Francisco, id: type: ROOFTOP
- Name: Mongoose Income Fund LLC, id: 866440 type: ROOFTOP
- Name: Cars Dawydiak, Inc., id: 894935 type: ROOFTOP
- Name: San Francisco Infiniti, id: type: ROOFTOP
- Name: San Francisco Nissan, id: type: ROOFTOP
- Name: Spyker of San Francisco, id: type: ROOFTOP
- Name: British Motor Cars Jaguar, id: type: ROOFTOP
- Name: British Motor Car Distributors, id: type: ROOFTOP
- Name: British Motor Cars Land Rover, id: type: ROOFTOP
- Name: MINI of San Francisco, id: type: ROOFTOP
- Name: San Francisco Honda, id: type: ROOFTOP
- Name: smart center San Francisco, id: type: ROOFTOP
- Name: Mercedes-Benz of San Francisco, id: type: ROOFTOP
- Name: BMW of San Francisco, id: type: ROOFTOP
- Name: Royal Volvo, id: type: ROOFTOP
- Name: Royal Audi, id: type: ROOFTOP
- Name: Volkswagen of San Francisco, id: type: ROOFTOP
- Name: Royal Mazda, id: type: ROOFTOP
- Name: San Francisco Toyota, id: type: ROOFTOP
- Name: EC Auto Sales, id: 874758 type: ROOFTOP
- Name: East Bay Truck Center, id: type: ROOFTOP

Now let's look a little more in depth at the specific calls that are made when these buttons are pressed. Since we wanted to render the results in the same home page used for this application, it makes sense that the rest of the calls use to the same page to render the new results. However, because we are making additional calls for information to be displayed, these new requests will instead be POST:

```
app.post('/', function (req, res, next) {
    if (req.body['makesAPI']) {
        renderMakes(req, res, next);
    }
    if (req.body['dealerAPI']) {
        renderDealers(req, res, next);
    }
});
```

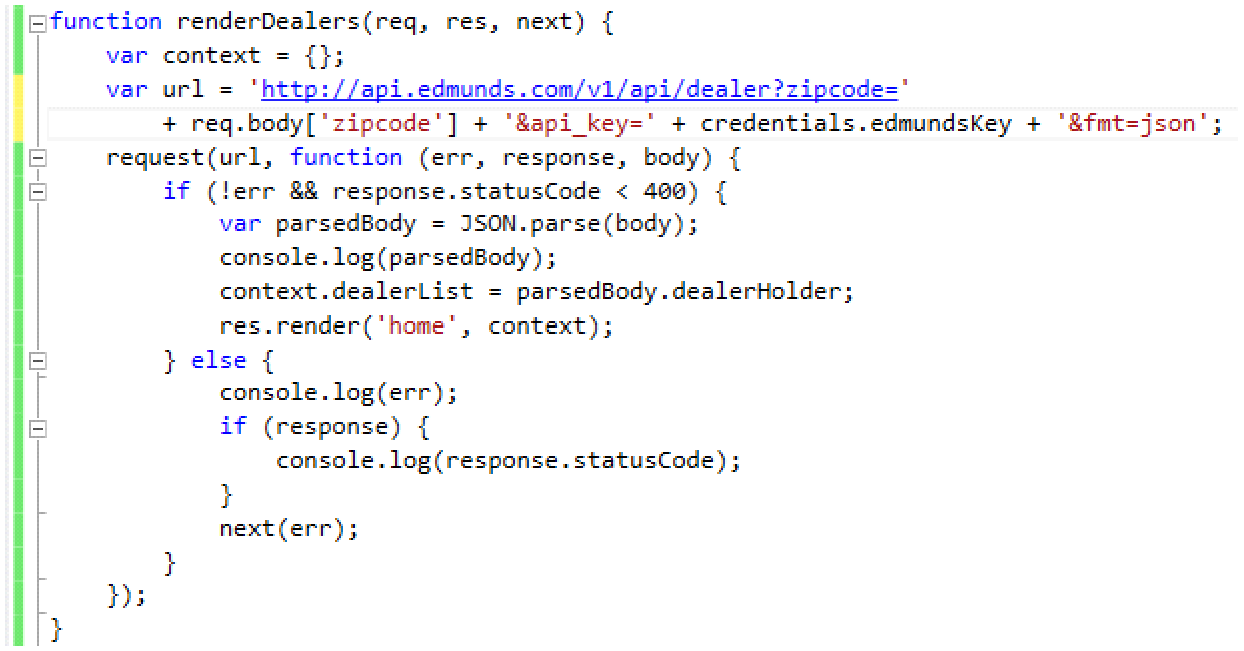
Here, we basically tell the computer that if the form is returned via the 'makesAPI' button, it should execute the function renderMakes() (which intuitively, it should probably display the makes of the cars), whereas if the 'dealerAPI' button was pressed, it should execute the renderDealers() function (which we would assume would display the dealers instead).

These functions are defined separately. Indeed, renderMakes() requests data from the vehicle API using the unique key provided via the credentials.js file mentioned earlier:



```
function renderMakes(req, res, next) {
    var context = {};
    request('https://api.edmunds.com/api/vehicle/v2/makes?fmt=json&api_key='
    + credentials.edmundsKey, function (err, response, body) {
        if (!err && response.statusCode < 400) {
            var parsedBody = JSON.parse(body);
            console.log(parsedBody);
            context.makesCount = parsedBody.makesCount;
            context.makesList = parsedBody.makes;
            res.render('home', context);
        } else {
            console.log(err);
            if (response) {
                console.log(response.statusCode);
            }
            next(err);
        }
    });
}
```

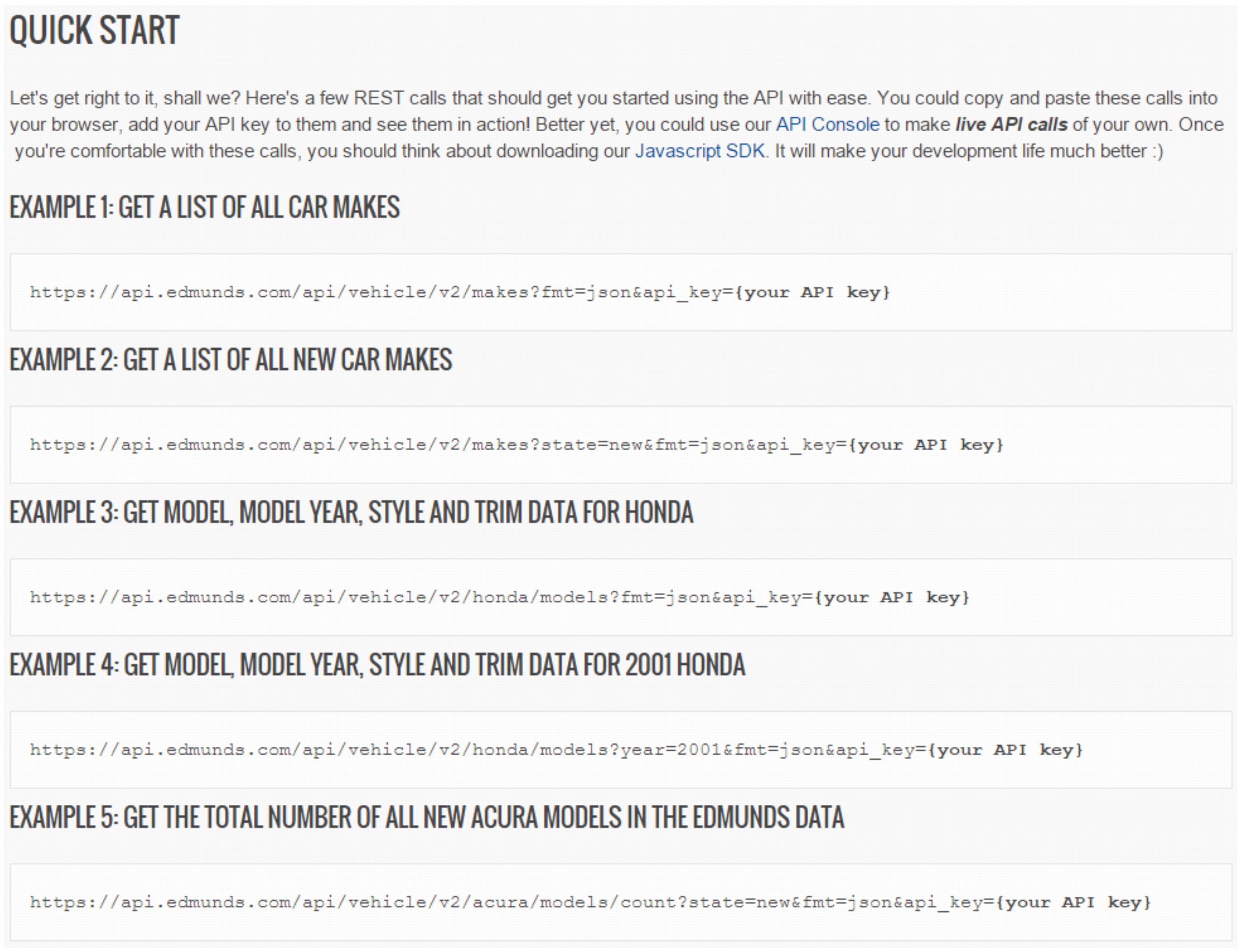
On the other hand, renderDealers() makes the corresponding dealers call using the API key, but it also takes into account the zip code inputted by the user in the interface:



```
function renderDealers(req, res, next) {
    var context = {};
    var url = 'http://api.edmunds.com/v1/api/dealer?zipcode='
        + req.body['zipcode'] + '&api_key=' + credentials.edmundsKey + '&fmt=json';
    request(url, function (err, response, body) {
        if (!err && response.statusCode < 400) {
            var parsedBody = JSON.parse(body);
            console.log(parsedBody);
            context.dealerList = parsedBody.dealerHolder;
            res.render('home', context);
        } else {
            console.log(err);
            if (response) {
                console.log(response.statusCode);
            }
            next(err);
        }
    });
}
```

As you can see, both of these functions render their outputs into the 'home' page. The data format received is JSON, which means that the file needs to be processed into javascript array form in order to be better used. Using the body parser, the content of the response are simultaneously parsed and stored in a new variable called 'parsedBody' which then contains an array of objects with all of the respective properties contained in the Edmunds database. This array is also logged in the console for testing purposes every time the function is called. Once in array form, each element in the array is added sequentially to the context variable of the function, only to then be passed to the render function that generates the content in the homepage using the handlebars library.

Other calls can be customized in order to personalize the results gathered from the server. The documentation lists numerous examples of customization:



QUICK START

Let's get right to it, shall we? Here's a few REST calls that should get you started using the API with ease. You could copy and paste these calls into your browser, add your API key to them and see them in action! Better yet, you could use our API Console to make ***live API calls*** of your own. Once you're comfortable with these calls, you should think about downloading our Javascript SDK. It will make your development life much better :)

EXAMPLE 1: GET A LIST OF ALL CAR MAKES

```
https://api.edmunds.com/api/vehicle/v2/makes?fmt=json&api_key={your API key}
```

EXAMPLE 2: GET A LIST OF ALL NEW CAR MAKES

```
https://api.edmunds.com/api/vehicle/v2/makes?state=new&fmt=json&api_key={your API key}
```

EXAMPLE 3: GET MODEL, MODEL YEAR, STYLE AND TRIM DATA FOR HONDA

```
https://api.edmunds.com/api/vehicle/v2/honda/models?fmt=json&api_key={your API key}
```

EXAMPLE 4: GET MODEL, MODEL YEAR, STYLE AND TRIM DATA FOR 2001 HONDA

```
https://api.edmunds.com/api/vehicle/v2/honda/models?year=2001&fmt=json&api_key={your API key}
```

EXAMPLE 5: GET THE TOTAL NUMBER OF ALL NEW ACURA MODELS IN THE EDMUNDS DATA

```
https://api.edmunds.com/api/vehicle/v2/acura/models/count?state=new&fmt=json&api_key={your API key}
```