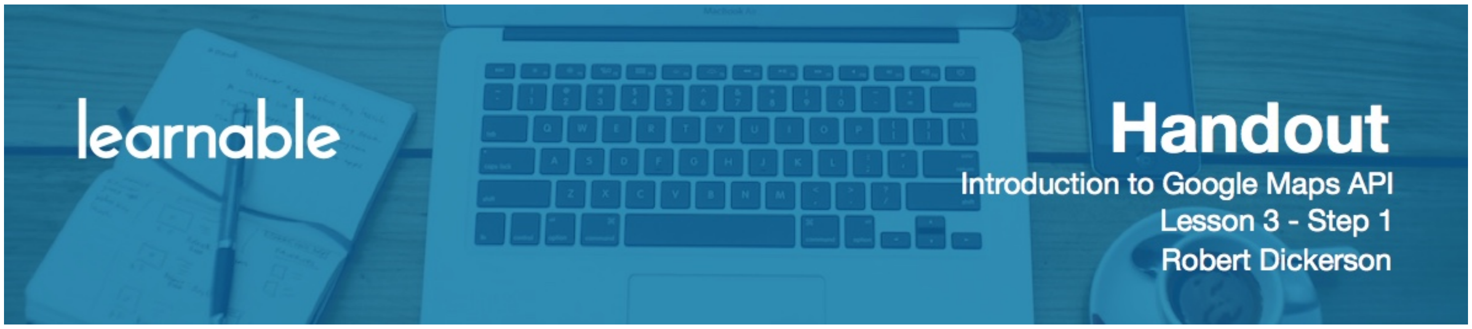

Initial Setup

Let's start coding and create our first map.

If you download the sample code for this lesson or clone the GitHub repository, you'll see that the lesson contains three files: index.html, scripts.js, and styles.css.

- The index.html file is just our main page and it's pretty standard. One thing to note is that I've set the viewport here, so that it'll maximize to the device width.
- The scripts.js is where we'll be doing the majority of the work in future lessons, because that's where we'll interact with the API and the add functionality.
- The styles.css is our stylesheet, and that's where we'll be styling the map and other properties.

Getting Started with Map creation

So there's a couple things you need to do to start incorporating the Google Map.

First of all, include the `script` tag which pulls down the Google Maps API:

```
<script type="text/javascript" src="https://maps.google.com/maps/api/js?v=3"></script>
```

Right now, we're going to use a stable release.

You'll also have to include your API key here, that we talked about earlier.

The next thing you should do is define a `div`, or an element that you can attach the Google Map to:

```
<div id="map"></div>
```

Let's go over and check out the stylesheet. What we want is a totally responsive map that fills up the entire browser. To do that, we're going to take the `html`, `body` and `map` id tag and set the height to 100%, margins and paddings equal to zero:

```
html, body, #map {
    margin: 0;
    padding: 0;
    height: 100%;
}
```

Now open scripts.js. Go ahead and add a global variable, that will be the container for your Google Map object. We're going to just call it map:

```
var map;
```

The next thing you want to do is define a function that gets called when the DOM finishes loading. We're going to call that function `loadMap`:

```
function loadMap() {}
```

Now within the `loadMap` function we instantiate the map object by passing two parameters into the constructor: the DOM element of the container that we want to use to hold the map, and a set of options for configuring the map setup:

```
map = new google.maps.Map(mapId,mapOptions);
```

This is how `mapId` is defined:

```
var mapId = document.getElementById('map');
```

Remember, in the HTML file we defined a `div` with the id of `map`, so that's why we're using `map` here.

The next parameter is a set of options that will be used to configure the basic setup of the map:

```
var mapOptions = {
  //Zoom on load
  zoom: 11,
  //Map center
  center: new google.maps.LatLng(40.748817,-73.985428)

};
```

These are the two required parameters when you go about setting a map. `LatLng(40.748817,-73.985428)` focuses on US and `zoom: 11` just sets kind of a regional view.

Now you need to bind this `loadMap` function to the DOM, so that when the window finishes loading, then you can load the Map API. Google has an event handler called `addDomListener`, which will do it for you. Otherwise you can use the `onLoad` event in the `body` tag:

```
google.maps.event.addDomListener(window, 'load', loadMap());
```

Save and refresh your page. You should see the basic map centered on New York.

Default Controls

Let's walk through some of the default controls and functionality of this basic map we've created.

Using the pan controls, you can move left, right, up and down.

Using the zoom controls, you can zoom in and zoom out.

Using the Street View control, you can pan and zoom and look around.

You also have mouse control so that you can pan and zoom, move around the map using the mouse. All of the styling on the map is the basic styling, and in a future lesson we'll discuss how you can change that.

Styling Map

Now let's go back to the stylesheet, and talk a little bit about what if you wanted to set a fixed height and width. You can do that fairly easily by setting both the `width` and the `height` in CSS. So let's say that you wanted 640 by 480 map:

```
#map {
    width: 640px;
    height: 480px;
}
```