Advanced Maps Using CartoDB and JavaScript

CartoDB includes basic map interactivity such as zooming, infowindows, and layer switchers. You can modify some of these by setting options in `cartodb.createVis()`. See "**Basic Maps Using CartoDB and JavaScript**" for more information on loading and configuring your maps.

This document will walk you through adding other types of interactivity to your map such as filtering within a layer (showing only some of the features for a layer). Using the CartoDB and jQuery[1] libraries, it can be relatively simple to add buttons to your page that affect the map.

jQuery

When you want your map to be more interactive, this often means you want the map to change when someone clicks an element on your page. Luckily for us, jQuery makes it easy to know when this happens.

In jQuery you select elements in your page, then modify them or wait for things to happen to them[2]. You can use any CSS selector to select elements. So, for example, if you have a div like this:

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

then you can select it with the selector `#map`, as you would in CSS. To select the element with jQuery, you say

```
$('#map')
```

This on its own does not do anything. You have to follow your selection with a jQuery function. For instance, if you wanted to hide your map `div`, you could easily do this by saying

```
$('#map').hide()
```

and you could show it again by saying

```
$('#map').show()
```

There is much more that you can do with jQuery by selecting elements and calling functions on them. Right now we want to listen for clicks happening to our element.

---

1 jQuery is a JavaScript library. It contains a significant amount of code that you can use on your pages. Learn more at http://jquery.com

2 When "things happen to" HTML elements, we call it an `event`. When you are waiting for an event to happen, we say you are "listening for an `event`."

Listening for Clicks

It's very common in JavaScript to need to listen for clicks on HTML elements. It's the most basic kind of interaction that we have with web pages. For that reason, jQuery makes it very quick and simple to do it. For example:

```
$('#map').click(function () {
    $('#map').hide();
});
```

When you load a page with the above code, nothing noticeable will change on the page. That's because we're asking jQuery to hide the map `div` when it is clicked. Okay, that's pretty useless, but we'll be making it more useful soon. First, how does this work? There are three steps:

1. Select the map `div`. This is on the first line: `$('#map')`
2. Call `.click()` on the map `div`, which is also happening on the first line: `$('#map').click(`
3. Run some code within the `click` function. This is everything between the `(` and `)` after click:
   ```
   function () {
       $('#map').hide();
   }
   ```

   You could run anything in here, but it always has to start with `function () {` and end with `}`. In this case, we're selecting the map `div` again and calling `hide()` on it.

This is some pretty weird-looking code, but it's very common in JavaScript. Look over it again, but don't obsess over the syntax. You'll get used to it eventually. For now, you should know that you can essentially copy:

```
$('yourSelector').click(function () {
    // Add your code here
});
```

and (1) replace `yourSelector` with a selector that picks the elements you want to be clickable, then (2) replace the second line with whatever code you want to run when those elements are clicked.

Changing the SQL for a CartoDB Layer

Now that we know how to listen for clicks on elements let's look at how you can change the SQL for a CartoDB layer in your visualization through JavaScript. This can be broken into two steps:

1. Get the layer you will want to change. In "**Basic Maps Using CartoDB and JavaScript**" we added CartoDB visualizations using the following code:

   ```
   $(document).ready(function () {
   ```

```
    cartodb.createVis(your_map_id, your_visualization_url);
});
```

In order to hold on to the data layer from the visualization, change those three lines to:

```
$(document).ready(function () {
    var dataLayer;
    cartodb.createVis(your_map_id, your_visualization_url)
    .done(function (vis, layers) {
        dataLayer = layers[1].getSubLayer(0);
    });
});
```

The new code is bold. What does this code do?

```
var dataLayer;
```

creates a variable named `dataLayer` where we will put the layer.

```
.done(function (vis, layers) {
    dataLayer = layers[1].getSubLayer(0);
})
```

once the visualization is done loading, it grabs the data layer and puts it in our `dataLayer` variable. If you have more than one layer in your visualization, you can access each by changing `1` to the appropriate number—for layer 3, use `dataLayer = layers[3].getSubLayer(0)`.

2. Set the SQL for that layer. First you should determine the SQL that you will want to use to filter the data in your layer by logging in to CartoDB and experimenting with SQL statements until you find one that does what you need. Remember that the default SQL statement for tables is

   ```
   SELECT * FROM tableName
   ```

   Your SQL statement will likely be similar, but you will replace `tableName` with the name of your table and add a `WHERE` clause. For example, you might use

   ```
   SELECT * FROM cities WHERE population > 6000000
   ```

   to filter a table called `cities` to just those cities with `population` above 6 million.

   Once you find an SQL statement that does the right thing you can set the SQL on your data layer by saying

   ```
   dataLayer.setSQL('yourSQLStatement')
   ```

   where `yourSQLStatement` is the SQL statement that you decided to use. If you add this line

after the `dataLayer = …` line from (1) above, you will filter the layer as soon as the map loads. If you want to filter the layer when something is clicked, you will have to listen for clicks on an element and use `setSQL()` there. Let's do that now.

Putting it All Together

Now that we know how to listen for clicks and set the SQL for a layer, we can put the two together to make a button that filters the layer only when someone clicks on it.

First, create a button. This will often be a `div`, and it could be on top of the map or to the side. Whatever works for your page. Give this `div` an `id`. For example:

```
<div id=”large-cities-only”>
    show large cities only
</div>
```

Style the `div` as you like.

Listen for clicks on the `div`:

```
$('#large-cities-only').click(function () {
    // Button-click code here
});
```

Then set the SQL within the click handler:

```
$('#large-cities-only').click(function () {
    dataLayer.setSQL('SELECT * FROM cities WHERE population > 6000000');
});
```

(Don't forget to replace the selector and SQL as suits your page and map.)

Finally, put this code in `ready()`, at the end. Like this:

```
$(document).ready(function () {
    var dataLayer;
    cartodb.createVis(your_map_id, your_visualization_url)
    .done(function (vis, layers) {
        dataLayer = layers[1].getSubLayer(0);
    });
    $('#large-cities-only').click(function () {
        dataLayer.setSQL('SELECT * FROM cities WHERE population > 6000000');
     });
});
```

Conclusion

Your code should look something like this by the time you're finished:

```
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" href="http://libs.cartocdn.com/cartodb.js/v3/themes/css/cartodb.css" />
        <!--[if lte IE 8]>
        <link rel="stylesheet"
href="http://libs.cartocdn.com/cartodb.js/v3/themes/css/cartodb.ie.css" />
        <![endif]-->
```

```
        <script src="http://libs.cartocdn.com/cartodb.js/v3/cartodb.js"></script>
        <script src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.0/jquery.min.js"></script>

        <style>
            #map {
                height: 500px;
                width: 500px;
            }
        </style>

        <script>
            $(document).ready(function () {
                var dataLayer;
                cartodb.createVis('map', 'your_visualization_url')
                .done(function (vis, layers) {
                    dataLayer = layers[1].getSubLayer(0);
                });
                $('#large-cities-only').click(function () {
                    dataLayer.setSQL('SELECT * FROM cities WHERE population > 6000000');
                });
            });
        </script>

    </head>
    <body>
        <div id="map"></div>
        <div id=”large-cities-only”>
            show large cities only
        </div>
    </body>
</html>
```

Next Steps

Now that you've added a button that filters the features on your map, think about how you would add multiple buttons to your page. How do you think you would you change the CartoCSS for the layer? What else would you like to be able to do with your map?