JavaScript V

Alerts and Dialogs

- For many years, alerts and dialogs, which pop up over the browser, were popular forms of user interaction
  - These days there are nicer ways to handle these interactions, collectively known as modals
  - Using modals requires libraries and a bit more than I want to get into for this class, so we will learn the old fashion alerts and dialogs
- Alerts and dialogs have their place
  - Just don't over use them
  - Most modern browsers allow these to be supressed

Alert

- An alert is triggered by calling the `alert` function, which takes a single string parameter
- The alert will contain the message of the string, and a single button allowing the window to be dismissed

In [ ]:
```
%%javascript
alert("HELLO");
```

Confirm

- `confirm` is similar to alert, taking one string parameter
- The popup will contain two choices that allow for the box to be dismissed
  - 'Cancel'
  - 'OK'
- The function will return `true` or `false` depending on the answer

In [ ]:
```
%%javascript
var resp = confirm("Please select cancel or ok");
if(resp === true){
    alert("Its ok!");
}
else{
    alert("Its been cancled");
}
```

Prompt

- The final popup box is prompt
- Can take one or two string parameters
  - Prompt messsage
  - Default value
- Provides a text box to collect input
  - Function returns what is typed in box

In [ ]:
```
%%javascript
var resp = prompt("Please type your name:","Batman");
alert("Hello " + resp + "!")
```

JavaScript Libraries

- Like most scripting languages, there is a wide number of JavaScript libraries available and commonly used
  - There is no one source for these like in R, but there are several popular ones
- JavaScript libraries can either be downloaded and hosted by you, or you can use one hosted by someone else
  - The script tag takes a URL, it can be a fully formed web address

Finding JavaScript Libraries

- Besides word of mouth, there are several websites I use to find libraries when I need them
  - javascripting.com
  - cdnjs.com

Content Delivery Network

- A content delivery network (CDN) is a distributed network that will load JavaScript libraries (or other resources) from which ever server is faster for the user
- This provides speed up in two ways
  - Ensures the transfer over the network is fast
  - Allows many different sites to use the same cached version of a file

jQuery

- jQuery is one of the oldest and most popular JavaScript libraires around
- Provides
  - Easy DOM selection
  - Simple events
  - CSS manipulation
  - AJAX wrapper classes

Using jQuery

- As JavaScript has gotten more advanced, the need for jQuery has decreased
  - If something is available in native JS, it will usually be faster than the jQuery equivalent
- That being said, many libraries rely on jQuery, and it is great for writing code quickly
- To use it, always use a CDN
  - Since so many sites use it, its unlikely you'll ever have to actually send it over the network

Current jQuery Script Tag

```
<script
    src="https://code.jquery.com/jquery-3.2.1.min.js"
    integrity="sha256-hwg4gsxgFZhOsEEamdOYGBf13FyQuiTwlAQgxVSNgt4="
    crossorigin="anonymous"></script>
```

The `$()` function

- The main function in jQuery is `$()` which is used primarily one of two ways
  - To run code when the document is ready

```
$(function(){
})
```

  - To select from the DOM

```
$('selector')
```

In [ ]:

```
%%html
<!DOCTYPE html>
<html>
    <head>
    <script
    src="https://code.jquery.com/jquery-3.2.1.min.js"
    integrity="sha256-hwg4gsxgFZhOsEEamdOYGBf13FyQuiTwlAQgxVSNgt4="
    crossorigin="anonymous"></script>
    <script>
    $(function(){
        $('#jq1').html("HELLO")
    }
    );
    </script>
    </head>
    <body>
        <p id="jq1"></p>
    </body>
</html>
```

AJAX in jQuery

- The $.ajax function allows a JavaScript object to be passed to make an AJAX request
- There are too many parameters to list, but a common example might look like

```
$.ajax({
    url: 'myurl.php',
    success: function(){},
    dataType: 'json',
    data: {"key":val},
});
```

MomentJS

- As shown earlier, the Date object is very bare bones in JavaScript
- The `moment` function returns a suped-up date object
- Common operations
  - `.format()` allows the date to be formatted in a wide array of strings
  - `.subtract()` \ `add()` performs calendar math
  - `.fromNow()` returns a string representing how long ago\ from now something was\ will be

In [ ]:

```
%%html
<!DOCTPYE>
<html>
    <head>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.19.1/moment.mi
n.js"></script>
    </head>
    <body>
        <p id="mo1"></p>
        <p id="mo2"></p>
        <p id="mo3"></p>
        <p id="mo4"></p>
        <p id="mo5"></p>
    <script>
        var now = moment();
        document.getElementById("mo1").innerHTML = now.format();
        document.getElementById("mo2").innerHTML = now.format("MM/DD/YYYY");
        document.getElementById("mo3").innerHTML = now.endOf('hour').fromNow();
        document.getElementById("mo4").innerHTML = now.subtract(12,'days').format
("DD/MM/YYYY");
        document.getElementById("mo5").innerHTML = moment().add(12,'days').calenda
r();
    </script>
    </body>
</html>
```

LeafletJS

- Adding maps to pages is a very common task
  - Google Maps has long been the goto API, but due to wanting a bit more control, many other alternatives have appeared
- LeafletJS allows you to add interactive mapping to your page
  - Uses any number of tile servers to actually produce the map

Creating a Map

- To create a map in Leaflet JS, you need to tell it the tile server you want to use
  - You should also place some attribute about where the data is coming from
- The `tileLayer` of the map handles the drawing of the tiles

```
javscript
my_map.tileLayer(tileURL, {attribution: 'attribution html'})
```

In [ ]:

```
%%html
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.2.0/
leaflet.css"
  integrity="sha256-LcmP8hlMTofQrGU6W2q3tUnDnDZ1QVraxfMkP060ekM="
  crossorigin="anonymous" />
<script src="https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.2.0/leaflet.js"
        integrity="sha256-kdEnCVOWosn3TNsGslxB8ffuKdrZoGQdIdPwh7W1CsE="
        crossorigin="anonymous"></script>
        <style>
        #mapid{height:30em;}
        </style>
</head>
<body>
    <div id="mapid"></div>
    <script>
        var mymap = L.map('mapid');
        L.tileLayer('https://cartodb-basemaps-{s}.global.ssl.fastly.net/rastertile
s/voyager/{z}/{x}/{y}.png',
        {
        attribution: '&copy; <a href="http://osm.org/copyright">OpenStreetMap</a>
 contributors,  &copy; <a href="https://carto.com/attribution">CARTO</a>'
        }).addTo(mymap);
    </script>

</body>
</html>
```

Setting the Location

- The location is set by passing the latittude and longitue to the `setView` function as an array.
  - This function takes a second parameter which is the zoom level, 0 being the furthest out;

In [ ]:

```
%%html
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.2.0/
leaflet.css"
  integrity="sha256-LcmP8hlMTofQrGU6W2q3tUnDnDZ1QVraxfMkP060ekM="
  crossorigin="anonymous" />
<script src="https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.2.0/leaflet.js"
        integrity="sha256-kdEnCVOWosn3TNsGslxB8ffuKdrZoGQdIdPwh7W1CsE="
        crossorigin="anonymous"></script>
        <style>
        #mapid{height:30em;}
        </style>
</head>
<body>
    <div id="mapid"></div>
    <script>
        var mymap = L.map('mapid');
        mymap.setView([0,0],0);
        L.tileLayer('https://cartodb-basemaps-{s}.global.ssl.fastly.net/rastertile
s/voyager/{z}/{x}/{y}.png',
        {
        attribution: '&copy; <a href="http://osm.org/copyright">OpenStreetMap</a>
 contributors,  &copy; <a href="https://carto.com/attribution">CARTO</a>'
        }).addTo(mymap);
    </script>

</body>
```

Interacting with the Map

- LeafletJS handles zooming and moving around the map (panning) for you
- If you want to perform an action when this happens, you can listen to events by calling the `on` function on your map object

```
map.on('move',function(){
                        });
```

- A full list of events is available in the leafletJS documentation, http://leafletjs.com/reference-1.2.0.html#map-event

Adding to the Map

- Adding markers to the map is very simple, use the `marker([lat,long])` function
  - This is called on the leaflet object, `L` and then added to the map
- By default, this simply places a marker,
  - To have something happen when you click on it, use `bindPopup()`

In [ ]:

```
%%html
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.2.0/
leaflet.css"
  integrity="sha256-LcmP8hlMTofQrGU6W2q3tUnDnDZ1QVraxfMkP060ekM="
  crossorigin="anonymous" />
<script src="https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.2.0/leaflet.js"
        integrity="sha256-kdEnCVOWosn3TNsGslxB8ffuKdrZoGQdIdPwh7W1CsE="
        crossorigin="anonymous"></script>
</head>
<style>
        #mapid {height:30em;}
</style>
<body>
    <div id="mapid"></div>
    <script>
        var mymap = L.map('mapid');
        mymap.setView([0,0],5);
        L.tileLayer('https://cartodb-basemaps-{s}.global.ssl.fastly.net/rastertile
s/voyager/{z}/{x}/{y}.png',
        {
        attribution: '&copy; <a href="http://osm.org/copyright">OpenStreetMap</a>
 contributors,  &copy; <a href="https://carto.com/attribution">CARTO</a>'
        }).addTo(mymap);
        var m = L.marker([0,0]).addTo(mymap);
        m.bindPopup("This is 0,0");
    </script>

</body>
</html>
```

More APIs

- As more and more of daily computing moves to the internet, it becomes important to be able to interact with the devices themselves
- Modern APIS aren't part of JavaScript proper, but allow this
  - Geolocation API
  - Vibration API
  - Notification API

Geolocation

- The geolocation API is found off of the `window.navigator` object
  - If not present, `window.navigator.geolocation` will not exist
- Most web browsers will prompt the user for permission to share this information
- Important methods
  - `getCurrentPosition()`
  - `watchPosition`

In [ ]:

```
%%html
<!DOCTYPE html>
<html>
<head>
<body>
    <div id="loc1"></div>
    <script>
       if('geolocation' in window.navigator)
       {
           window.navigator.geolocation.getCurrentPosition(
            function(position) {

              //document.getElementById('loc1').innerHTML =
               alert("You are at " +
                   position.coords.latitude + ", " +
                   position.coords.longitude + "!");
           });
       }
       else{
           document.getElementById("loc1").innerHTML = "Position Not Available";
       }
    </script>

</body>
</html>
```

Notification API

- The notification API also requires permission, but this must be handled programmatically

```
javsascript
Notification.requestPermission();
```

- Once permission is given, a OS notification can be initiated by creating a new notification object

```
new Notification("Hello");
```

In [ ]:

```
%%html
<!DOCTYPE html>
<html>
<head>
<body>
    <div id="not1"></div>
    <script>
       if('Notification' in window)
       {
           Notification.requestPermission(function(perm){
                if(perm == "granted"){
                    var note = new Notification("Hello from your OS!");
                }
           });
       }
       else{
           document.getElementById("not1").innerHTML = "Notifications not supporte
d";
       }
    </script>

</body>
</html>
```