Events & Ajax

Make it interactive

Review Variables

- Scope – function or global
- avoid "hoisting" by defining at the top
- think of the file as a unified JavaScript application, instead of a bunch of one-off actions

Review Functions

Defining a function:

```
function sayHello() {
   alert("Hello");
}
```

Calling a function:

```
sayHello();
```

Anonymous Functions

- Have no name (anonymous)
- Generally they either:
  - run immediately (IIFE)
  - are connected to events
  - are methods in an object

Object data & methods

Functions in an object

```
var cat = {
   name: "Bart",
   meow: function() {
      alert("Meow");
   }
}
```

Getting object data

Refer to the object first, then the property:

```
cat.name;    // not just name
```

Calling a method

Refer to the object first, then the method:

```
cat.meow();   // not just meow()
```

Event Handlers

These have popped up before:

```
myButton.onclick = function() {
  alert(“You clicked on me!”);
}
```

Event Handlers

- Generally use anonymous functions.
- Name is all lower case, on + event:
  - onclick, onblur, onfocus, etc.
  - Be careful **not to** use onClick, onBlur, etc
  - This does not cause an error, but does not handle event either. I see this mistake often.
- Limitation: There is only one per event.

Event Listeners

These have also popped up before:

```
myButton.addEventListener(“click”,
    functionName, false);
```

Note: do not put () after the function name. Why not?

Event Listeners

- Generally use a named function, but may also use anonymous functions.
- First parameter is the event
- Second is the function name or anonymous function definition
- Third is “capture” which is almost always set to false.

Pros & Cons of Listeners

The key benefit of listeners is that you can have more than one per event.

Drawback is no support in IE 8 and earlier.

If you use listeners, you check for support and fallback with the attachEvent method.

Looking Ahead

You need to have a basic understanding of out-of-the-box JavaScript event handlers and listeners.

BUT - libraries such as jQuery are often used to do this in a cross-browser supported way.

Address Book Example

Listing 7.4.8 - updated Address Book

Example of defining a “application” object with various methods.

The application object may also have data inside of it, but here it is outside.

Ajax

This is where things can start to get fun.

Using Ajax, we can get data from all sorts of interesting places to show on our pages.

Weather, Maps, Pictures, Videos, News, Social Media, Blogs, Comics…

Ajax

It is a way to get data into *part* of your HTML page, using JavaScript, without a complete request - response cycle to load an entire new page from the server.

It often uses JSON for that data, but may also use other formats.

How it works

An HTTP request is initiated by JavaScript.

When the data is returned, JavaScript is used to update the DOM with the new data.

This allows some content on the page to change, without reloading the entire page.

How it works

Usually it is asynchronous, meaning other things can happen while you wait for your data to come back.

Can also be synchronous (or blocking) meaning nothing else happens while you wait.

Getting the HttpRequest

The XMLHtttpRequest is the start of Ajax requests.

Of course IE has to be different, so you must check for support first and react accordingly.

```
function getHTTPObject() {
    var xhr;

    if (window.XMLHttpRequest) {
        xhr = new XMLHttpRequest();
    }
    else if (window.ActiveXObject) {
        xhr = new ActiveXObject("Msxml2.XMLHTTP");
    }
    return xhr;
}
```

Cross-browser support for getting HttpRequest

Making the Ajax call

Once you have the HttpRequest, you can use it to make the call.

Can be GET or POST:

- GET performs better, uses URL parameters
- POST is more secure

Open the Request

The first step is to open the request:

```
    request.open("GET", fileOrUrl, true);
```

First parameter is GET or POST
Next parameter is the request URL
Last parameter is for asynchronous or not

Send the Request

Next you send the request, including any data needed, to the server.

```
request.send("null or optional string data");
```

Handler for Response

Then you wait:

```
request.onreadystatechange = function () {
   var text;
   if (request.readyState === 4 && request.status === 200) {
        text = request.responseText;
        data = JSON.parse(text);
        showData(data);  // Some function to generate HTML
    }
}
```

```
function loadAjaxData( fileOrUrl ) {
    var request = getHTTPObject();

    request.open("GET", fileOrUrl, true);
    request.send(null);

    request.onreadystatechange = function () {
        var text;
        if (request.readyState === 4 && request.status === 200) {
            text = request.responseText;
            data = JSON.parse(text);
            showData(data);  // Some function to generate HTML
        }
    }
}
```

Making the Ajax call and handling data

Check for errors

Server status of 200 is good.

Server status for possible errors:

- 404 - Page not found
- 500 - Internal Server Error
- 403 - Forbidden (access error)

Homework

Part 2 of the First-App project. Will do more detailed presentations next week.

Read chapter 9 on code organization and apply what you learn to your code.