INTRODUCTION TO JAVASCRIPT

Aubrey Drescher

ATX GIS Day

Free yearly event to learn about GIS and connect with other professionals

- Technical Training
- Map/App Gallery
- Panel Discussions
- Lightning Talks

www.gisday.com
atxgisday.org

What we'll be doing today

- **Introductions:** A little about me, a little about you

  Please share:

  - Your name
  - Past experience with programming
  - What you hope to get out of this class

- **Preface:** What is JavaScript and why is it good to know?
- **Lecture:** Explanations and demos of basic JavaScript concepts and syntax
- **Workshop:** Build a photo browser website

What is JavaScript?

- JavaScript is a scripting language that can be inserted into any HTML page, and executed by all types of web browsers.
- JavaScript first appeared in Netscape (RIP) in 1995.
- JavaScript is now the world's most popular programming language.

What can we use JavaScript for?

- Make websites respond to user interaction
- Build apps and games
- Access information on the Internet
- Organize and present data

JavaScript is easy to learn.

Why is JavaScript good to know?

From ESRI's Road Map for Web Developers, Feb 2014:

- Advances in modern browser technology combined with limited browser support for Flex and Silverlight, encourage the use of JavaScript/HTML5 for web GIS implementations.
- JavaScript/HTML5 has become the technology of choice among our user community for web GIS solutions going forward.

From Dave Bouwman's blog post "Transitioning to JavaScript":

- Over the last few years JavaScript has matured as a language and as a community.
- The community is exploding, and it seems every day there is some new exciting project in JavaScript.

Change the Content of HTML Elements

- The HTML DOM (**D**ocument **O**bject **M**odel) is the official W3C standard for accessing HTML elements.
- It is very common to use JavaScript to manipulate the DOM (to change the content of HTML elements).
- **document.getElementById()** is one of the most commonly used HTML DOM methods.

Example

Find the HTML element with id="demo":

```
x = document.getElementById("demo");
```

Change the content of the HTML element:

```
x.innerHTML = "Hello JavaScript";
```

Js.Fiddles

Js.Fiddle is a website that lets you experiment with JavaScript, HTML and CSS code.

http://aubreyrhea.github.io/swiggis/jsfiddles.html

The <script> Tag

- In HTML, JavaScripts must be inserted between <script> and </script> tags.
- Scripts can also be placed in external files.
- External JavaScript files have the file extension .js.
- To use an external script, put the name of the script file in the source (src) attribute of the <script> tag:

```
<html>
<body>
<script src="myScript.js"></script>
</body>
</html>
```

- You can place an external script reference in <head> or <body> as you like.

Changing the Value of an Attribute

To change the value of an HTML attribute, use this syntax:

```
document.getElementById(id).attribute=new value
```

This example changes the value of the src attribute of an <img> element:

```
<html>
<body>

<img id="image" src="smiley.gif">

<script>
document.getElementById("image").src="landscape.jpg";
</script>

</body>
</html>
```

JavaScript Functions

- A JavaScript function is defined with the **function** keyword, followed by a ***functionName***, and single brackets: **()**
- The single brackets may include a list of parameter names: **(*parameter1, parameter2, ...*)**
- The code to be executed by the function is placed inside curly brackets: **{}**

```
function functionName(parameters) {
  code to be executed
}
```

A JavaScript function is a block of code that may be executed when "someone" invokes (calls) it.

JavaScript Events

- A JavaScript can be executed when an event occurs, like when a user clicks on an HTML element. To execute code when a user clicks on an element, add JavaScript code to an HTML event attribute.

```
<button onclick="displayDate()">Try it</button>
```

- The onload and onunload events are triggered when the user enters or leaves the page.

```
<body onload="checkCookies()">
```

- The onchange event are often used in combination with validation of input fields.

```
<input type="text"
id="fname" onchange="upperCase()">
```

- The onmouseover and onmouseout events can be used to trigger a function when the user mouses over, or out of, an HTML element.

JavaScript Variables

- JavaScript variables are "containers" for storing information.

```
var x=5;
var y=6;
var z=x+y;
```

- JavaScript variables can also hold other types of data, like text values.

```
var person="John Doe";
```

- Creating a variable in JavaScript is most often referred to as "declaring" a variable. You declare JavaScript variables with the **var** keyword

```
var carname;
```

- After the declaration, the variable is empty (it has no value). To assign a value to the variable, use the equal sign:

```
carname="Volvo";
```

- You can also assign a value to the variable when you declare it

```
var carname="Volvo";
```

JavaScript Data Types

String, Number, Boolean, Array, Object, Null, Undefined

- JavaScript Strings store a series of characters.
```
var answer="He is called 'Johnny'";
```

- JavaScript Numbers can be written with, or without decimals.
```
var x=34.00;
```

- JavaScript Booleans can only have two values: true or false.
```
var x=true;
```

- JavaScript Arrays store lists of data
```
var cars=["Saab","Volvo","BMW"];
```

- JavaScript Objects store name and value pairs
```
var person={firstname:"John", lastname:"Doe", id:5566};
```

- **Undefined** is the value of a variable that has not been assigned a value. Variables can be emptied by setting the value to **null**;

- You can use the JavaScript function typeof() to find the type of a variable.
```
typeof(3.14);     returns number
```

Function Parameters and Arguments

- When you call a function, you can pass values to it. These values are called arguments or parameters.
- Identifiers used in the function definition are called **parameters**.
- Multiple parameters are separated by commas.

Example function being defined:

```
function myFunction(parameter1, parameter2) {
  code to be executed
}
```

- Values used for the parameters when the function is invoked are called **arguments**.
- The parameters and the arguments must be in the same order.
- Inside the function, the arguments can be used as variables.

Example function being called:

```
var x = myFunction(argument1, argument2);
```

The Return Statement

- Functions often compute a **return value**. This way a function can return a value back to the "caller".
- When JavaScript reaches a **return statement** inside a function, the function will stop executing.

Example: Calculate the product of two numbers, and return the result:

```
function myFunction(a, b) {
  return a * b;
}

document.getElementById("demo").innerHTML = myFunction(4,
3);
```

The innerHTML of the "demo" element will be:

12

JavaScript Timing Events

- With JavaScript, it is possible to execute some code at a specified time delay.
- setTimeout() - executes a function, once, after waiting a specified number of milliseconds

```
setTimeout("javascript function",milliseconds);
```

Wait 3 seconds, then alert "Hello":

```
setTimeout("alert('Hello')",3000);
```

JavaScript Objects

- In JavaScript, objects are complex variables made of name and value pairs
- Objects can be thought of as an unordered collection of properties

This example creates an object called "person", and adds four properties to it:

```
var person = {firstName:"Tom", lastName:"Smith",
age:40, eyeColor:"Blue"};
```

You can access the object properties in two ways:

```
name = person.lastName;
name = person["lastName"];
```

For…in statement
Looping Through an Object

A useful way to systematically access every element in an object is to use a for loop.

Its syntax is:

```
for (key in objectName) {
        // some code
}
```

```
var cities = {0:"Dublin", 1:"London", 2:"Paris", 3:"NYC"};

for (var i in cities) {
      alert("I would like to visit " + cities[i]);
}
```

Control Flow (if/then)

- You can change the flow of the code's execution using an if statement.
- The use of the if keyword signals the beginning of a conditional test.
- The test is inside of the parenthesis, and the statements that will be executed if the test returns true are inside the curly braces.
- If you want code to execute when the if statement is false, you can use the else statement.

```
if (Some condition) {
    // Do something
} else if (Some other condition) {
    // Do something else
} else {
    // Do a third thing
}
```

References & Resources

- ESRI's Road Map for Web Developers: http://blogs.esri.com/esri/arcgis/2014/02/21/esris-roadmap-for-web-developers/
- Dave Bouwman's blog post "Transitioning to JavaScript": http://blog.davebouwman.com/2014/03/03/transitioning-to-javascript/
- Code Academy: http://www.codecademy.com/tracks/javascript
- Mozilla Developer Network JavaScript Guide: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide
- Eloquent JavaScript: http://eloquentjavascript.net/contents.html

Workshop Introduction

- Today you will be building a simple web page that provides multiple ways to browse through a collection of photos.
- In the second half of the class you will add a map to this web page that shows all the locations where the photos were taken.
- Inspiration for the web page content courtesy of the City of Austin Capitol View Corridors map ftp://ftp.ci.austin.tx.us/GIS-Data/Regional/standard_maps/Capitol_View_Corridors.pdf
- Preview of the final product http://aubreyrhea.github.io/swiggis/

Workshop Preparation

- If you have not yet, download and install Adobe Brackets from http://www.brackets.io/
- If you don't have the Chrome browser, download and install it from https://www.google.com/intl/en-US/chrome/browser/
- Create a new folder on your hard drive. Then download the starter source code files from https://github.com/AubreyRhea/swiggis/blob/master/starter.zip?raw=true and unzip them into the folder.
- In Adobe Brackets, go to File > Open Folder and select the folder where you saved the source code

Populate the Drop Down box

- A drop down box option is created using the syntax box[option position] = new Option[option text, option value]

- In this code, box.length grows by 1 each time a new option is added, so box[box.length] is always equal to the last option index.

- The variable i also grows by 1 each time the loop runs, so it can be used to reference each member of the images object in order.

```
function PageLoad() {
    var box = document.getElementById("PhotoSelectBox");
    for (var i in images) {
        box[box.length] = new Option(images[i].title, i);
    }
}
PageLoad();
```

Load the Selected Photo

Type this code right before the PageLoad(); function:

```
function MoveSelectedPhoto(newPhotoIndex) {
  UpdateSelectedPhoto(newPhotoIndex);
}

function UpdateSelectedPhoto(newPhotoIndex) {
     var photo = document.getElementById("Photo");
     photo.src = images[newPhotoIndex].path;
}

document.getElementById("PhotoSelectBox").onchange =
  function () { MoveSelectedPhoto(this.value); }
```

Change Photo Dimensions

Type this code inside the UpdateSelectedPhoto(newPhotoIndex) function:

```
if (newPhotoIndex == 8) {
    photo.width = "510";
    photo.height = "737";
} else {
    photo.width = "737";
    photo.height = "510";
}
```

Move First and Last

Type this code after the UpdateSelectedPhoto(newPhotoIndex) function:

```
function MoveFirst() { MoveSelectedPhoto(0); }
function MoveLast() {
    MoveSelectedPhoto(Object.keys(images).length - 1);
}
```

Type this code after the ("PhotoSelectBox").onchange event handler:

```
document.getElementById("FirstButton").onclick =
function () { MoveFirst(); }

document.getElementById("LastButton").onclick =
function () { MoveLast(); }
```

Move Forward and Back

```
function MovePrevious() {
    var box = document.getElementById("PhotoSelectBox");
    var oldPhotoIndex = box.selectedIndex;
    MoveSelectedPhoto(oldPhotoIndex - 1);
}
function MoveNext() {
    var box = document.getElementById("PhotoSelectBox");
    var oldPhotoIndex = box.selectedIndex;
    MoveSelectedPhoto(oldPhotoIndex + 1);
}

document.getElementById("NextButton").onclick =
function () { MoveNext(); }
document.getElementById("PreviousButton").onclick =
  function () { MovePrevious(); }
```

Change Selected Item in Drop Down

Type this code inside the UpdateSelectedPhoto(newPhotoIndex) function:

```
var box = document.getElementById("PhotoSelectBox");
box.selectedIndex = newPhotoIndex;
```

Enable Navigation Buttons

Type this code below the MoveNext() function:

```
function EnableNavigationButtons() {
    var firstButton = document.getElementById("FirstButton");

    var previousButton =
  document.getElementById("PreviousButton");

    var nextButton = document.getElementById("NextButton");

    var lastButton =
document.getElementById("LastButton");

    firstButton.disabled = false;
    previousButton.disabled = false;
    nextButton.disabled = false;
    lastButton.disabled = false;
}
```

Selectively Disable Navigation Buttons

Type this code inside the EnableNavigationButtons() function:

```
var box = document.getElementById("PhotoSelectBox");
var photoIndex = box.selectedIndex;

if (photoIndex == 0) {
    firstButton.disabled = true;
    previousButton.disabled = true;
} else if (photoIndex == Object.keys(images).length - 1) {
    nextButton.disabled = true;
    lastButton.disabled = true;
}
```

Type this code inside the UpdateSelectedPhoto(newPhotoIndex) function:

```
EnableNavigationButtons();
```

Make a Slideshow!

Type this code after the EnableNavigationButtons() function:

```
var slideIndex = 0;
function PlaySlideshow() {
    MoveSelectedPhoto(slideIndex);
    slideIndex++;
    if (slideIndex >= Object.keys(images).length) {
        slideIndex = 0;
        return;
    }
    setTimeout(function () { PlaySlideshow(); }, 2000);
}

document.getElementById("SlideshowButton").onclick =
 function () { PlaySlideshow(); }
```