Chapter 3: Introduction to HTML, JavaScript, and CSS

There are certain fundamental concepts that you need to understand before you can get started with developing GIS applications with the ArcGIS Server API for JavaScript. For those of you already familiar with HTML, JavaScript, and CSS you may wish to skip ahead to the next chapter. However, if you're new to any of these concepts read on. We are only going to cover these topics at a very basic level. Just enough to get you started. For a more advanced treatment of any of these subjects there are many learning resources available including books and online tutorials. Please consult Appendix I for a more comprehensive list of these resources.

Basic HTML Page

Before we dive into the details of creating a map and adding layers of information you need to understand the context of where the code will be placed when you're developing applications with the API for JavaScript. The code you write will be placed inside an HTML page or JavaScript file. HTML files typically have a file extension of .html or .htm and JavaScript files have an extension of.js. Once you have created a basic HTML page you can then go through the necessary steps to create a basic map with the ArcGIS Server API for JavaScript.

The core of a web page is an HTML file. Coding this basic file is quite important as it forms the basis for the rest of your application. Mistakes that you make in the basic HTML coding can result in problems down the line when your JavaScript code attempts to access these tags.

Below is a code example for a very simple HTML page. This example is about as simple as an HTML page can get. It contains only the primary HTML tags <DOCTYPE>, <html>, <head>, <title>, and <body>.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

    <title> Topographic Map</title>

  </head>

  <body>
    Hello World
  </body>

</html>
```

There are a number of flavors of HTML currently in used. The most common are HTML 4.01 (seen in the code example above) and XHTML 1.0. In addition, XHTML 1.1 and the new HTML 5 are also in use. The new HTML 5 is getting a lot of press and you'll likely see this implementation being used more and more.

DOCTYPE

The first line of your HTML page will contain the DOCTYPE. This is used to tell the browser how the HTML should be interpreted. The two most common DOCTYPEs are HTML 4.01 Strict and XHTML 1.0 Strict:

HTML 4.01 Strict

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

XHTML 1.0 Strict

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Primary Tags

At a minimum all your web pages will need to contain the <html>, <head>, and <body> tags. The <html> tag defines the whole HTML document. All other tags must be placed inside this tag. Tags that define how the web page will appear in the browser are placed inside the <body> tag. For instance, your mapping applications will contain a tag inside the <body> tag that is used as a container for displaying the map.

Loading this HTML page in a browser would produce the content you see in the figure below. Most of the ArcGIS Server API for JavaScript code that you write will be placed between the <head> tags and within a <script> tag or inside a separate JavaScript file. As you gain experience you will likely begin placing your JavaScript code inside one or more JavaScript files and then referencing them from the <head> section. We'll explore this topic later. For now just concentrate on placing your code inside the <head> tags.

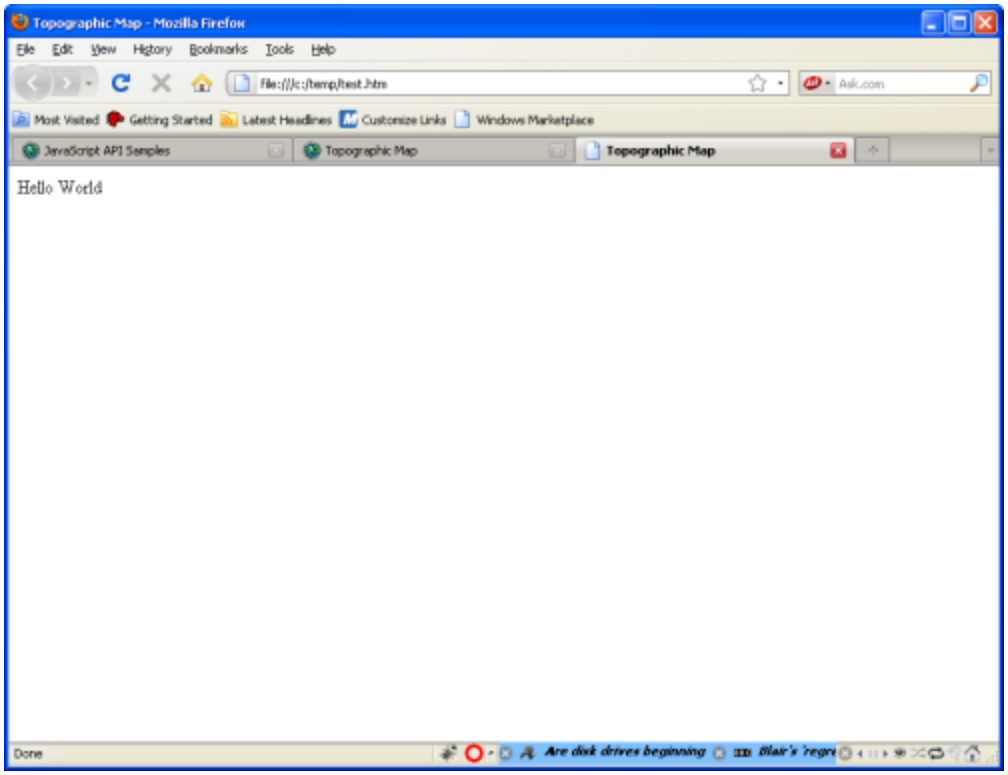

Figure 12: Hello World

Validating HTML Code

I've mentioned that it is very important that your HTML tags be coded correctly. This is all well and good you say, but how do I know my HTML has been coded correctly? Well, there are a number of HTML code validators that you can use to check your HTML. The W3C HTML validator (http://validator.w3.org/) shown in the figure below can be used to validate HTML code by URI, file upload, or direct input.

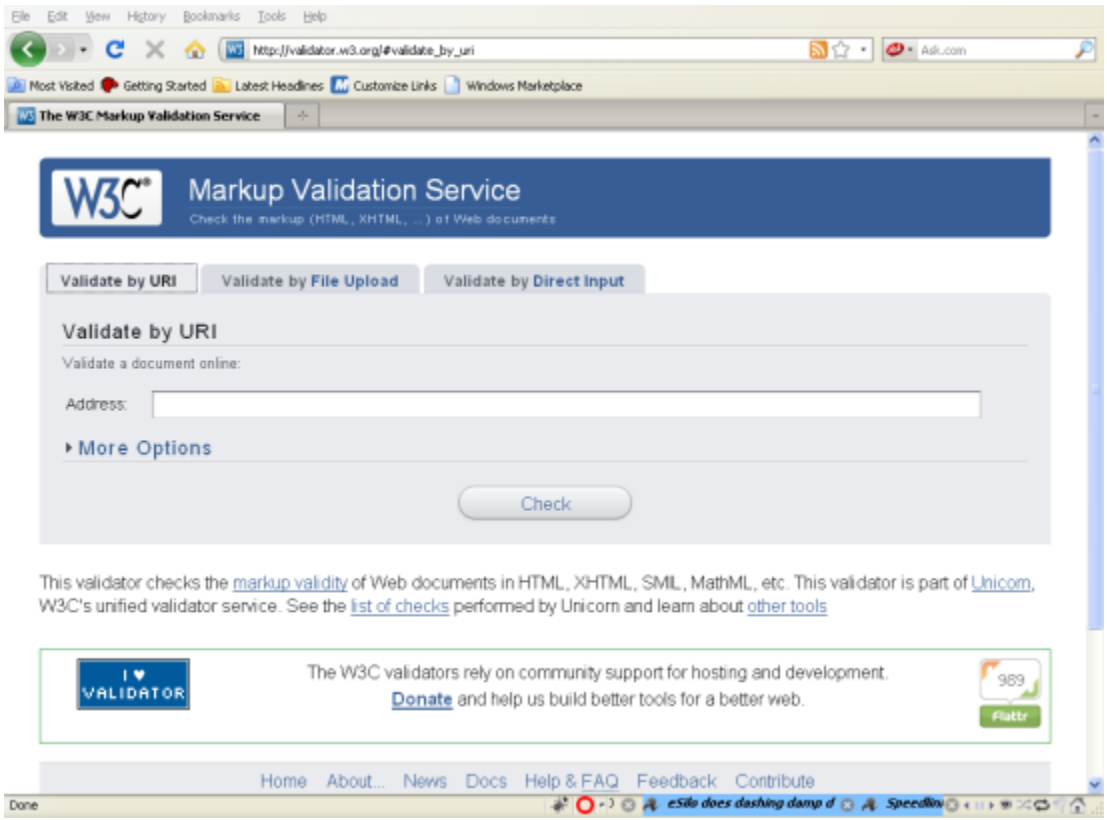

Figure 13: W3C Validator

Assuming that your HTML code successfully validates you will get a nice screen with a green message indicating a successful validation.

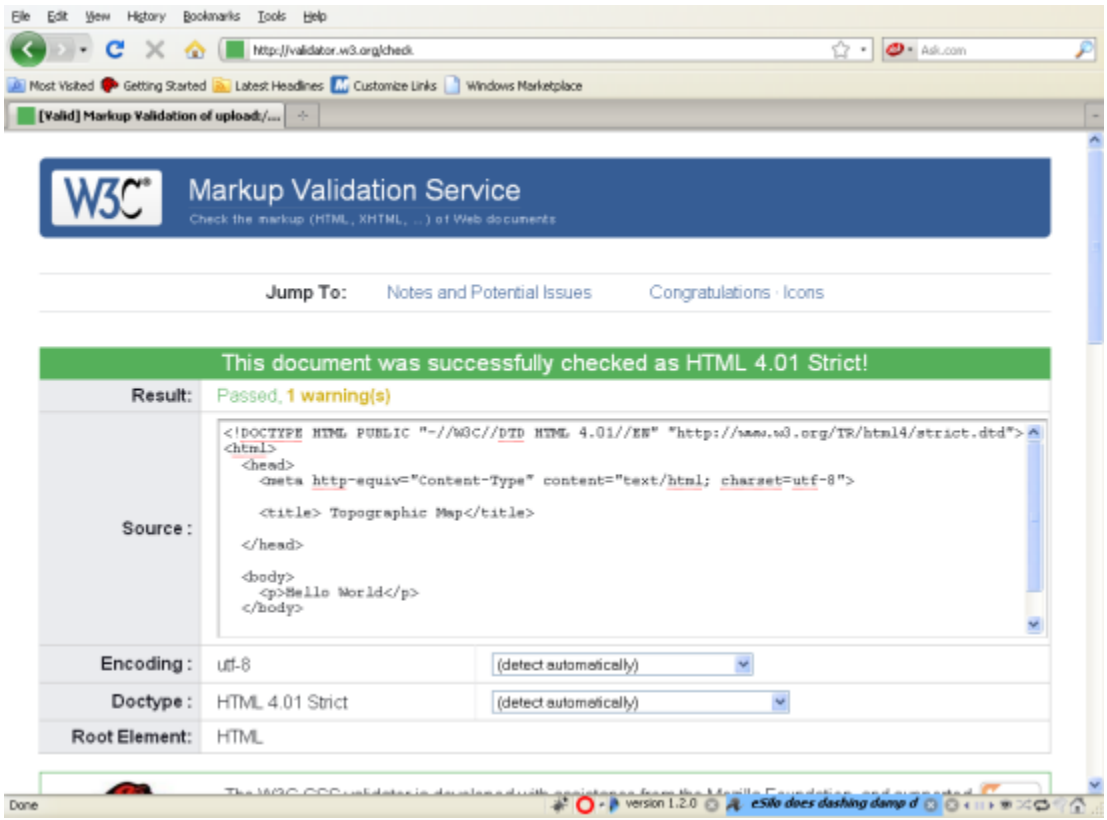

Figure 14: Successful Validation

On the other hand it will identify any problems with a red error message. Errors are described in detail which makes it easier to correct problems. Often a single error can lead to many other errors so it is not uncommon to see a long list of error items. Don't panic if this is the case. Fixing one error often resolves many others.

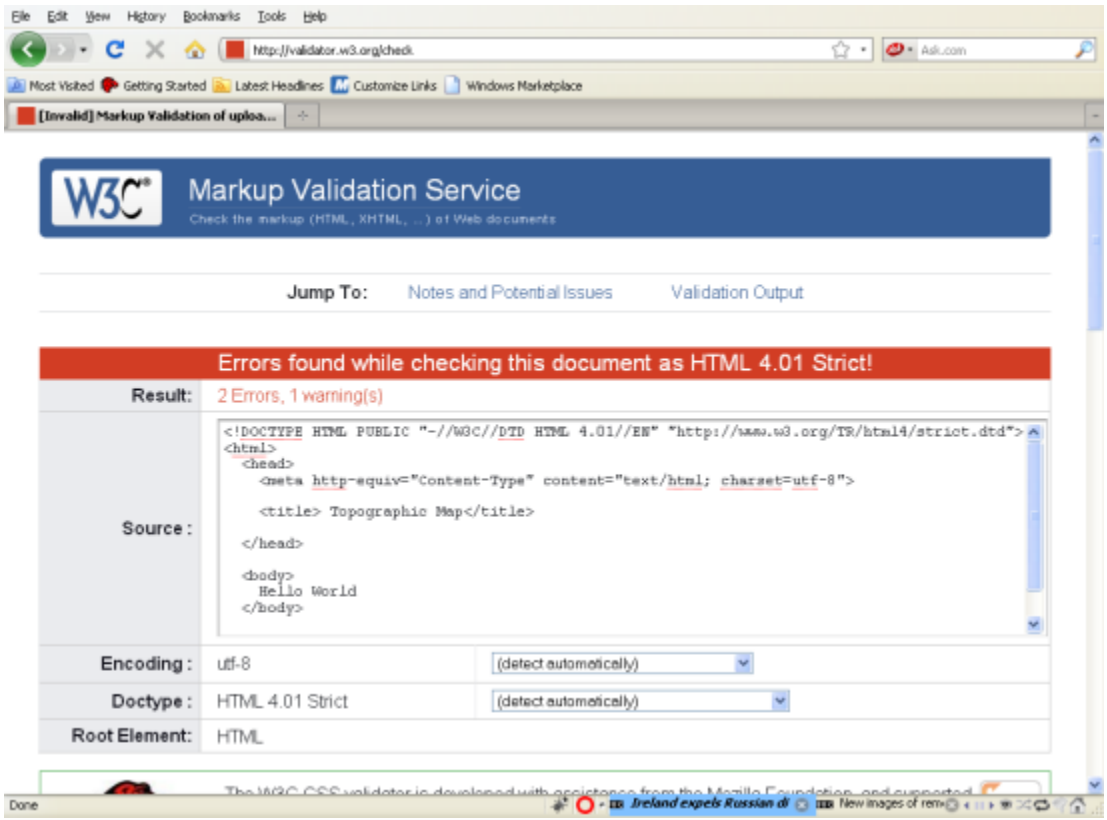

Figure 15: Validation Error

JavaScript Fundamentals

As the name implies, the ArcGIS Server API for JavaScript, requires that you use the JavaScript language when developing your application. There are some fundamental JavaScript programming concepts that you will need to know before starting to build your application.

JavaScript is a lightweight scripting language that is embedded in all modern web browsers. Although JavaScript can certainly exist outside the web browser environment in other applications it is most commonly known for its integration with web applications. All modern web browsers including Internet Explorer, Firefox, and Chrome have JavaScript embedded. The use of JavaScript in web applications gives the ability to create dynamic applications that do not require round trips to the server to fetch data, and thus the applications are more responsive and natural to the user. However, JavaScript does have the capability of submitting requests to the server, and is a core technology in the AJAX stack. One common misconception regarding JavaScript is that it is actually a simplified version of Java. The two languages are actually unrelated with the exception of the name.

Variables

Variables are a fundamental concept that you need to understand when working with any programming language. Variables are simply names that we use to associate with some type of

data value. At a lower level, these variables are areas of space carved out in a computer's memory that store data.

You can think of a variable as a box that has a name and which contains some sort of data. When we initially create the variable it is empty until data is assigned. Basically, variables give us the ability to store and manipulate data. In the figure below we create a variable called 'ssn'. Initially this variable is empty but is then assigned a value of '467-63-6732'. The data value assigned to a variable can be of various types including numbers, strings, booleans, objects, and arrays.

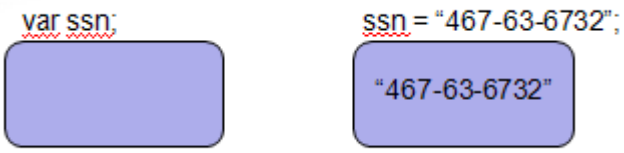

Figure 16: Creating a Variable

In JavaScript, variables are declared with the 'var' keyword. In general, the names that you assign to your variables are completely up to you. However, there are certain rules that you need to follow when creating a variable. Variables can contain both text and numbers but should never start with a number. Always start your variable name with a letter or underscore. In addition, spaces are not allowed within variable names nor are special characters such as percent signs and ampersands. Other than that you are free to create variable names as you wish but you should try to assign variable names that describe the data that the variable will be assigned. It is also perfectly legal to declare multiple variables with the same 'var' keyword, and you can also combine variable declaration with data assignment as seen in the examples seen below.

```
var i = 10;
var j = 20;
var k = 30;
```

JavaScript is Case-Sensitive

One very important point that I need to make is that JavaScript is a case-sensitive language and you need to be very careful about this because it can introduce some difficult to track down bugs in your code. All variables, keywords, functions, and identifiers must be typed with a consistent capitalization of letters. This gets even more confusing when you consider that HTML is not case sensitive. This tends to be a stumbling block for new JavaScript developers. Below I have created three variables, all with the same spelling, but because they do not follow the same capitalization pattern you end up with three different variables.

```
var myName = 'Eric';
var myname = 'Eric';
var MyName = 'Eric';
```

Variable Data Types

JavaScript supports various types of data that can be assigned to your variables. Unlike other strongly typed languages like .NET or C++, JavaScript is a loosely typed language. What this means is that you don't have to specify the type of data that will occupy your variable. The JavaScript interpreter does this for you on the fly. You can assign strings of text, numbers, boolean true/false values, arrays, or objects to your variables.

Numbers and strings are pretty straight-forward for the most part. Strings are simply text enclosed by either a single or double quote. For instance:

```
var baseMapLayer = "Terrain";
var operationalLayer = 'Parcels';
```

Numbers are not enclosed inside quote marks and can be integers or floating point numbers.

```
var currentMonth = 12;
var layered = 3;
var speed = 34.35;
```

One thing I would point out to new programmers is that numeric values can be assigned to string variables through the user of single or double quotes that enclose the value. This can be confusing at times for some new programmers. For instance, a value of 3.14 without single or double-quotes is a numeric data type while a value of "3.14" with single or double quotes is assigned a string data type.

Other data types include booleans which are simply true or false values, and arrays which are a collection of data values. An array basically serves as a container for multiple values. For instance, you could store a list of geographic data layer names within an array and access them individually as necessary.

```
var male = false;
var female = true;
```

Objects deserve special consideration and we'll examine them in a lot greater detail as we advance through the book. Basically, objects are named pieces of data that have both properties and methods. Objects that are part of the ArcGIS Server API for JavaScript are stored in object variables.

Functions

Now let's cover the very important topic of functions. Functions are simply named blocks of code that execute when called. The vast majority of the code that you write in this course and in your development efforts will occur within functions that you define.

Best practice calls for you to split your code into functions that perform small, discrete units of operation. These blocks of code are normally defined in the <head> section of a web page inside a <script> tag, but can also be defined in the <body> section. However, in most cases you will want your functions defined within the <head> section so that you can ensure that they are available once the page has loaded.

To create a function you need to use the function keyword follow by a function name that you define and any variables necessary for the execution of the function passed in as variables. In the event that you need your function to return a value to the calling code you will need to use the return keyword in conjunction with the data you want passed back.

```
function prod(a,b)
{
   x = a * b;
   return x;
}
```

Objects

Now that we've gone through some basic JavaScript concepts we'll tackle the most important concept in this section. In order to effectively program mapping applications with the ArcGIS Server API for JavaScript you need to have a good fundamental understanding of objects so this is a critical concept that you need to grasp to understand how to develop web mapping applications.

The ArcGIS Server API for JavaScript makes extensive use of objects. We'll cover the details of both programming libraries, but for now we'll hit the high level concepts. Objects are complex structures capable of aggregating multiple data values and actions into a single structure. This differs greatly from our primitive data types like numbers, strings, and booleans which can hold only a single value. Objects are much more complex structures.

Objects are composed of both data and actions. Data, in the form of properties, contains information about an object. For example, with a Map object found in the ArcGIS Server JavaScript API there are a number of properties including the map extent, graphics associated with a map, the height and width of the map, layer ids associated with the map and more. These properties contain information about the object.

Objects also have actions which we typically think of as methods, but we can also group constructors and events into this category. Methods are actions that a map can perform such as adding a layer, setting the map extent, or getting the map scale.

<OMD figure here>

Constructors are special purpose functions that are used to create new instances of an object. With some objects it is also possible to pass parameters in to the constructor to give more control over the object that is created. Events are actions that take place on the object and are triggered by the end user or the application. This would include events such as a map click, mouse move, or a layer being added to the map.

Properties and methods are accessed via dot notation wherein the object instance name is separated from the property or method by a dot. For instance, to access the current map extent you would enter map.extent in your code. The same is the case with methods except that methods have parentheses. When data is passed into a method via parameters they are placed inside the parentheses.

```
var theExtent = map.extent;
var graphics = map.graphics;

map.centerAt(pt);
map.panRight();
```

Basic CSS Principles

CSS or Cascading Style Sheets is a language used to describe how HTML elements should be displayed on a web page. For instance, CSS is often used to define common styling elements for a page or set of pages such as the font, background color, font size, link colors, and many other things related to the visual design of a web page.

```
body
{
font-size:75%;
font-family:verdana,arial,'sans serif';
background-color:#FFFF0;
color:#000080;
margin:10px;
}

h1 {font-size:200%;}
h2 {font-size:140%;}
h3 {font-size:110%;}

th {background-color:#ADD8E6;}

ul {list-style:circle;}
ol {list-style:upper-roman;}

a:link {color:#000080;}
a:hover {color:red;}
```

CSS Syntax

A CSS rule has two main parts; a selector and one or more declarations. The selector is typically the element that you want to style. In the example below, the selector is 'p'. A 'p' element in HTML represents a paragraph. The second part of a CSS rule is one or more declarations each of which consists of a property and a value. The property represents the style attribute that you want to change. In our example, we are setting the 'color' property to red. In effect what we have done with this CSS rule is to define that all text within our paragraph should be red.

p {color:red}

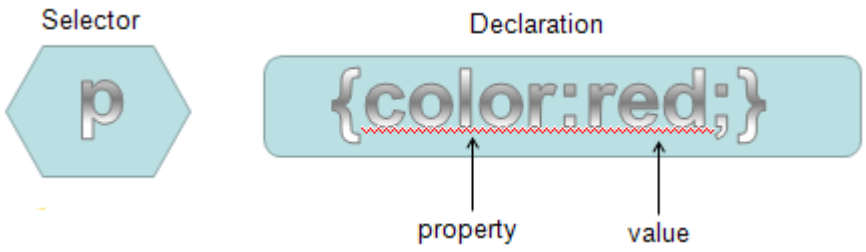

Figure 17: CSS Syntax

You can include more than one declaration in a CSS rule as you see in the example below. A declaration is always surrounded by curly brackets and each declaration ends with a semi-colon.

In addition, a colon should be placed between the property and the value. In this particular example two declarations have been made. One for the color of the paragraph and another for the text alignment of the paragraph. Notice that the declarations are separated by a semi-colon.

p {color:red;text-align:center}

CSS comments are used to explain your code. You should get into the habit of always commenting your CSS code just as you would with any other programming language. Comments are always ignored by the browser. On this slide you see an example of how a comment is defined in CSS. Comments begin with a slash followed by an asterisk and end with an asterisk followed by a slash. Everything in between is assumed to be a comment and is ignored.

```
/* This is a comment */

h1 {font-size:200%;}
h2 {font-size:140%;}
h3 {font-size:110%;}
```

In addition to specifying selectors for specific HTML elements you can also use the 'id' selector to define styles for any HTML elements with an 'id' value that matches the 'id' selector. An 'id' selector is defined in CSS through the use of the #pound sign followed by an 'id' value.

For example, in the code example below you see three 'id' selectors: rightPane, leftPane, and map. In ArcGIS Server API for JavaScript applications you almost always have a map. When you define a tag that will serve as the container for the map you define an 'id' and assign it a value which is often 'map'. In this case we are using CSS to define several styles for our map including a margin of 5 pixels along with a solid styled border of a specific color and a border radius.

```
#rightPane {
    background-color:white;
    color:#3f3f3f;
    border: solid 2px #224a54;
    width: 20%;
}
#leftPane {
    margin: 5px:
    padding: 2px;
    background-color:white;
    color:#3f3f3f;
    border: solid 2px #224a54;
    width: 20%;
}
#map {
     margin: 5px;
     border: solid 4px #224a54;
     -moz-border-radius: 4px;
}
```

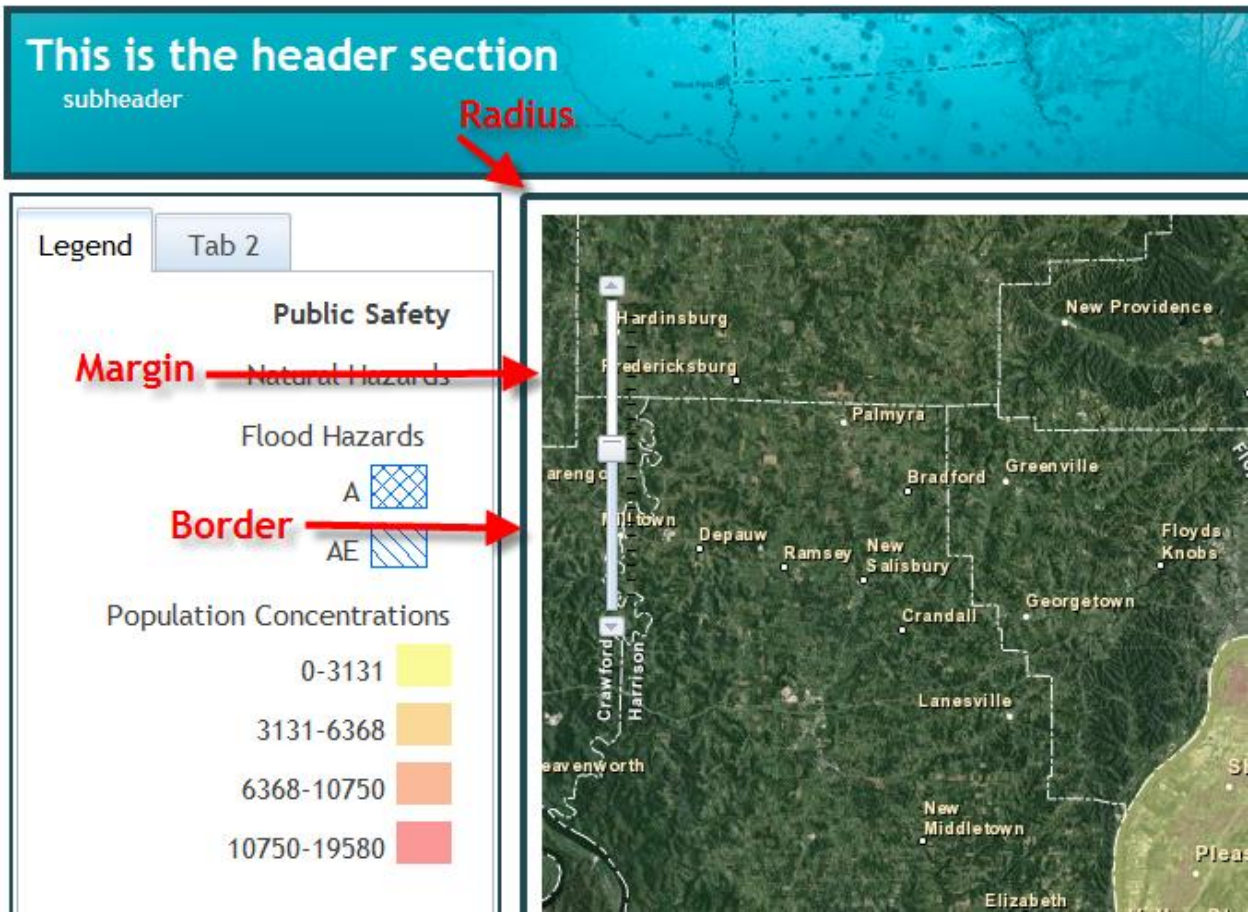

Figure 18: Map Border Styling

Unlike 'id' selectors which are used to assign styles to a single element, 'class' selectors are used to specify styles for a group of elements all of which have the same HTML class attribute. A class selector is defined with a period followed by the class name. You may also specify that only specific HTML elements with a particular class should be affected by the style. Examples of both are shown in the code example below.

```
.center {text-align:center;}
p.center {text-align:center;}
```

There are three ways to insert CSS into your application: external style sheets, internal style sheets, and inline.

External Style Sheet

An external style sheet is simply a text file containing CSS rules and saved with a file extension of .css. This file is then linked into all web pages that want to implement the styles defined within the external style sheet through the use of the HTML <link> tag. This is a commonly

used method for splitting out the styling from the main web page and gives you the ability to change the look of an entire website through the use of a single external style sheet.

Internal Style Sheet

An internal style sheet moves all the CSS rules into a specific web page. Only HTML elements within that particular page have access to the rules. All CSS rules are defined inside the <head> tag and enclosed inside a <style> tag as seen in the code example below.

```
<head>
<style type="text/css">
hr {color:sienna;}
p {margin-left:20px;}
body {background-image:url("images/back40.gif");}
</style>
</head>
```

Inline Styling

The final method of defining CSS rules for your HTML elements is through the use of inline styles. This method is not recommended because it mixes style with presentation and is difficult to maintain. It is an option though in some cases where you only need to define a very limited set of CSS rules. To use inline styles simply place the 'style' attribute inside the relevant HTML tag.

```
<p style="color:sienna;margin-left:20px">This is a paragraph.</p>
```

Now let's put some emphasis on the "cascading" of cascading style sheets. As you now know, styles can be defined in external style sheets, internal style sheets, or inline. There is a fourth level that we didn't discuss which is the browser default. You don't have any control over that though. In CSS, an inline style has the highest priority, which means that it will override a style defined in an internal style sheet, an external style sheet, or the browser default. If an inline style is not defined then any style rules defined in an internal style sheet would take precedence over styles defined in an external style sheet. The caveat here is that if a link to an external style sheet is placed after the internal style sheet in HTML <head>, the external style sheet will override the internal sheet! That's a lot to remember! Just keep in mind that style rules defined further down in the hierarchy override style rules defined higher in the hierarchy.

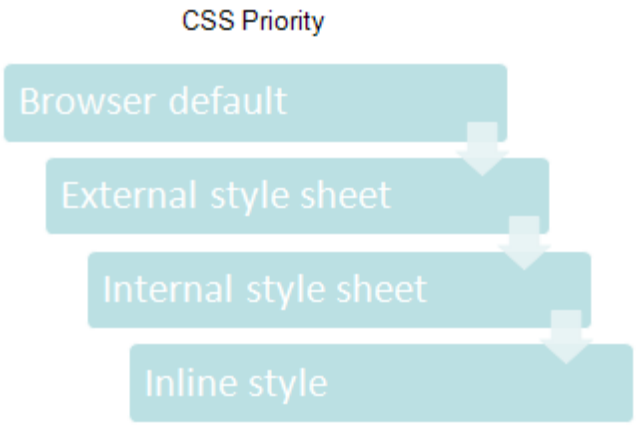

Figure 19: CSS Priority

These are the basic concepts that you need to understand with regard to CSS. You can use CSS to define styles for pretty much anything on a web page including backgrounds, text, fonts, links, lists, images, tables, maps, and any other visible objects.

Note: For a more in-depth treatment of the ArcGIS Server API for JavaScript please refer to our Building Custom ArcGIS Server Applications with JavaScript. This course is offered as both a traditional instructor led course and an instructor guided online course. For more information on this course please visit our website at geospatialtraining.com