

JavaScript

CS/IT 490 WD, Fall 2013

Last update 2013-11-05

Written by Rachel J. Morris
Creative Commons Attribution-NonCommercial-ShareAlike
3.0 Unported License

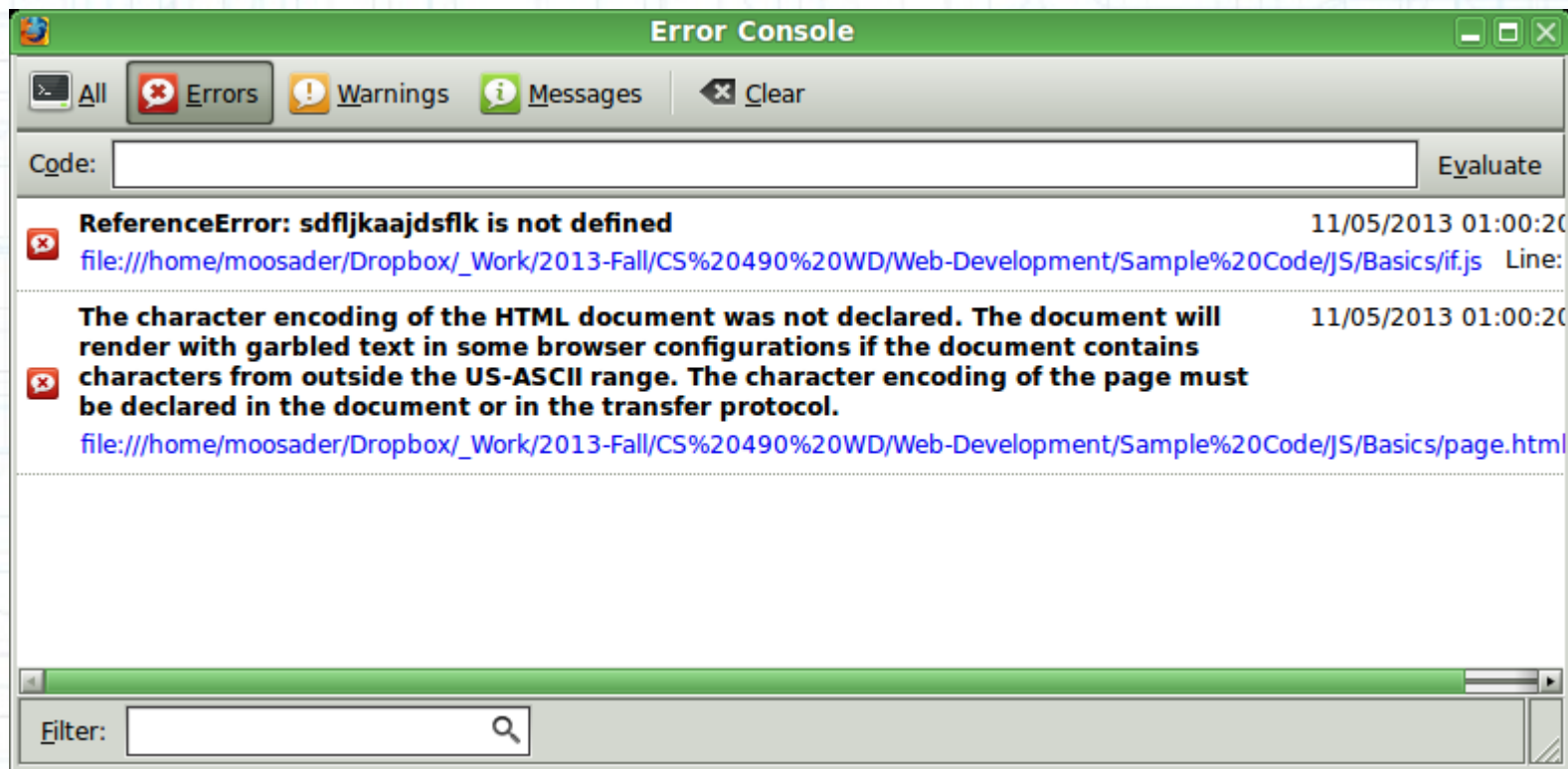
TOOLS!

These are the tools I use to test and debug my JavaScript projects.

- Firefox!
- Web Developer Toolbar
 - <https://addons.mozilla.org/en-US/firefox/addon/web-developer/>
 - <https://addons.mozilla.org/en-US/firefox/addon/toggle-web-developer-toolbar/>

TOOLS!

The Web Developer Toolbar will track errors in your JS code, making it much easier to debug!



Breakdown

- What is JS?
- What is it good for?
- How do you add JS to your page?
- Syntax
 - Variables & Scope
 - Arrays
 - Control Flow
 - Functions & Anonymous Functions
 - JSON Objects

Breakdown

- Using JavaScript
 - Alerts
 - Selecting elements & modifying them
 - Timeout and increment timers
 - Mouse and keyboard detection
 - Simple form validation

What is JS?

- JavaScript is a client-side scripting language, while PHP is a server-side scripting language.
- JavaScript code runs on the user's machine, rather than the server.
- This can make websites appear more interactive, by cleanly transitioning elements, adding animations, and adding nice features.

What is it good for?

- JS is good for simple things like Form Validation, where you may not want to wait to refresh the page in order to check if a field is correct.
- It can also be used to create beautiful scroller banners, or modal windows, or have your images pop up when you click a thumbnail.

What is it good for?

- It can be used with AJAX to call server-side functionality from the client-side.
- It can be used with HTML5 to make items similar to what you might have used Silverlight, Flash, Java Applets, or (gasp) Shockwave for in the past!

What is it good for?

- Some companies are moving away from Flash (since it's proprietary – owned and controlled by one company) and towards HTML5.
 - YouTube has an HTML5 player
 - Netflix has a beta HTML5 player, and is pushing for DRM functionality added to the HTML5 standard.

What is it good for?

- Web-based applications are much more easily cross-platform than traditional software applications. No download required, no compiling for other OSes.
- Even modern smartphones can run JavaScript, even though not all can run Flash.

How do you add JS to your page?

- To add JS scripts to your page, you can include them from an external file:

```
<script src="scripts/config.js"></script>
```

- Or embed them in your markup:

```
<script>  
...  
</script>
```

- Though of course, storing them in separate .js files is preferred.

Syntax

- Variables & Scope
- Arrays & JSON Objects
- Control Flow
- Functions & Anonymous Functions

Variables & Scope

- You can declare a variable with **var**. This will create a variable that is *local in scope*.

```
var age = 100;
```

- If you begin using a variable without first declaring it with **var**, it will search parent scopes for this variable. *If no variable is found, a global variable by the name is created.*

```
age = 100;
```


Arrays & JSON Objects

- You can create an array in JavaScript, by specifying:

```
var ages = new Array();
```

- Arrays are standard 0, 1, 2, ..., n arrays. You can only store integers as the keys, and they are in order.

```
ages[0] = 20;
```

```
ages[1] = 50;
```


Arrays & JSON Objects

- To set an array's values as its declared:

```
var ages = new Array( 25, 50, 75, 100 );
```

- JS arrays can be resized at any time.
- You can also use the **push** function to add a value to the end of an array.

```
ages.push( 125 );
```

Arrays & JSON Objects

- You can also store JSON objects in JavaScript.
- These are essentially how you can have key-value pairs (associative arrays) that aren't sequential, and can have non-integers for keys.
- You can also use JSON objects to imitate a class.

Arrays & JSON Objects

```
1  image_handler = {
2      Setup: function() {
3          this.images = {};
4          // Setup
5          this.images.pathbase = CONFIG.imagePath;
6      },
7
8      Reset: function() {
9          // Reset
10         while ( this.images.length > 0 ) { this.images.pop(); }
11     },
12
13     LoadImage: function( filename, fileextention ) {
14         this.images[filename] = new Image();
15         this.images[filename].src = this.images.pathbase + filename + "." + fileextention;
16     },
17
18     GetImage: function( key ) {
19         if ( this.images[key] == null )
20         {
21             debug.Err( "Unable to find image '" + key + "'" );
22         }
23         else
24         {
25             return this.images[key];
26         }
27     }
28 }
29
```

A JSON object, `image_handler`, which has multiple functions (Setup, Reset, LoadImage, GetImage). It could also store variables as key/value pairs.

Arrays & JSON Objects

```
29
30 ages = {
31     "bob" : 25,
32     "jim" : 50,
33     "hana" : 75
34 };
35
```

Note that, for most variables in JavaScript we assign them with the equal sign =

Within a JSON object, you specify a key-value pair with a COLON in-between!

Arrays & JSON Objects

```
30  helloWorld = {  
31      name : "world",  
32  
33      Setup: function( newName ) {  
34          name = newName;  
35      },  
36  
37      Greet: function() {  
38          alert( "Hello, " + name + "!" );  
39      }  
40  };  
41
```

Here, similar to a class, we can declare a variable (name) but also functions (Setup, Greet).

Note that it's the same key-value pair: The function name, and THEN what it is. In this case, a function.

Also note that there should be commas at the end of each element, within the JSON object.

Arrays & JSON Objects

```
var pets = [  
  { // 0  
    name : "Tesla",  
    type : "Cat",  
    age : 5  
  },  
  { // 1  
    name : "Magellan",  
    type : "Cat",  
    age : 5  
  }  
];  
  
alert( pets[0].name );
```

You can make an array of JSON objects by enclosing them with square brackets [].

Control Flow

- If statements, while loops, for loops are similar to with C++ or PHP.
- There is not a foreach loop, unless you use the jQuery library (more on that later).

Control Flow

For Loop

```
1  var numbers = new Array( 1, 1, 2, 3, 5, 8, 13, 21 );
2
3  var sum = 0;
4
5  // normal for loop
6  for ( var i = 0; i < numbers.length; i++ )
7  {
8      sum += numbers[i];
9  }
10
11 alert( sum );
```

While Loop

```
13  sum = 0;
14
15  // while loop
16  var counter = 0;
17  while ( counter < numbers.length )
18  {
19      sum += numbers[ counter ];
20      counter++;
21  }
22
23  alert( sum );
24
```


Control Flow

```
var age = prompt( "What is your age?" );  
  
if ( age < 21 )  
{  
    alert( "Sorry, you can't come in!" );  
}  
else if ( age >= 18 )  
{  
    alert( "You can come in, but you can't drink!" );  
}  
else  
{  
    alert( "Welcome!" );  
}
```

If Statement

Functions & Anonymous Functions

- Functions in JavaScript are declared with the **function** keyword. You do not need to specify a return-type.

```
1  // normal function
2
3  function Sum( num1, num2, num3 )
4  {
5      return num1 + num2 + num3;
6  }
7
8  alert( Sum( 1, 3, 5 ) );
9
```

Functions & Anonymous Functions

- An Anonymous Function is a function with no name.
- In other words, you never declare the function, you simply create it as soon as it is needed.

Functions & Anonymous Functions

- Notice that we're assigning a value to a variable, but the value we're assigning to it is the return value of the function.
- The function has no name, it only exists during this assignment statement. It is anonymous.

```
11 // Anonymous function
12
13 var average = function( num1, num2, num3 ) {
14     return ( num1 + num2 + num3 ) / 3;
15 };
16
```

Functions & Anonymous Functions

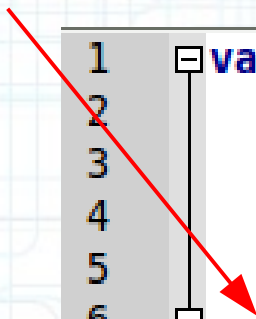
- When you're emulating a class via a JSON object, you're using Anonymous Functions here. (Key-value pairs... the VALUE is an anonymous function).

```
1  var pet = {  
2      type : "dog",  
3      breed : "corgi",  
4      name : "Freddie",  
5  
6      DisplayInfo: function() {  
7          alert( this.name + " is a " + this.type );  
8          alert( "Its breed is " + this.breed );  
9      }  
10 };  
11
```

Functions & Anonymous Functions

- And, by referencing the *variable* DisplayInfo, we call that function.

```
pet.DisplayInfo();
```

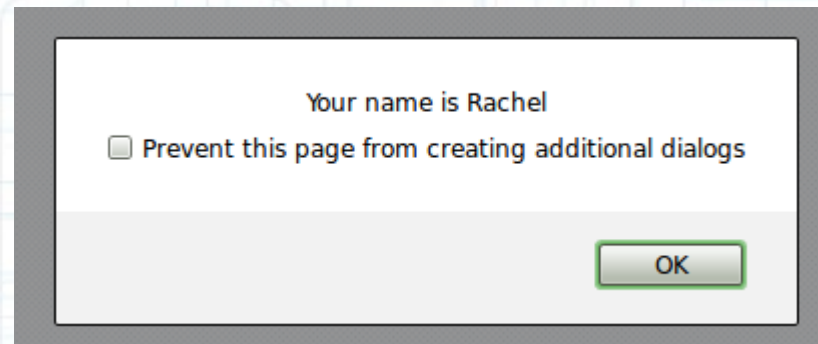
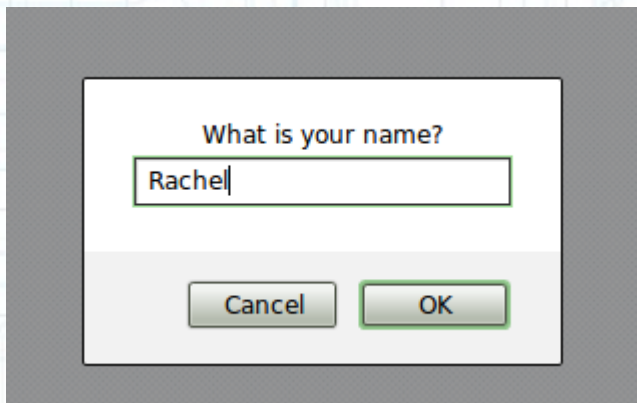


```
1  var pet = {  
2    type : "dog",  
3    breed : "corgi",  
4    name : "Freddie",  
5  
6    DisplayInfo: function() {  
7      alert( this.name + " is a " + this.type );  
8      alert( "Its breed is " + this.breed );  
9    }  
10 };  
11
```


Handy JS Functionality

- While really ugly to use on your webpage, you can ask for user input and display popup boxes with **prompt** and **alert**.

```
1 var name = prompt( "What is your name?" );  
2  
3 alert( "Your name is " + name );  
4
```



Handy JS Functionality

- You can select XHTML elements by ID or Class with JavaScript
- We will cover easier ways to do this with jQuery, too.

Handy JS Functionality

```
page.html x select-elements.js x
1 window.onload = function() {
2
3     var inputBox = document.getElementById( "input-name" );
4
5     alert( inputBox );
6
7     inputBox.value = "Testing";
8
9 };
10
```

```
page.html x select-elements.js x
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>JavaScript</title>
5
6     <script src="select-elements.js"></script>
7   </head>
8
9   <body>
10
11     <input type="text" id="input-name" value="" />
12
13   </body>
14 </html>
15
```



Handy JS Functionality

```
page.html x select-elements.js x
1 window.onload = function() {
2
3     var inputBox = document.getElementById( "input-name" );
4
5     alert( inputBox );
6
7     inputBox.value = "Testing";
8
9 };
10
```

- We use document.getElementById to select our element.
- We can store the 'reference' to this element in the variable, inputBox.
- Then, we can change the value while the webpage is running!

```
<input type="text" id="input-name" value="" />
```

Handy JS Functionality

```
page.html x select-elements.js x
1 window.onload = function() {
2
3     var inputBox = document.getElementById( "input-name" );
4
5     alert( inputBox );
6
7     inputBox.value = "Testing";
8
9 };
10
```

- Notice that the above JavaScript is enclosed with **window.onload**.
- If the window.onload function weren't there, we would get a **null** value when selecting the input-name element.
- This is because the JS file is included before the XHTML has rendered. But, we can make our JS run once the page is loaded, similar to above.

Handy JS Functionality

- You can change an element's value with **.value =**
- You can change an element's style with **.style =**
- This is all a lot easier and prettier with jQuery, so we're not going to go *too* in-depth with selecting and altering elements with pure JavaScript.

Handy JS Functionality

```
page.html ✕ simple-form.js ✕
1  window.onload = function() {
2
3      var inputBox = document.getElementById( "username" );
4
5      // Add error checking when the box is changed
6      inputBox.onchange = function() {
7          if ( inputBox.value == "" ) {
8              inputBox.style.background = "#FFAAAA";
9          }
10     };
11
12 };
13
```

Username:

Email:

Age:

- When the document loads, we can get the input box for “username”.
- Then, we can add a **callback function**, for when that element is **changed**. (onchange)
- Within this callback, anonymous function, we check to see if the field is blank.
- If the field is blank, we set its style to have a background of red.

Handy JS Functionality

- We can also add timers and timeout functions to our JavaScript...

```
2 // Ongoing Timer
3 setInterval( function() {
4     var box = document.getElementById( "counter" );
5     box.value = parseInt( box.value ) + 1;
6 }, 1000 );
```

```
9 // One-time countdown
10
11 setTimeout( function() {
12     alert( "It's been two seconds!" );
13 }, 2000 );
14
```

18

Username:

Email:

Age:

Our second timer

References

-