Functions, The DOM

Today’s Outline

- Recap last week
- Work through an Exercise
- Functions
- The DOM
- Modifying the DOM with Functions

Recap - Basic Types

Strings - any group of characters enclosed in quotations

"Hello"

Number - positive, negative, decimals, exponential

34   3.4   -3

Boolean - value representing TRUE or FALSE

"Hello".length > 3

Recap - Variables

Variables are used to store data for later use.

```
var country = "USA";
```

When creating a variable, you always need the keyword var.

Give your variables meaningful names.

Recap - Conditionals

If-statements are used to run different code depending on circumstances.

```
var moneyLeft = 1000;

if ( moneyLeft > 500 ) {

    console.log(‘You have enough money to gamble’);

}
```

Recap - Conditionals

Combine If-statements with the keyword Else

```
var moneyLeft = 200;

if ( moneyLeft > 500 ) {
	console.log( ‘You have enough money to gamble’ );
} else {
	console.log( ‘You do not have enough’ );
}
```

Workshop example

Pig Latin

Questions ?

Questions regarding any material from last week.

Functions

“A function is a block of code designed to perform a particular task.“

Function Declaration

```
function add (firstNumber, secondNumber) {
    return firstNumber + secondNumber;
}
```

Functions - Why?

- Code Reuse
- Communicate Intent
- Refactoring
- Abstraction - managing complexity

Functions - Syntax

- Keyword - “function”
- Name of the function
- Parameters - separated by parentheses (optional)
- Code block - surrounded by curly braces
- Return statement (optional)

Function - Examples

```
function subtract ( firstParameter, secondParameter ) {
	return firstParameter - secondParameter;
}


function makeNoise () {
	console.log(‘Making some noise.’);
}
```

Function - Invocation

Functions are run, or invoked, by using the () operator.

You can run a function with or without parameters.

```
subtract(5, 1);        // The result is 4

makeNoise();           // Writes ‘Making some noise.’
```

Functions - Return

The return keyword in a function is used to signal the function that it should return a value to the caller and stop executing.

```
function divide ( numerator, denominator ) {
        return numerator / denominator;
}
```

It is not necessary to have a return statement. The function will return undefined if no return statement is supplied.

Functions - Early Return

As soon as the return statement is called, the function will exit and no more code will be run.

```
function concatenate ( firstString, secondString ) {
    if ( firstString === undefined ) {
        window.alert(“You the first parameter”);
        return;
    }

    return firstString + secondString;
}
```

Functions - FizzBuzz

Now that we know how to use functions...

Functions and the Browser

HTML Events

HTML events are used to respond to user input. When a user clicks a button, enters data into a form or performs any action on the page, the HTML events will let javascript know what is happening.

Using HTML events, we can write code to effectively respond to the user's behavior.

Functions and the Browser

You can run javascript functions using basic HTML controls by attaching functions to html events.

```
function tellUser() {
        window.alert(“You pushed me!”);
}

<button onclick=”tellUser()”>Push Me</button>
```

Functions and the Browser

HTML event examples

- onclick - when clicking an HTML element with a mouse
- onchange - when a value in an input changes
- onkeydown, onkeyup - responds to user input in a text box
- onblur - when a user leaves a form element
- onfocus - when a user enters a form element

Functions - Browser

Events Demo

Functions - Selection

In addition to responding to events from the browser, you can select HTML elements using your javascript code.

Your browser has built-in functions for selecting items on screen and dealing with HTML.

```
<div id=”myFirstDiv”></div>

var myFirstDiv = document.getElementById(“myFirstDiv”);

myFirstDiv.innerHTML = “Hello world”;
```

Functions - Selection

Demo

Document - DOM

The DOM (Document Object model) in javascript is an abstraction around the contents of your browser window.

We use the object document to interact with the DOM.

Listing out all of the functions and properties the document object has would be exhaustive, see the citations for a list.

Javascript frameworks have been popularized around the idea of simplifying DOM interaction.

Document - DOM

At this point you might be feeling overwhelmed. The DOM is a strange concept and it’s the first time you might be hearing the words “abstraction”.

You will not learn everything about the manipulating the DOM until you need to perform a specific function (i.e. responding to user interaction).

The key is knowing what to search for and how to read documentation.

- “How do I know when a user clicked a button?”
- “I need to run a function when a certain key is pressed.”
- “When the mouse is hovered over an area, THIS needs to happen.”

Functions - Variables

Variables defined inside a function can only be accessed inside of the function.

This is called “function scope”;

```
function printMyName ( firstName, lastName ) {
        var myFullName = firstName + “ “ + lastName;
        console.log(myFullName);
}

console.log(myFullName);        // This is undefined
```

Function - Variables

Demo

Functions - Nesting

You can define functions within other functions.

```
function myFirstFunction () {
    console.log( “Nesting functions” );

    function mySecondFunction () {
        console.log( “My second function” );
    }
    mySecondFunction();
}
```

mySecondFunction is inaccessible outside of myFirstFunction.

Functions - Nesting

Nested functions have access to any variables from the parent function.

This is called a Closure, or Lexical Scope.

```
function printMyFullName (firstName, lastName) {
        var fullName = firstName + “ “ + lastName;
        function sayName() {
                console.log(fullName);
        }
}
```

Functions - Nesting Demo

Functions - Assignment

Since javascript treats functions as first-class citizens, functions can be assigned to variables.

```
var add = function ( firstNumber, second Number ) {
    return firstNumber + secondNumber;
};

add(3, 4);      // returns 7
```

This is called a function expression.

Functions - Assignment

Differences

- Function expressions need a semi-colon, it is an expression after all
- Function expressions cannot be used before they are defined, this is safer for programmers.
- Function expressions are preferred (because of hoisting and potential syntax errors).

Functions - Assignment

Demo - now that we know function expressions are preferred.

Workshop

- Form validation (together)
- Make our own calculator

Citations

http://www.w3schools.com/js/js_functions.asp

http://eloquentjavascript.net/03_functions.html

https://developer.mozilla.org/en-US/docs/Web/API/document

http://en.wikipedia.org/wiki/Abstraction_%28computer_science%29

http://www.w3schools.com/tags/ref_eventattributes.asp - Full list of html events