Web Advanced: Javascript APIs

"We will learn JavaScript properly. Then, we will learn useful design patterns. Then we will pick up useful tools for making cool things better."

FALL 2018

SESSION #3

SYNTAX: FUNCTIONS

jaink@newschool.edu

https://canvas.newschool.edu/courses/1407281

https://classroom.github.com/classrooms/42809645-parsons-web-advanced-javascript-fall-2018

RECAP

FUNCTIONS

Functions encapsulate a block of code that does a specific task to make it reusable.

- ➔ First Class Object
- ➔ Initialization
- ➔ Scopes
- ➔ Functions as Values
- ➔ Closures

BUILT-IN FUNCTIONS

```
// typical built-in functions
my_string.charAt(1); //returns a string
parseInt(12.34); //returns integer
Math.random(); //returns a floating number
[1,2,3,4].map(); //returns array
isNaN(varname); //returns true or false
```

NEW FUNCTIONS

```
// a new basic function - pretty useless
function randomNumber() {
    console.log('I am returning', Math.random());
}
// a new basic function - better
function convertToCelsius(deg_fah) {
    let converted_deg = (deg_fah-32) * 5/9;
    console.log('The converted temperature is', converted_deg);
}
```

CALLING FUNCTIONS

```
// this returns the actual reference, not
the function evaluation

randomNumber;


// this executes the function

randomNumber();
```

FUNCTION PARAMETERS / ARGUMENTS

- ➔ **Parameters: variables needed by the function itself to run. These are set and then destroyed once complete**
- ➔ **Arguments: the vars or values sent to the function when called.**

```
function convertToCelsius(deg_fah) {
    let converted_deg = (deg_fah-32) * 5/9;
    return(converted_deg);
}
console.log( convertToCelsius(32) );
```

FUNCTION PARAMETERS / ARGUMENTS

➔ **Default params can be set by checking if undefined:**

Method 1:

```
function convertToCelsius(deg_fah) {
    if (deg_fah === undefined)  deg_fah = 32;
    let converted_deg = (deg_fah-32) * 5/9;
    return(converted_deg);
}
console.log( convertToCelsius() );
```

Method 2:

```
function convertToCelsius(deg_fah) {
    deg_fah = deg_fah || 32;
    let converted_deg = (deg_fah-32) * 5/9;
    return(converted_deg);
}
```

FUNCTION PARAMETERS / ARGUMENTS

➔ **Default params can be set by checking if undefined:**

Method 3 (ES6):

```
function convertToCelsius(deg_fah = 32) {
    let converted_deg = (deg_fah-32) * 5/9;
    return(converted_deg);
}
console.log( convertToCelsius() );
console.log( convertToCelsius(32) );
```

SCOPE

Scope is how a variable is available to the entire program:

➔ Global scope:

Any variables or functions declared outside of a function will be available to all JavaScript code on the page, whether that code is inside a function or otherwise

➔ Functional/Local scope:

Variables and functions declared inside a function are visible only inside that function—no code outside the function can access them.

Local variables also have a lifetime - they die when the function finishes executing.

FUNCTION: A 1st CLASS OBJECT

- ➔ One of the more powerful and flexible features of Javascript.
- ➔ functions can be treated just like any other type of value

```
// assign function to variable
const new_function =  convertToCelsius;
console.log(new_function(100));

// pass function as a parameter
function convert(converter, temperature) {
    console.log(converter(temperature));
}
convert(convertToCelsius, 23);
```

FUNCTION: ALT. DEFINITIONS

Function Expression:

```
// assign anonymous function to variable
const convertToCelsius = function(deg_fah)
{
    let converted_deg = (deg_fah-32) * 5/9;
    return converted_deg;
};

// assign named function to variable
const newFunction =  function
convertToCelsius(deg_fah) {
    let converted_deg = (deg_fah-32) * 5/9;
    return converted_deg;
};
```

FUNCTION: ALT. DEFINITIONS

Function Constructors:

Not used much as it's not very secure...similar to new String('string_value'):

```
// assign anonymous function to variable
const convertToCelsius =  new
Function(deg_fah, 'let converted_deg =
(deg_fah-32) * 5/9; return
converted_deg;');
```

FUNCTION: ALT. DEFINITIONS

Arrow Notation (ES6):

- ➔ **Very compact**
- ➔ **Works great for one line functions**

```
// assign anonymous function to variable
const convertToCelsius = deg_fah =>
(deg_fah-32) * 5/9;

const convertToCelsius = (deg_fah,dec) =>
(deg_fah-32) * 5/9;

const convertToCelsius = deg_fah => {
  return (deg_fah-32) * 5/9;
}
```

FUNCTION CALLBACKS

Function that are sent as arguments to another function:

```
// pass function as a parameter
const convertToCelsius =  function(deg_fah)
{
    let converted_deg = (deg_fah-32) * 5/9;
    return converted_deg;
};
// pass function as a parameter
function convert(converter, temperature) {
    console.log(converter(temperature));
}
convert(convertToCelsius, 23);
```

FUNCTION CALLBACKS

```
// pass action function as a parameter
const convertToCelsius =  function(deg_fah,
callback) {
    let converted_deg = (deg_fah-32) * 5/9;
    callback(converted_deg);
};
// pass function as a parameter
function convert(temperature) {
    console.log(temperature);
}
convertToCelsius(23, convert);
```

REVISITING ARRAY METHODS

```
var my_array =  ["blue", "red", "green"];
my_array.map(function (val) {
    return (val).toUpperCase();
});
```

Similar to:

```
var toUppercase = function (value) {
    return (value).toUpperCase();
}
my_array.map( toUppercase );
```

```
var array_sum = [1,2,3,4,5].reduce(
function(prev,current){
  return prev + current;
});
console.log(array_sum);
```

CLOSURES

- ➔ **Function that contain functions get access to the outer functions variables and parameters, even after the call is made to the outer function.**
- ➔ **Used for data privacy as the enclosed variables are only in scope within the containing (outer) function.**

```
const module = function() {
    let localVar = 1913;
    let localFunc = function() {
        return localVar;
    }
    let otherLocalFunc = function(num) {
        localVar = num;
    }
    return {
        getVar: localFunc,
        setVar: otherLocalFunc
    }
};

let newmod = module();
console.log(newmod.getVar()); // 1913
newmod.setVar(1776);
console.log(newmod.getVar()); //1776
```

EXAMPLES

```
/* build out an example of using function..: */

let converted_deg;

for (let i=-148; i<=212;i=i+10) {

    converted_deg = (i-32) * 5/9;

    console.log('The converted temperature of',i,' is',
converted_deg);

}


// a new basic function - better

function convertToCelsius(deg_fah) {

  let converted_deg = (deg_fah-32) * 5/9;

  return(converted_deg);

}


let converted_deg;

for (let i=-148; i<=212;i=i+10) {

    converted_deg = convertToCelsius(i);

    console.log('The converted temperature of',i,' is',
converted_deg);

}
```

EXAMPLES

```
// multiple parameters and arguments

let converted_deg;

for (let i=-148; i<=212;i=i+10) {

    converted_deg = convertToCelsius(i,2);

    console.log('The converted temperature of',i,' is',
converted_deg);

}


// retype the prev 2 param version to set default for fixed:

function convertToCelsius(deg_fah, dec_fixed = 4) {

  let converted_deg = (deg_fah-32) * 5/9;

  let dec_fixed = dec_fixed || 4;

  converted_deg = converted_deg.toFixed(dec_fixed)

  return(converted_deg);

}
```

EXAMPLES

```
// function expressions
console.log(convertToCelsius(12)); // will error
const convertToCelsius = function templateFunction(deg_fah) {
  let converted_deg = (deg_fah-32) * 5/9;
  return converted_deg;
};
console.log(convertToCelsius(2)); // will work

//callbacks
// pass function as a parameter
const convertToCelsius =  function(deg_fah, callback) {
  let converted_deg = (deg_fah-32) * 5/9;
  callback(converted_deg);
};
const displayInConsole = function(value) {
  console.log('The converted temperature is: ', value);
}
const displayInModal = function(value) {
  alert('The converted temperature is: ' + value);
}
```

EXAMPLES

```
//array iterators example:

let my_quote = "The Answer to the Great Question Of Life, the
Universe and Everything is Forty-two";

let words_array = my_quote.split(" ");

console.log(words_array);

let total = words_array.reduce(function(prev, curr) {

  return prev + curr.length;

}, 0);

let average = total/words_array.length;

console.log(average);
```

EXAMPLES

```
 // Closure example
function multiplier(factor) {
  let ret = function(number) {
    return number * factor;
  };
  return ret;
}
let twice = multiplier(2);
let thrice = multiplier(3);

// now square is a custom function that returns number * 2
console.log(twice(3));

// now cubed is a custom function that returns number * 3
console.log(thrice(3));
```

EXAMPLES

exercise:

```
//a function that searches a string and find if it contains a
pattern eg. life case insensitive.


var my_quote = "The Answer to the Great Question Of Life, the
Universe and Everything is Forty-two";


function findVal (string, pattr) {
  if (string.toLowerCase().indexOf(pattr) !== -1) {
    return true;
  } else {
    return false;
  }
}


console.log(findVal(my_quote, "life"));
```

Assignment:

Next Steps

- ➔ More Javascript Syntax:
  - ◆ datatype: Objects