



# Module 4 Day 2

## JavaScript Functions

# Array Functions

- `concat()`
  - Let `newArray = array1.concat(array1)`
- `join()`
  - Joins all elements into a string
- `slice()`
  - Returns a portion of an array (akin to substring)
- `indexOf()`, `lastIndexOf`
- [https://www.w3schools.com/jsref/jsref\\_obj\\_array.asp](https://www.w3schools.com/jsref/jsref_obj_array.asp)
- [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Array#](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array#)

# Named Functions

- Denoted by the function keyword plus name
- Parameters, but no types
- No return type declared
- Functions not overloaded, but overwritten

```
/**
 * Create JSDoc documentation so that VSCode can understand and give you IntelliSense.
 *
 * @param {any} parameter1 The first parameters to append
 * @param {string} parameter2 The second parameter to append
 * @returns {string} The two parameters concatenated.
 */
function someFunction(parameter1, parameter2) {
  let s = '' + parameter1 + parameter2;
  return s;
}
```

# Parameters

- Always optional, but default to **undefined**
- Default values can be assigned
  - Can be expressions, call functions, new objects, create arrays
  - Can refer to parameters to the left in the list
- `function greet(name, greeting, message = greeting + ' ' + name) {`
- Passing **undefined** results in the default value
- [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Default\\_parameters](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Default_parameters)
- **arguments** can be used to get all the arguments passed in
- Arguments and parameters are *not quite* the same thing ([w3schools](https://www.w3schools.com/js/default.asp))

Function **parameters** are the **names** listed in the function definition.

Function **arguments** are the real **values** passed to (and received by) the function.

# Anonymous Functions

- Functions (blocks of code) can be passed around like other values
  - You can pass strings, numbers, arrays, objects and functions as parameters
  - (this is true in C# also)
- Are all values named?
  - We don't always create a variable before passing it into a function
  - We don't have to create a function before passing it in either
- Syntax:

```
function (s) {return s.toUpperCase();}
```

  - OR, the "fat arrow" / lambda syntax:

```
(s) => {return s.toUpperCase();}
```
- NOTE: any amount of code can be inside the { }

# Array functions requiring function parameter

| JS Function | Parameters   | Returns  | C# Linq        |
|-------------|--------------|--|----------------|
| Filter      | Item         | Array of the same type ( $\leq$ original size), filtered by the function   | Where          |
| Map         | Item         | Array of same size, original elements "mapped" to something new  | Select         |
| Sort        | Item1, Item2 | Array of same size and type, with elements sorted. Function returns 1 if Item1 > Item2, -1 if item1 < item2, 0 otherwise | OrderBy        |
| Reduce      | Accum, Item  | A single value, allows calculating a running value   | Sum, Aggregate |
| Every       | Item         | Boolean, true if every item meets the condition  | All            |
| Some        | Item         | Boolean, true if at least one item meets the condition   | Any            |



# JSDoc

- VS Code uses this documentation for IntelliSense
- `@param {type} param-name Parameter-description`
- `@param {type} [param-name=default-value] Description...`
- `@returns {type} Return-value-description`
- <https://jsdoc.app/>, <https://jsdoc.app/index.html#block-tags>