JAVASCRIPT FUNCTIONS

A JavaScript function is a block of code designed to perform a particular task.
A JavaScript function is executed when "something" invokes it (calls it).

```
function myFunction(p1, p2) {
  return p1 * p2;   // The function returns the product of p1 and p2
}

// CALLING A FUNCITION
var x = myFunction(4, 3);
```

A JavaScript function is defined with the **function** keyword, followed by a **name**, followed by parentheses **()**.
Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).
The parentheses may include parameter names separated by commas:
(*parameter1*, *parameter2*, ...)
The code to be executed, by the function, is placed inside curly brackets: **{}**

```
function name(parameter1, parameter2, parameter3) {
  // code to be executed
 var carName = "Volvo"; // can have local variables
}
```

Function **parameters** are listed inside the parentheses () in the function definition.
Function **arguments** are the **values** received by the function when it is invoked.
Inside the function, the arguments (the parameters) behave as local variables.

Create an Array

```
var fruits = ['Apple', 'Banana'];
console.log(fruits.length); // 2
```

Access (index into) an Array item

```
var first = fruits[0];
// Apple var last = fruits[fruits.length - 1];
// Banana
```

Loop over an Array

```
fruits.forEach(function(item, index, array) { console.log(item, index); }); // Apple 0 // Banana 1
```

Add to the end of an Array

```
var newLength = fruits.push('Orange'); // ["Apple", "Banana", "Orange"]
```

Remove from the end of an Array

```
var last = fruits.pop(); // remove Orange (from the end) // ["Apple", "Banana"];
```

Remove from the front of an Array

```
var first = fruits.shift(); // remove Apple from the front // ["Banana"];
```

Add to the front of an Array

```
var newLength = fruits.unshift('Strawberry') // add to the front // ["Strawberry", "Banana"];
```

Find the index of an item in the Array

```
fruits.push('Mango'); // ["Strawberry", "Banana", "Mango"]
var pos = fruits.indexOf('Banana'); // 1
```

Remove an item by index position

```
var removedItem = fruits.splice(pos, 1); // this is how to remove an item // ["Strawberry", "Mango"]
```

TESTING PLAYGROUND:

- https://www.w3schools.com/js/tryit.asp?filename=tryjs_function_return
- https://www.w3schools.com/js/tryit.asp?filename=tryjs_farenheit_to_celsius

JAVASCRIPT ARRAYS

An array is a special variable, which can hold more than one value at a time.
If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

- Normal variables:

```
var car1 = "Saab";
var car2 = "Volvo";
var car3 = "BMW";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300? **The solution is an array!**
An array can hold many values under a single name, and you can access the values by referring to an index number.

- Examples of array:

```
var cars = ["Saab", "Volvo", "BMW", "unlimited…."];


var arr = new Array(1, 2, 3, 4);
console.log(arr.length); // 4
arr[20] = 2;
console.log(arr.length); // 21 - even though there are no elements between index 5 and 19
```

ASSOCIATIVE ARRAYS

Many programming languages support arrays with named indexes.
Arrays with named indexes are called associative arrays (or hashes).
JavaScript does **not** support arrays with named indexes.
In JavaScript, **arrays** always use **numbered indexes**.

In JavaScript, **arrays** use **numbered indexes**.
In JavaScript, **objects** use **named indexes**.

Example:

```
var person = [];
person[0] = "John";
person[1] = "Doe";
person[2] = 46;
var x = person.length;     // person.length will return 3
var y = person[0];         // person[0] will return "John"
```