Cheat Sheet – Types & Scope

Primitive vs Reference Types

There are two different types of Types. Primitives and Reference Types.

Primitives are simple types like String, Number or Boolean.
Reference Types are more complex types like Array or Object.

The most important difference can be seen when copying variables with such types.
If you copy a primitive (e.g. a variable holding a number), then the new variable will be an actual copy. This means, if you change the first variable, the second one (holding the copy) will not be changed.

```
var number1 = 10;
var number2 = number1;
number1 = 15;
console.log(number1); // prints 15
console.log(number2); // prints 10
```

That changes when looking at reference types. Here, the variable doesn't actually store the data but it only stores a pointer to a place in memory, where the data is stored.
Therefore, if you copy such a variable, you copy the pointer.
If you then change the value of the first variable, the value of the second one will also be changed since you changed the data in the memory. The pointer is still the same.

```
var array1 = [1, 2];
var array2 = array1;
array1.push(3);
console.log(array1); // prints [1, 2, 3]
console.log(array2); // prints [1, 2, 3]
```

Global Scope

Scope is another important concept in JavaScript. By default, all your variables, functions and objects "belong" to the global scope. When executing code in the browser, that scope is the Window object. The

Window refers to the browser window and holds the actual document. But you'll learn more about that in the DOM section.
Since everything belongs to the global scope, you may not have multiple variables with the same name and if you change a variable, it will be changed in all the places in the code.

Function Scope

Function scope is another scope in JavaScript. If you write a function, then it has its own scope – inside the curly braces. Inside the function body you may use variable names you also use in the global scope since they won't interfere as they belong to a different scope.

Learn more about scope here: http://www.w3schools.com/js/js_scope.asp