JAVASCRIPT

Created by Alexandre Rivet

BASICS

VARIABLES

A storage container of different kind of objects

```
var a = undefined;
var b = 2;
var $c = "string";
var _d = {};
```

JavaScript is case-sensitive. So var a and var A are two different variables.

*To avoid global scope issue, always prefix with **var** keyword*

TYPES (1/2)

Type	Examples
Number	`var a = 1;` `var b = -3.14;`
string	`var a = "One string";` `var b = 'A second String with "double quotes".';` `var c = "An escape \"double quote\"";`
Boolean	`var trueBool = true;` `var falseBool = false;`

TYPES (2/2)

Type	Examples

Object	`var obj1 = {};`
	`var obj2 = new Object();`
	`// The first notation is more recommended`

null	`var emptyVar = null;`
undefined	`var undefinedVar;`

To find the type of a variable, you can use the operator:

```
console.log(typeof variableName);
```

OPERATORS (1/3)

The = symbol assigns a value to a variable

```
var negNumber = -3.14;
var string = "One String";
var trueBool = true;
var emptyVar = null;
```

OPERATORS (2/3)

The +, -, *, / are the arithmetic operators

```
var a = 2;
var b = 3;
var sum = a + b;
var sub = a - b;
var mul = a * b;
var div = a / b;
```

The arithmetic operators respect the algebra rules

OPERATORS (3/3)

The +=, -=, *=, /=, ++, -- are the shorthand math operators

```
var a = 2;
a += 2; // a = a + 2
a -= 2; // a = a * 2
a *= 2; // a = a * 2
a /= 2; // a = a / 2
a++;    // a = a + 1
a--;    // a = a - 1
/*
  For the two last operators, order is important
  Pre-incrementation or Post-incrementation
  var a = 1;
  var b = a++ * 2; // b = 2
  a = 1;
  b = ++a * 2; // b = 4
*/
```

STRINGS & NUMBERS TOGETHER (1/2)

When there is a string in any combination, JavaScript will interpret as you want to combine and get a string.

```
var a = 29;
var b = "9";
var sum = a + b;
```

sum is equals to "299" and not 38

```
var sum = 38 + true + "string" + 38 + false + 10;
```

sum is equals to "39string38false10"

STRINGS & NUMBERS TOGETHER (2/2)

When another operator than the + is used, it will not work

```
var a = 29;
var b = "9";
var sub = a - b; // NaN (Not a Number)
var mul = a * b; // NaN
var div = a / b; // NaN
```

To test if a variable is a Number or not, you can use:

```
console.log(isNaN(sub)); // false
```

CONDITIONAL STATEMENTS (1/2)

```
// If conditional statement
if (someCondition) {
  // Do something
}
```

If someCondition is true, whatever is inside the statement, the code block will run.
Otherwise, it will go to the next statement in the code.

CONDITIONAL STATEMENTS (2/2)

```
// If Else statement
if (someCondition) {
  // Do something when someCondition is true
} else {
  // Do something else when someCondition is false
}
```

LOGICAL OPERATORS (1/5)

```
var x = 5;
```

Operator	Description	Comparison

==	equal to	`x == 8;   // Return false`
		`x == 5;   // Return true`
		`x == '5'; // Return true`

===	equal value and equal type	`x === 5;  // Return true`
		`x === '5';  // Return false`

LOGICAL OPERATORS (2/5)

```
var x = 5;
```

Operator	Description	Comparison

!=	not equal to	`x != 8;   // Return true`
		`x != 5;   // Return false`
		`x != '5'; // Return false`

!==	not equal value or not equal type	`x !== 5;   // Return false`
		`x !== '5';  // Return true`

LOGICAL OPERATORS (3/5)

```
var x = 5;
```

Operator	Description	Comparison
>	greater than	`x > 8;  // Return false`
<	less than	`x < 8;  // Return true`
>=	greater than or equal to	`x >= 8; // Return false`
<=	less than or equal to	`x <= 8; // Return true`

LOGICAL OPERATORS (4/5)

```
var x = 6;
var y = 3;
```

Operator	Description	Comparison
&&	and	`(x < 10 && y > 1) // Return true`
||	or	`(x == 5 || y == 5) // Return false`
!	not	`!(x == y)      // Return true`

LOGICAL OPERATORS (5/5)

```javascript
// Classical way to do an if/else
if (a == b) {
  console.log("a & b match");
} else {
  console.log("a & b don't match");
}

// Ternary operator
a == b ? console.log("a & b match") : console.log("a & b don't match");
```

LOOPS (1/3)

Loop type	Example
For	`for(var i = 0; i < 10; i++) {` `  console.log(i);` `}`
While	`var i = 0;` `while(i < 10) {` `  console.log(i++);` `}`
Do...While	`var i = 0;` `do {` `  console.log(i++);` `} while (i < 10);`

LOOPS (2/3)

Keyword	Example

Break	`// Terminate the current loop`
	`var output = 0;`
	`for (var i = 0; i < 10; i++) {`
	`  if (i === 5) {`
	`    break;`
	`  }`
	`  output += i;`
	`}`
	`console.log(output); // 10`

LOOPS (3/3)

Keyword	Example

Continue

```
// Terminate the current iteration of the loop
var output = 0;
for (var i = 0; i < 10; i++) {
  if (i === 5) {
    continue;
  }
  output += i;
}
console.log(output); // 40
```

ARRAYS (1/3)

Arrays are used to store multiple values in a single variable.

```
var fruits1 = ["orange", "apple", "lemon", "pear"];
// or
var fruits2 = new Array("orange", "apple", "lemon", "pear");
```

Access to an element in one array

```
var element = fruits1[1]; // element == "apple"
```

Arrays are objects

```
console.log(typeof fruits1); // object
```

ARRAYS (2/3)

```
var fruits = ["orange", "apple", "lemon", "pear"];
```

Properties (Get it by name)

```
var nbElements = fruits.length; // 4
```

Methods

```
// Reverse the array
fruits.reverse(); // ["pear", "lemon", "apple", "orange"]

// Remove the first value of the array
fruits.shift();   // ["lemon", "apple", "orange"]

// Add new elements in front of the array (separated by commas)
fruits.unshift("cherry"); // ["cherry", "lemon", "apple", "orange"]

// Remove the last value of the array
fruits.pop(); // ["cherry", "lemon", "apple"]
```

ARRAYS (3/3)

```
var fruits = ["orange", "apple", "lemon"];
```

Methods

```
// Add new elements at the end of the array (separated by comma)
fruits.add("pear"); // ["orange", "apple", "lemon", "pear"]

// Remove elements from a specific position in array
fruits.splice(1, 2); // ["orange", "pear"]

// Create a copy of an array
var newArr = fruits.slice();

// Get the first element that matches the search pattern
var result = fruits.indexOf("pear"); // 1

// Create a string with all items separated by a separator (comma as default)
var stringArray = result.join(); // "orange,pear"
```