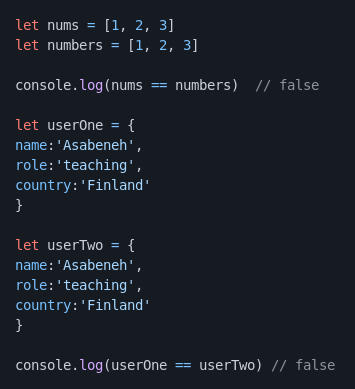
30 Days of JavaScript

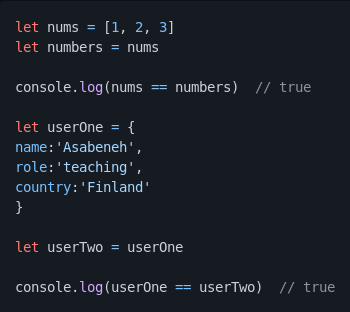
**Day 2 - *Data Types***

*Non-Primitive Data Types:*

Non-primitive data types cannot be compared by value. If two non-primitive data types hae the same prperties and values, they are not strictly equal. Rule of thumb: we do not copare non-primitive data types. Do not compare arrays, functions, or objects.



Non-primitive values are referred to as reference types, because they are being compared by reference instead of value. Two objects are only estryctly equal if they refer to the same underlying object.



*Numbers*

Numbers are integers and decimal values which can do all the arithmetic operations.

We use const for non-changing values, for example gravitational constant or pi constant. Once we declared a variable with const, his value will not change, and if we try to do it, JavaScript will return an error.

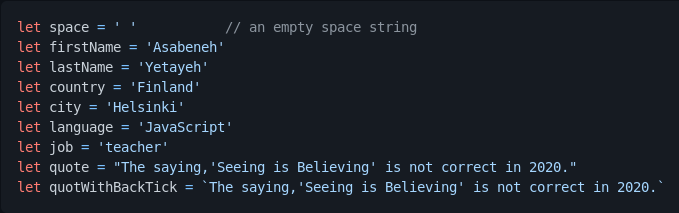
*Math Objects*

In JS the math object provides a lots of methods to work with numbers.

|  |  |  |
| --- | --- | --- |
| Math object | Usage | Example |
| Math.round() | To round numbers (redondear) | Console.log(Math.round(PI));  Output will be 3 |
| Math.floor() | To round down | Console.log(Math.floor(PI));  Output will be 3 |
| Math.ceil() | To round up | Console.log(Math.ceil(PI));  Output will be 4 |
| Math.min() | Returns the minimum value | Console.log(Math.min(-5, 3, 20));  Output will be -5 |
| Math.max() | Returns the maximum value | Console.log(Math.max(-5, 3, 20));  Output will be 20 |

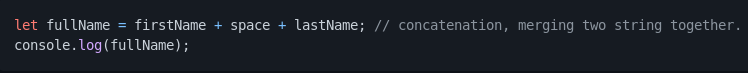
*Strings*

Strings are texts, which are under single, double, back-tick quote. To declare a string, we need a variable name, assignment operator (=), a value under a single quote, double quote, or backtick quote. Let’s see some examples of strings:



*String concatenation*

Connecting two or more strings together is called concatenation. Using the strings declared previusly:



*Escape sequences in strings*

In JS and other programming languages \ following by some characters is an escape sequence. Let’s see the most common escape characters:

* \n: New line
* \t: Tab, means 8 spaces
* \\: Backslash
* \’: Single quote (‘)
* \”: Double quote (“)

*Template literals (Template strings)*

To create a template strings, we use two back-ticks. We can inject data as experessions inside a template string. To inject data, we enclosses the expressions with a curly bracket({}) preceded by a $ sign.

*Strings methods*

Everithing in JavaScript is an object. A string is a primitive data type that means we cannot modify it once it is created. The string object has many string methods. There are different string methods that can help us work with strings.

1.- length: The strign length method returns the number of characters in a string included empty space.

2.- Accesing charactes in a string: We can access each character in a string using its index. In programming, counting starts from 0. The first index of a string is zero, and the last index is the length of the string minus one.

**Secuencias de escape en cadenas**

En JS y otros lenguajes de programación \ seguido de algunos caracteres es una secuencia de escape. Veamos los caracteres de escape más comunes:

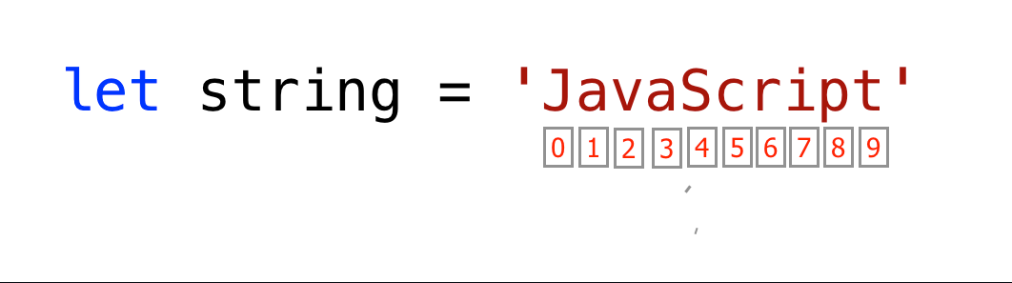
* \n: Nueva línea
* \t: Tabulador, significa 8 espacios
* \\: Barra invertida
* \': Una frase (')
* \”: comillas dobles (“)

**Literales de plantilla (cadenas de plantilla)**

Para crear una plantilla de cadenas, usamos dos tildes de retroceso. Podemos inyectar datos como experiencias dentro de una cadena de plantilla. Para inyectar datos, encerramos las expresiones con un corchete ({}) precedido por un signo $.

**Métodos de cadenas**

Todo en JavaScript es un objeto. Una cadena es un tipo de datos primitivo, lo que significa que no podemos modificarla una vez que se crea. El objeto de cadena tiene muchos métodos de cadena. Existen diferentes métodos de cadenas que pueden ayudarnos a trabajar con cadenas.

1. Longitud: El método de longitud de cadena devuelve el número de caracteres de una cadena incluidos los espacios vacíos.
2. Acceder a los caracteres de una cadena: Podemos acceder a cada carácter de una cadena utilizando su índice. En programación, el conteo comienza desde 0. El primer índice de una cadena es cero y el último índice es la longitud de la cadena menos uno. 
3. to
4. toUpperCase(): this method changes the string to uppercase letters.
5. toLowerCase(): this method changes the string to lowercase letters.
6. substr(): it takes two arguments