FLIPPED CLASS: TYPES, VARIABLES

AVISO: TODAS LAS IMÁGENES ESTÁN OBTENIDAS DE:

Flanagan, D. (2020). JavaScript: The Definitive Guide : Master the World's Most-used Programming Language. O'Reilly Media.

El objetivo de la flipped classroom es que la teoría se vea en casa, y los ejercicios se hagan en el aula. Siguiendo este procedimiento, las lecturas/actividades propuestas son:

Nº	Descripción	Enlace	Tiempo aproximado (min)
1	Introducción a las variables	https://www.codecademy.com/resources/docs/javascript/variables	15
2	Introducción a las operaciones	https://www.codecademy.com/resources/docs/javascript/operators	20
3	Introducción a las strings	https://www.codecademy.com/resources/docs/javascript/strings	20
4	Introducción a los data types	https://www.codecademy.com/resources/docs/javascript/data-types	15
5	Introducción a los bitwise operators	https://www.codecademy.com/resources/docs/javascript/bitwise-operators	15
6	Documento teórico de abajo basado en el libro de javascript		30

1. Formato léxico del lenguaje

Cuando escribimos castellano, por ejemplo, de manera irracional escribimos una serie de código que lo tenemos muy implementado en nuestra cabeza, pero que no tiene por qué ser así. Por ejemplo, en castellano utilizamos una serie de letras que en otros idiomas no (ñ, ll), un alfabeto que otros idiomas no (el chino) y una ortografía que a veces no tiene sentido (lógico, al menos) pero que debe ser así.

En el caso de javascript, hay una serie de normas que tenemos que cumplir a la hora de escribirlas. Haremos un breve repaso, pero esto será el día a día en nuestros programas:

- **JavaScript sí diferencia entre mayúsculas y minúsculas**.
- JavaScript no controla (exceptuando algunas cosas) los saltos de línea ni los espacios, aunque en formato texto sí que los valora.
- JavaScript sí entiende de tabulaciones y algunos caracteres especiales Unicode.
- **Los comentarios en el código** (podemos disertar sobre si son importantes o no, aquí, cada maestrillo con su librillo) tienen dos formatos:
    - Línea simple: //
    - Bloque: /* */
- **Literals**: valores que aparecen "a pincho" y se reconocen como tales:
    - 12 (enteros)
    - 1.2 (floats)
    - "hello world"
    - 'Hi'
    - true
    - false
    - null
- **Palabras reservadas**, en casi todos los códigos hay palabras que no se pueden usar como variables o nombres de función, porque son bases del programa:

```
as      const      export     get         null     target   void
async   continue   extends    if          of       this     while
await   debugger   false      import      return   throw    with
break   default    finally    in          set      true     yield
case    delete     for        instanceof  static   try
catch   do         from       let         super    typeof
class   else       function   new         switch   var
```

```
enum  implements  interface  package  private  protected  public
```

- **Identificadores**: las palabras para definir variables, pueden empezar por letras, _, $
- **Unicode**: infórmate como estudiante de qué trata esto (busca en goolge, chatgpt...)
- **Punto y coma**: por norma general, java script no requiere de punto y coma, aunque a veces puede venir bien para separar dos sentencias según lo que quiera hacer el programador. Aquí, de nuevo, permitiremos que se haga como cada uno quiera

  let y = x + f

  (a+b).toString()

  Frente a

let y = x + f(a+b).toString();

2. Variables y tipos

Hay dos tipos de variables principales en javascript, los primitivos (e inmutables) y los objetos (conjuntos de primitivos).

NOTA:

Algo importante de las variables es su definición, las constantes se definen con *const* y las variables con *let/var*.

1. Primitivos
    i. Números

JavaScript utilza 64bit-floating-point ($\pm 1.7976931348623157 \times 10^{308}$ y $\pm 5 \times 10^{-324}$.

Los números se pueden escribir de diferentes maneras en JavaScript (se puede poner – delante siempre para los negativos):

- Literals:
    - 0, 2, 1000 (base10)
    - 0xff (base16)
    - 0b1010 (base2)
    - 0o377 (base8)
- Float (reales)
    - [digits][.digits][(E|e)[(+|-)]digits]
    - 3.14
    - .37373
    - 4.02e23
    - 99.99E-22
- Aritmética
    - + - * /
    - Some special cases: Nan, Infinity

```
Math.pow(2,53)           // => 9007199254740992: 2 to the power 53
Math.round(.6)           // => 1.0: round to the nearest integer
Math.ceil(.6)            // => 1.0: round up to an integer
Math.floor(.6)           // => 0.0: round down to an integer
Math.abs(-5)             // => 5: absolute value
Math.max(x,y,z)          // Return the largest argument
Math.min(x,y,z)          // Return the smallest argument
Math.random()            // Pseudo-random number x where 0 <= x < 1.0
Math.PI                  // π: circumference of a circle / diameter
Math.E                   // e: The base of the natural logarithm
Math.sqrt(3)             // => 3**0.5: the square root of 3
Math.pow(3, 1/3)         // => 3**(1/3): the cube root of 3
Math.sin(0)              // Trigonometry: also Math.cos, Math.atan, etc.
Math.log(10)             // Natural logarithm of 10
Math.log(100)/Math.LN10  // Base 10 logarithm of 100
Math.log(512)/Math.LN2   // Base 2 logarithm of 512
Math.exp(3)              // Math.E cubed
```

ii. Strings

La variable de javascript para representar texto es la string. Es una secuencia inmutable de valores de 16-bits, y usa el cero indexing (el primer valor en la cadena es el de índice 0). Una string vacía es aquella que tiene tamaño 0, y para representar un char simplemente se utiliza una string de tamaño 1.

Javascript utiliza por defecto el encoding UTF-16, y esto significa que tienen tamaño de 16bits siempre por caracter.

- Los string son iterables, esto significa que puedes realizar un for para recorrerlos carácter a carácter.
- Se pueden definir con varios tipos de comillas: ‘, “, ` .

```
""  // The empty string: it has zero characters
'testing'
"3.14"
'name="myform"'
"Wouldn't you prefer O'Reilly's book?"
"τ is the ratio of a circle's circumference to its radius"
`"She said 'hi'", he said.`
```

- Manera de romper un string en diferentes partes:

```
// A string representing 2 lines written on one line:
'two\nlines'

// A one-line string written on 3 lines:
"one\
 long\
 line"

// A two-line string written on two lines:
`the newline character at the end of this line
is included literally in this string`
```

- Se pueden escapar caracteres especiales con \
  Ejemplo: *'You\'re right, it can\'t be a quote'*

- Operaciones:
  - Se puede usar el operador + para unir diferentes string
  - Se pueden comparar con === o not equal con !==
  - Se pueden comparar con <, >, <=, >=
  - Se puede calcular la longitud con .length, y también hay una serie de funciones

```
let s = "Hello, world"; // Start with some text.

// Obtaining portions of a string
s.substring(1,4)        // => "ell": the 2nd, 3rd, and 4th characters.
s.slice(1,4)            // => "ell": same thing
s.slice(-3)             // => "rld": last 3 characters
s.split(", ")           // => ["Hello", "world"]: split at delimiter string

// Searching a string
s.indexOf("l")          // => 2: position of first letter l
s.indexOf("l", 3)       // => 3: position of first "l" at or after 3
s.indexOf("zz")         // => -1: s does not include the substring "zz"
s.lastIndexOf("l")      // => 10: position of last letter l

// Boolean searching functions in ES6 and later
s.startsWith("Hell")    // => true: the string starts with these
s.endsWith("!")         // => false: s does not end with that
s.includes("or")        // => true: s includes substring "or"

// Creating modified versions of a string
s.replace("llo", "ya")  // => "Heya, world"
s.toLowerCase()         // => "hello, world"
s.toUpperCase()         // => "HELLO, WORLD"
```

- Strings plantilla: se pueden meter variables dentro de la propia string
  Ejemplo:
  let name = “maricarmen”;
  let sayhello = ` Hello ${ name }` ;

- Expresiones regulares

```
/^HTML/;             // Match the letters H T M L at the start of a string
/[1-9][0-9]*/;       // Match a nonzero digit, followed by any # of digits
/\bjavascript\b/i;   // Match "javascript" as a word, case-insensitive
```

RegExp objects define a number of useful methods, and strings also have methods that accept RegExp arguments. For example:

```
let text = "testing: 1, 2, 3";   // Sample text
let pattern = /\d+/g;            // Matches all instances of one or more digits
pattern.test(text)               // => true: a match exists
text.search(pattern)             // => 9: position of first match
text.match(pattern)              // => ["1", "2", "3"]: array of all matches
text.replace(pattern, "#")       // => "testing: #, #, #"
text.split(/\D+/)                // => ["","1","2","3"]: split on nondigits
```

iii. Booleanos

Los valores booleanos representan si algo es verdad o mentira, son valores que sólo pueden tener dos rangos: 1 o 0. Sí o No.

Se utilizan para expresiones condicionales (veremos más adelante).

De momento lo más importante que podemos ver sobre los booleanos, son la tabla de las leyes de Boole:

CONDITION 1	CONDITION 2	AND	OR
FALSE	FALSE	FALSE	FALSE
FALSE	TRUE	FALSE	TRUE
TRUE	FALSE	FALSE	TRUE
TRUE	TRUE	TRUE	TRUE

iv. null y undefined

- null es generalmente el valor que se utiliza para definir la ausencia de valor, de tal manera que nos puede indicar que realmente no hay nada.
- undefined es un tipo de valor que significa que es el valor de una variable que no ha sido inicializada.
- A pesar de tener diferentes matices, se pueden mezclar muy a menudo y el resultado es el mismo. == los compara como iguales, mientras que === sí que da diferente resultado.

2. Objetos

Cualquier variable que no es ningún primitivo se denomina objeto. Generalmente, cualquier conjunto desordenado de primitivos se considera objeto, siendo los arrays un objeto muy específico y valioso en JavaScript, que consideraremos más adelante. Asimismo, hay muchos otros tipos de objetos especiales que iremos viendo a lo largo del curso y más adelante, como son los sets (conjunto de datos), los maps (relaciones clave valor), las expresiones regulares (búsqueda de textos concretos), los dates (para fechas) o los errors (para gestión de errores del programa).

Las funciones y las clases también son tipos especiales de objetos, de tal manera que pueden ser manipuladas por los propios programas de javascript.

3. Conversiones

Table 3-2. JavaScript type conversions

Value	to String	to Number	to Boolean
`undefined`	`"undefined"`	`NaN`	`false`
`null`	`"null"`	`0`	`false`
`true`	`"true"`	`1`	
`false`	`"false"`	`0`	
`""` (empty string)		`0`	`false`
`"1.2"` (nonempty, numeric)		`1.2`	`true`
`"one"` (nonempty, non-numeric)		`NaN`	`true`
`0`	`"0"`		`false`
`-0`	`"0"`		`false`

BIBLIOGRAFÍA

Flanagan, D. (2020). *JavaScript: The Definitive Guide : Master the World's Most-used Programming Language*. O'Reilly Media.

JavaScript | CodeCademy. (s. f.). Codecademy. https://www.codecademy.com/resources/docs/javascript