INTRODUCTION, 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.

Título	Descripción	Enlace
Por qué Javascript e intro	Lectura de la introducción a por qué javascript en este documento apartados 1, 2, 3. Se recomienda instalar los elementos que se proponen para una correcta adecuación a la clase y al sistema de trabajo.	Este documento sección 1-2-3
IA en programación	Sección 4 de este documento. Se explicará mejor en clase, es simplemente un pequeño debate sobre el uso de IA en programación.	Este documento sección 4.
Introducción a las variables	Cómo definir variables en javascript y cómo entender que son dinámicas, no hay que explicitar el tipo de variable que son.	https://www.codecademy.com/resources/docs/javascript/variables
Introducción a las operaciones	Cómo asignar variables y qué operadores principales tenemos entre variables. También operadores de comparación y de lógica.	https://www.codecademy.com/resources/docs/javascript/operators
Introducción a las strings	Introducción a la variable string	https://www.codecademy.com/resources/docs/javascript/strings
Introducción a los data types	Definición de los diferentes tipos de variables existentes: números, booleanos, strings, objetos.	https://www.codecademy.com/resources/docs/javascript/data-types
Documento teórico de abajo basado en el libro de javascript	Resumen de lo expuesto anteriormente y algún dato más, como las palabras clave reservadas.	
Inicializar arrays	Pequeño ejercicio para inicializar arrays, una estructura que sirve para guardar una colección de datos.	https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/store-multiple-values-in-one-variable-using-javascript-arrays
Sumario de operadores en javascript	Resumen de las operaciones anteriores en otro formato. El último apartado de Bitwise Operators no hace falta leérselo.	https://www.w3schools.com/js/js_operators.asp
Crear objetos (intro)	Una pequeña introducción a los objetos, se verá mejor más adelante.	https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-

		javascript/build-javascript-objects
Acceso condicional (OPCIONAL)	Sólo leer el primer epígrafe: Introduction to the JavaScript optional chaining operator. Es una curiosidad sobre cómo usar el operador condicional.	https://www.javascripttutorial.net/javascript-optional-chaining-operator/
Operador Ternario (OPCIONAL)	Se trata del operador ternario, que simplifica el uso de los if else según qué ocasiones.	https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/use-the-conditional-ternary-operator

1. ¿Por qué JavaScript?

Esta pregunta es muy importante, debemos entender que hay muchísimos **lenguajes de programación, y todos son igualmente válidos**. Algunos son más complejos, otros más sencillos de aprender, otros más eficientes, otros más encaminados a aplicaciones concretas. Todos los lenguajes son buenos y todos nos abren la puerta al mundo de la tecnología. Generalmente, si sabemos un lenguaje, casi podemos descifrar, con un poco de esfuerzo y ganas, cualquier otro. (Cuidado con C!)

Dicho esto, existen estudios (a día de hoy) que hablan de la **escasez de programadores en el ámbito de la web** (oh, amigo, internet, claro). Y uno de los lenguajes con los que se comunican los ordenadores para poder sacar por los navegadores información de manera bonita y útil es JavaScript. Antes se usaba más PHP, pero ha caído en desgracia y ahora todo se basa en JavaScript.

JavaScript es un nombre que puede parecer confuso, ya que el lenguaje, aunque basa sus raíces "sintácticas" en Java, no tiene mucho que ver con el propio lenguaje de Java. Además, ahora es muchísimo más utilizado que Java, sin duda. Originalmente, JavaScript fue un lenguaje creado por el actual Oracle para usar Netscape, ahora conocido como Mozilla. **El estándar de JavaScript se denomina como ECMAScript.**

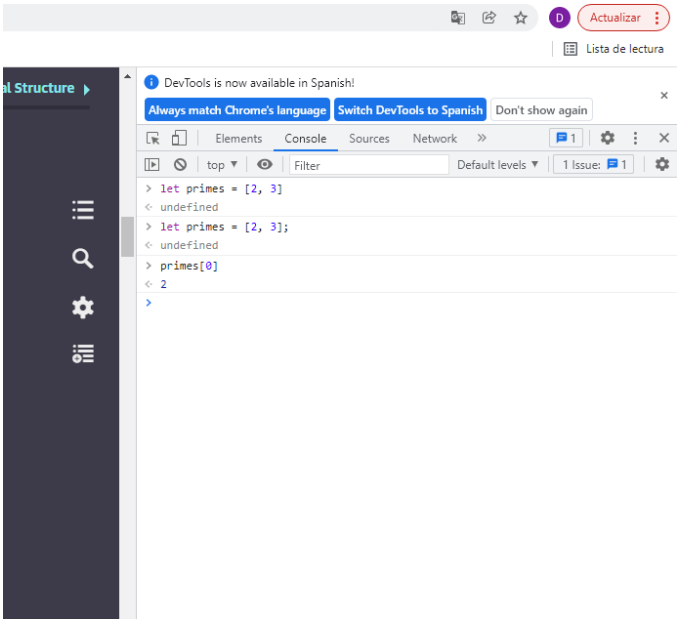
Originalmente, JavaScript se ejecutaba únicamente en los navegadores, al contrario que PHP, que lo hacía en los servidores. Por ello, ésta sigue siendo la ejecución más normal del código. El lenguaje se encuentra en el navegador y por medio de peticiones HTTP obtiene datos y los expone en el navegador haciendo uso de CSS y HTML.

Hoy en día, viendo la importancia que cobraba el lenguaje se creó otra manera de trabajar con JavaScript en el lado de los servidores, y se llamó Node. Node ahora es una elección típica para trabajar con servidores web y eso os tocará aprender más adelante.

2. ¿Dónde vamos a trabajar?

Aquí, cada maestrillo con su librillo, podéis usar lo que consideréis más importante. Sin embargo, nosotros utilizaremos **VisualStudio code**, para prepararnos para nuestras futuras clases, y por ello nos suscribiremos a eso. Si alguna persona prefiere trabajar con otro IDE adelante, pero los problemas que deriven de esa razón deberán ser solventados por el propio usuario. (aunque intentaremos ayudar)

- También se puede usar la consola del propio navegador (en este caso, utilizaremos Google Chrome)



- Nos descargaremos Node.js para tenerlo a punto: https://nodejs.org/en/. Esto nos puede ser útil a veces cuando necesitemos probar cosas de manera interactiva.
- Y, por último, Visual Studio Code: nos lo descargaremos en: https://code.visualstudio.com/

3. Hello World

Vamos a hacer una prueba sencilla:

- Crea un archivo en vscode llamado hello_world.js
- Escribe un log de consola con: *console.log("hello world")*
- Crea un archivo en vscode llamado hello_world.html
- Escribe la detección del archivo de jso con: *<script src="hello.js"></script>*

- Ahora ejecuta el código html y saca la pantalla de herramientas para desarrolladores. Deberías de tener algo así:

4. IA

a. Introducción

Hoy en día, la inteligencia artificial ha llegado también a los programadores, de momento con la incertidumbre de saber si acabará supliéndolos o no por completo. Lo que está claro es que hoy en día es una buena herramienta para mejorar los procesos y la eficiencia de los programadores, siempre y cuando se utilice con compresión y supervisión.

Cuando hablamos de programar con IA, se trata de conseguir que la inteligencia artificial nos cree código para lo que queremos hacer, y se me ocurren hoy en día dos maneras más típicas de hacerlo:

- Preguntar a un Bot tipo chatgpt para que te resuelva el código
- Instalarte un asistente de IA tipo Copilot para que te vaya sugiriendo código conforme lo vas haciendo.

b. Precauciones

El código generado por IA, a pesar de poder mejorar nuestros procesos y nuestros tiempos, no está exento de problemas, por ello, a la hora de usarlo hay que tener cuidado y tener en cuenta una serie de precauciones:

- Investigaciones sobre copilot, por ejemplo, aducen que el 40% de las veces el código tiene fallas de seguridad
- Siempre se debería revisar lo que se ha escrito en código con diferentes herramientas: pylint, comprobadores de dependencias, comprobadores de seguridad...
- A veces pueden generar estructuras demasiado complejas para lo que se está llevando a cabo. Las nomenclaturas también deben ser coherentes y revisadas.
- Siempre hay que entender el código creado por IA, si no, puede dar lugar a errores o a no entender el propio programa de uno.
- Tener en cuenta los problemas con los copyrights, ¿se está usando código de otras personas con la IA utilizada?
- Lectura extra: https://madappgang.com/blog/chat-gpt-code-errors/

c. Github copilot

Es una herramienta que se puede instalar en vscode, y por eso nos resulta interesante en nuestras prácticas. Sus características principales son:

- Proporciona sugerencias de código en tiempo real
- Se integra fácilmente con vscode y con otros
- Sus sugerencias están basadas en el propio contexto del proyecto, de tal manera que crea estructuras y nombres coherentes
- Mejora productividad ya que en muchos casos (testing) ahorra tiempo.
- Cierta privacidad y seguridad, según otros revisores

5. 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();

6. 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*.

a. 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.

b. 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.

c. 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`

EJERCICIOS

Ejercicios de iniciación 1	https://www.w3schools.com/js/exercise_js.asp?filename=exercise_js_variables1
Ejercicios de iniciación 2	https://www.w3schools.com/js/exercise_js.asp?filename=exercise_js_variables2
Ejercicios de iniciación 3	https://www.w3schools.com/js/exercise_js.asp?filename=exercise_js_variables3
Ejercicios de iniciación 4	https://www.w3schools.com/js/exercise_js.asp?filename=exercise_js_variables4
Ejercicios de iniciación 5	https://www.w3schools.com/js/exercise_js.asp?filename=exercise_js_variables5
Ejercicios de iniciación 6	https://www.w3schools.com/js/exercise_js.asp?filename=exercise_js_operators1
Ejercicios de iniciación 7	https://www.w3schools.com/js/exercise_js.asp?filename=exercise_js_operators2
Ejercicios de iniciación 8	https://www.w3schools.com/js/exercise_js.asp?filename=exercise_js_operators3

Ejercicios de iniciación 9	https://www.w3schools.com/js/exercise_js.asp?filename=exercise_js_operators4
Ejercicios de iniciación 10	https://www.w3schools.com/js/exercise_js.asp?filename=exercise_js_operators5
Ejercicio 11	Evalúa los statements propuestos en el script "expressions.js" con nodejs

BIBLIOGRAFÍA

Castles, R. (2022, 4 noviembre). *Why You Should Apply Caution When Using AI in Code Development*. Spiceworks Inc. https://www.spiceworks.com/tech/artificial-intelligence/guest-article/why-you-should-apply-caution-when-using-ai-in-code-development/

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

Polo, J. D., & Polo, J. D. (2023, 3 agosto). *Inteligencia Artificial para desarrolladores, recursos que ayudan a programar*. WWWhat's New. https://wwwhatsnew.com/2023/06/01/inteligencia-artificial-para-desarrolladores-recursos-que-ayudan-a-programar/#:~:text=En%20el%20mundo%20de%20la,la%20vida%20de%20los%20desarrolladores

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

FreeCodeCamp.org. (s. f.-b). https://www.freecodecamp.org/

JavaScript Optional Chaining Operator (?.). (2023, 17 diciembre). JavaScript Tutorial. https://www.javascripttutorial.net/javascript-optional-chaining-operator/

W3Schools.com. (s. f.). https://www.w3schools.com/js/js_operators.asp