

A yellow square containing the letters 'JS' in a large, bold, black sans-serif font.

# JS

# JAVASCRIPT

Tema 2 – Programación básica

# JAVASCRIPT



- Variables.
  - Distinción entre mayúsculas y minúsculas.
  - Utiliza Unicode.
  - Tipos de declaración:
    - var
    - let
    - const
  - Identificadores:
    - Comienzan por letra, guión bajo (\_) o símbolo del dólar (\$).
    - Puede contener letra, \_, \$ o dígitos numéricos 0-9.

# JAVASCRIPT

A yellow square containing the letters 'JS' in a bold, black, sans-serif font.

- Variables.
  - Tipado dinámico: en función de la asignación.
  - Declaración sin asignación: valor **undefined**.
  - Acceso a variable no declarada: **ReferenceError**.
  - `undefined` se convierte en **NaN** en contextos numéricos.
  - El valor **null**:
    - Propio de objetos.
    - El valor **null** es **0** en contextos numéricos y **false** en contextos booleanos.
  - Ámbitos:
    - Local vs Global.

# JAVASCRIPT

A yellow square containing the letters 'JS' in a bold, black, sans-serif font.

- Variables.
  - Hoisting o elevación.
    - Detección y elevación de declaración de las variables al principio de su ámbito.
    - El uso de una variable declarada en líneas posteriores genera un valor **undefined**.
  - Nota: *hoisting aplicado a funciones*:
    - Sólo eleva declaraciones, no expresiones (asignación de una función a una variable).

# JAVASCRIPT



- Tipos de datos primitivos (definición ECMAScript):
  - String
  - Number
  - Boolean
  - BigInt
  - Null
  - Undefined
  - Symbol
  - Object
- Otros:
  - Funciones
  - Arrays
- Operador typeof

# JAVASCRIPT

A yellow square containing the letters 'JS' in a bold, black, sans-serif font.

- Creación con constructor:
  - `let x = new String('cadena');`
  - `let x = new Number(10.5);`
  - `let x = new Boolean(true);`
- Creación con función:
  - `let x = BigInt(138);` // es una función se debe utilizar sin **new**

# JAVASCRIPT

A yellow square with the letters 'JS' in black, representing the JavaScript logo.

- **Conversión de datos:**

- Operador + aplicado a cadenas y números, convierte los números en cadenas.
- Operadores -, \*, /, \*\*, % aplicado a cadenas y números, convierte las cadenas en números si es posible.
- Funciones de conversión:
  - parseInt
  - parseFloat
- Métodos de conversión:
  - toString

# JAVASCRIPT



- Manejo de cadenas de caracteres (string):
  - Inclusión de comillas.
  - Caracteres de escape:
    - \n
    - \t
    - \'
    - \"
    - \\



# JAVASCRIPT



- Manejo de cadenas de caracteres (string):
  - Son inmutables.
  - Operadores:
    - + (equivalente a método **concat**)
    - +=
  - Atributos:
    - length

# JAVASCRIPT



- Manejo de cadenas de caracteres (string):
  - Métodos:
    - concat
    - toUpperCase
    - toLowerCase
    - indexOf
    - lastIndexOf
    - substring
    - split
    - join
    - replace
    - replaceAll

# JAVASCRIPT

JS

- Manejo de cadenas de caracteres (string):
  - Métodos:
    - startsWith
    - endsWith
    - toString
    - match
    - matchAll
  - Interpolación:
    - `const cadena = `Mi nombre es ${nombre}`;`

# JAVASCRIPT



- Manejo de números:
  - Propiedades:
    - NaN → Not a Number
    - Infinity
    - -Infinity
  - Métodos:
    - toFixed → Para ajustar y redondear decimales.
    - isNaN
    - toString
    - match
    - matchAll

# JAVASCRIPT

JS

## ■ Operadores:

- De asignación
  - =, +=, -=, ...
- De comparación
  - ==, !=, >, <, ===, !==, ...
- Aritméticos
  - +, \*, ++, %...
- De bit.
  - &, |, ^
- Lógicos
  - &&, || → Evaluación en cortocircuito.

# JAVASCRIPT

JS

## ■ Operadores:

- De cadena
  - +, +=
- Condicionales (ternarios)
- Operador coma (,) → Se utiliza en los bucles para realizar múltiples actualizaciones al finalizar la iteración.
- Unarios
  - delete
  - typeof
  - void
- Relacionales:
  - in
  - instanceof

# JAVASCRIPT

JS

- Precedencia de operadores.
  - Determina como se resuelven las expresiones.
  - Se pueden evitar con el uso de paréntesis.
  - Tabla de precedencias:
    - [https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Expressions\\_and\\_Operators#precedencia de los operadores](https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Expressions_and_Operators#precedencia_de_los_operadores)

# JAVASCRIPT

A yellow square containing the letters 'JS' in a bold, black, sans-serif font.

- Arrays:
  - Pueden ser heterogéneos.
  - Los índices comienzan en 0.
  - Creación:
    - `let array = [1,true,"Película"];`
    - `let array = Array(1, true, "Película");`
  - Atributos:
    - `length`
  - Acceso:
    - `array[posición]`
  - Operador delete: vacía el valor de la posición indicada.



# JAVASCRIPT



- Arrays:
  - Métodos:
    - map
    - filter
    - **foreach**
    - find
    - sort
    - some
    - concat
    - includes
    - join

# JAVASCRIPT



- Arrays:
  - Métodos:
    - reduce
    - indexOf
    - findIndex
    - fill
    - push
    - pop
    - shift
    - unshift
    - slice

# JAVASCRIPT



- Arrays:
  - Métodos:
    - reverse
    - splice
    - lastIndexOf
    - flat
    - isArray
    - from

# JAVASCRIPT

A yellow square containing the letters 'JS' in a bold, black, sans-serif font.

- Objetos (introducción):

- Creación (1):

```
let objeto = {  
    nombre : "JavaScript",  
    tipo : "Multiparadigma"  
}
```

- Creación (2):

```
let objeto = new Object();  
objeto.nombre : "JavaScript";  
objeto.tipo : "Multiparadigma";
```

# JAVASCRIPT

A yellow square containing the letters 'JS' in a bold, black, sans-serif font.

- Objetos (introducción):
  - Acceso (tanto para lectura, como asignación y creación):
    - `objeto.nombre`
    - `objeto["nombre"]`
  - Acceso con el operador **in**:
    - bucles `for...in` Recorre todas las propiedades de un objeto y su cadena de prototipos.
    - `Object.keys(objeto)` → Devuelve un array con todas las claves del objeto.
    - `Object.getOwnPropertyNames(objeto)` Devuelve un array con todos los nombres de las propiedades del objeto.

# JAVASCRIPT

A yellow square with the letters 'JS' in black, representing the JavaScript logo.

- Objetos iterables:
  - Se pueden recorrer con bucles:
    - String
    - Array
    - TypedArray
    - Map
    - Set
  - arguments (es el objeto en el que se recogen los parámetros de una función, similar a un array).

# JAVASCRIPT

JS

- Sentencias de control de flujo:
  - if-else
  - for
  - while
  - do-while
  - switch
  - for-in
  - for-of
  - break
  - continue

# JAVASCRIPT



## ■ Enlaces:

- [https://developer.mozilla.org/es/docs/Learn/JavaScript/First\\_steps/Variables](https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/Variables)
- [https://developer.mozilla.org/es/docs/Web/JavaScript/Data\\_structures](https://developer.mozilla.org/es/docs/Web/JavaScript/Data_structures)
- [https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global\\_Objects/String](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/String)
- [https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Expressions and Operators](https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Expressions_and_Operators)
- [https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global\\_Objects/Array](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Array)
- <https://dev.to/gdcodev/25-metodos-de-arrays-en-javascript-que-todo-desarrollador-debe-conocer-4a2d>
- [https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Working with objects](https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Working_with_objects)