

# “Guia de Javascript”

## Capítulo 1:

### 0) Introducción - Qué es JavaScript?

- Lenguaje de programación
  - Interpretado
  - Orientado a objetos
  - Imperativo
  - Case Sensitive (Upper and Lower Case)
  - Basado en prototipos / Instancias / Classless.
  - Tipado débil
  - Lenguaje Dinámico.
- 
- ECMAScript (5.1 y 6, 7, 8, Next)

### 1) ¿Cómo se usa JavaScript?

- En línea
- Como contenido en la etiqueta <script>
- Como contenido en un archivo .js
- Con un Require

### 2) Variables



TIPOS DE DATOS EN JAVASCRIPT	
Numbers	3, 25, 2.546
Boolean	true, false
Strings	"Estamos en primavera"

- Casos especiales de datos: Undefined, Null, NaN;
- Scope
- Hoisting
- Tipos de variables (Let-Var-Const)
- Crear múltiples variables (Separándolas con coma) -> Let num1 = 10, num2 = 20;
- Pruebas con Prompt

## 5) Operadores en JavaScript (Básico)

- Operadores de Asignación
- Operadores Aritméticos

## 6) Concatenación

- Con el signo +
- Para números forzamos string ("" + 9 + 4)
- Con Concat
- Con backticks y la variable entre \${ }

## 7) Backticks (FN + ESC)

- Sin Backticks: "línea 1\nlínea 2"
- Con Backticks: ``línea 1  
línea 2``
- Definición
- Escape de comillas simples
- Escape de comillas dobles

## 8) Operadores (Intermedio)

- Operadores Lógicos
- Operadores de Comparación

Equality ( a == b )
Inequality ( a != b )
Identity ( a === b )
Non-identity ( a !== b )
Greater than ( a > b )
Greater than or equal ( a >= b )
Less than ( a < b )
Less than or equal ( a <= b )

## 9) Camel Case

- Definición
- Usos
- Ejemplo

## 10) Condicionales

- Definición y usos
- If
- Else If
- Else

# Capítulo 2:

## 11) Arrays

- Definición y usos
- Formas de crear un array

## 12) Arrays Asociativos

- Definición
- Sintaxis y usos

## 13) Bucles e interacción

- Definición y usos
- Sentencia while
- Sentencia do while
- Sentencia for
- Sentencia for in
- Sentencia for of

- 
- Sentencia break
  - Sentencia label
  - Sentencia continue

## 14) Funciones

- Definición y usos
- Formas de crear una función (y llamarlas)
- Return
- Parámetros
- Scope

- Funciones flecha

#### 15) ForEach

- Sintaxis y uso
- Ventajas respecto a FOR
- Desventajas (Break, Continue)

#### 16) POO

- Definición y usos
- Ejemplos

#### 17) Conceptos basicos de POO

- Clase
- Objeto
- Atributo
- Método
- Constructor
- Instanciación

#### 18) Características de la POO

- Abstracción
- Modularidad
- Jerarquía

#### 19) Otros conceptos de POO

- Polimorfismo
- Herencia

#### 20) Otros conceptos de POO

- Encapsulamiento
- Métodos accesorios (Getters, Setters)

#### 21) Métodos de Cadena

- **concat()** - junta dos o más cadenas y retornan una nueva
- 

- **startsWith()** - si una cadena comienza con los caracteres de otra cadena, devuelve true, sino false.
  - **endsWith()** - si una cadena puede encontrarse dentro de la otra cadena, devuelve true, sino false
  - **includes()** - devuelve el índice del primer carácter de la cadena, si no existe, devuelve -1
  - **lastIndexOf()** - devuelve el último índice del primer carácter de la cadena, si no existe, devuelve -1
  - **padStart()** [Propuesta de ECMA] - Rellenar cadena al principio con caracteres deseados
  - **padEnd()** - [Propuesta de ECMA] - Rellenar cadena al final con los caracteres deseados
  - **repeat()** - Devuelve la misma cadena pero repetida la cantidad de veces que le indiquemos.
- 

- **split()** - Divide la cadena como le pidamos
- **substring()** - Nos retorna un pedazo de la cadena que seleccionamos
- **toLowerCase()** - convierte una cadena a minúscula
- **toUpperCase()** - convierte una cadena a mayúscula
- **toString()** - método que devuelve una cadena que representa al objeto específico
- **trim()** - elimina los espacios en blanco de una cadena
- **trimEnd()** - elimina los espacios en blanco al final de una cadena
- **trimStart()** - elimina los espacios en blanco al comienzo de una cadena
- **valueOf()** - retorna el valor primitivo de un objeto string

## 22) Métodos de Arrays

- **pop()** - elimina el último elemento de un array y lo devuelve.
  - **shift()** - elimina el primer elemento de un array y lo devuelve.
  - **push()** - agrega un elemento al array al final de la lista.
  - **reverse()** - invierte el orden de los elementos al inicio del array
  - **unshift()** - agrega uno o más elementos al inicio del array, y devuelve la nueva longitud del array.
  - **sort()** - ordena los elementos de un arreglo (array) localmente y devuelve el arreglo ordenado
  - **splice()** - cambia el contenido de un array eliminando elementos existentes y/o agregando nuevos elementos
- 

- **join()** - une todos los elementos de una matriz (u objeto similar) en una cadena y la devuelve.
  - **slice()** - devuelve una parte del array dentro de un nuevo array empezando por inicio hasta fin (fin no incluido)
  - Métodos ya vistos en cadenas: **toString()**, **indexOf()**, **lastIndexOf()**, **includes()**
-

- **filter()** - crea un nuevo array con todos los elementos que cumplan
  - **forEach()** - ejecuta la función callback una vez por cada elemento del array
- 

## 23) Objeto Math - Básico

### > Metodos:

- **sqrt()** - Devuelve la raiz cuadrada positiva de un numero
- **cbrt()** - Devuelve la raiz cubica de un numero
- **max()** - Devuelve el mayor de cero o mas numeros.
- **min()** - Devuelve el mas pequeño de cero o mas numeros
- **random()** - Devuelve un numero pseudo-aleatorio entre 0 y 1.
- **round()** - Devuelve el valor de un numero redondeado al entero mas cercano
- **fround()** - Devuelve la representacion flotante de precisión simple mas cercana de un número
- **floor()** - Devuelve el mayor entero menor que o igual a un número
- **trunc()** - Devuelve la parte entera del número x, la eliminacion de los dígitos fraccionarios.

### > Propiedades:

- **PI** - ratio de la circunferencia de un círculo respecto a su diámetro, aproximadamente 3.14159...
- **SQRT1\_2** - Raiz cuadrada de 1/2 ; Equivalente, 1 sobre la raíz cuadrada de 2, aproximadamente 0.707...
- **SQRT2** - Raiz cuadrada de de 2, aproximadamente 1.414...
  
- **E** - Constante de EULER, la base de los algoritmos naturales 2,71828
- **LN2** - Logaritmo natural de 2, aproximadamente 0.693
- **LN10** - Logaritmo natural de 10, aproximadamente 2.303
- **LOG2E** - Logaritmo de E con base 2, aproximadamente 1.443
- **LOG10E** - Logaritmo de E con base 10, aproximadamente 0.434

## 24) Console.

### ----- funciones de registro:

- **assert()** = aparece un mensaje de error en la consola si la afirmación es falsa, si la afirmación es verdadera no aparece nada (no estandar)
- **clear()** = limpia la consola
- **error()** = muestra un mensaje de error en la consola web.
- **info()** = emite un mensaje informativo en la consola web. en firefox y chrome, se muestra un pequeño icono "i" junto a estos elementos en el registro de la consola web.
- **log()** = muestra un mensaje en la consola web (o del intérprete de javascript).
- **table()** = esta funcion toma un argumento obligatorio: data, que debe ser un array o un objeto, y un parametro adicional: columns y nos muestra una tabla en consola.
- **warn()** = imprime un mensaje de advertencia en la consola web.
- **dir()** = despliega una lista interactiva de las propiedades del objeto javascript especificado. (no estandar)

### ----- funciones de conteo:

- **count()** = registra el numero de veces que se llama a count(). esta funcion toma como argumento opcional una etiqueta.
- **countReset()** = resetea el contador console.count()

### ----- funciones de agrupacion:

- **group()** = crea un nuevo grupo en linea en el registro de la consola web.
- **groupEnd()** = remueve un grupo en linea en el registro de la consola web.
- **groupCollapsed()** = crea un grupo en linea pero contraido, el usuario debe expandirlo para verlo.

### ----- funciones de temporización:

- **time()** = inicia un temporizador.
- **timeEnd()** = registra el valor actual de un temporizado.
- **timeLog()** = detiene un temporizador.

## 25) ADENTRANDONOS EN EL DOM:

- Definición
- Concepto Extendido

- **Nodo (un nodo en el DOM es cualquier etiqueta del cuerpo, como un párrafo, el mismo body, incluso las etiquetas de una lista**

**\*Document:** el nodo document es el nodo raíz, a partir del cual derivan el resto de nodos.

**\*Element: nodos** definidos por etiquetas html.

**\*Text:** el texto dentro de un nodo element se considera un nuevo nodo hijo de tipo text (texto)

**\*Attribute:** los atributos de las etiquetas definen nodos, en (Javascript no los veremos como nodos, sino como información asociada al nodo de tipo element)

**\*Comentarios y otros:** los comentarios y otros elementos como las declaraciones doctype en cabecera de los elementos HTML generan nodos.

## 26) **Document - Métodos de Selección de elementos**

- **getElementById()** - Selecciona un elemento por ID.
- **getElementsByTagName()** - Selecciona todos los elementos que coincidan con el nombre de la etiqueta especificada.
- **querySelector()** - Devuelve el primer elemento que coincida con el grupo especificado de selectores.
- **querySelectorAll()** - Devuelve todos los elementos que coincidan con el grupo especificado de selectores.

## 27) **Métodos para Definir, Obtener y Eliminar valores de atributos.**

- **setAttribute()** - Modifica el valor de un atributo.
- **getAttribute()** - Obtiene el valor de un atributo.
- **removeAttribute()** - Remueve el valor de un atributo.

## 28) **Atributos Globales**

- **Contenteditable** - indica si el elemento puede ser modificable por el usuario (bool)
- **Dir** - indica la direccionalidad del texto



- **hidden** - indica si el elemento aun no es, o ya no es, relevante.
- **tabindex** - indica si el elemento puede obtener un focus de input
- **title** - contiene un texto con información relacionada al elemento al que pertenece

## 29) **Atributos de Inputs**

- **className**
- **value**
- **type**
- **accept**
- **form**
- **minLength**
- **placeholder**
- **required**

## 30) **Atributo Style**

- **Usos y ejemplos**
- **propiedades Camel Case**

## 31) **Clases, ClassList y Métodos de classList**

- **Definición y usos**
- **add()** - añade una clase
- **remove()** - remueve una clase
- **item()** - devuelve la clase del indice especificado
- **contains()** - verifica si ese elemento posee o no, la clase especificada.
- **replace()** - reemplaza una clase por otra
- **toggle()** - si no tiene la clase especificada, la agrega, si ya la tiene, la elimina.

### 32) Obtención y Modificación de Elementos

- **textContent** - devuelve el texto de cualquier nodo.
- **innerHTML** - devuelve el contenido html de un elemento.
- **outerHTML** - devuelve el código HTML completo del elemento.

### 33) Creación de Elementos

- **createElement()**
- **createTextNode()**
- **appendChild()**
- **createDocumentFragment()**

### 34) Obtención y modificación de Childs (hijos)

- **firstChild**
- **lastChild**
- **firstElementChild**
- **lastElementChild**
- **childNodes**
- **children**

### 35) Métodos de Childs (hijos)

- **replaceChild()**
- **removeChild()**
- **hasChildNodes()**

### 36) Propiedades de Parents (padres)

- **parentElement**
- **parentNode**

### 37) Propiedades de Sibling (Hermanos)

- nextSibling
- previousSibling
- nextElementSibling
- previousElementSibling

### 38) Nodos - Extras

- closest()

## **FINAL DEL PRIMER EPISODIO**

---