



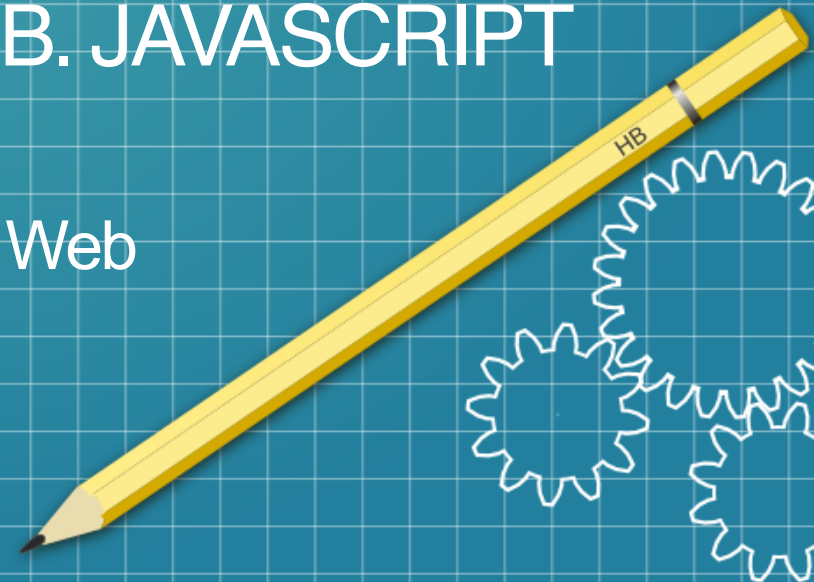
FPA

FORMACIÓN  
PROFESIONAL  
ANDALUZA

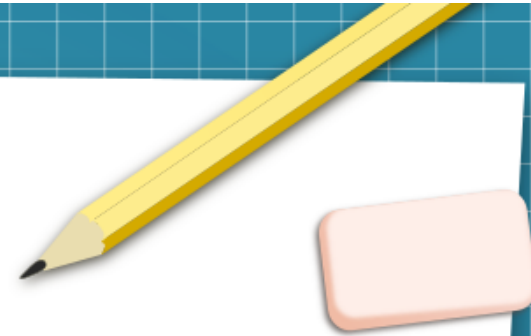
# UD1.- ARQUITECTURAS Y LENGUAJES DE PROGRAMACIÓN EN CLIENTES WEB. JAVASCRIPT

2º curso – CFGS Desarrollo de Aplicaciones Web

Profesora: María Ojeda García



# Objetivos de la unidad



- Conocer las diferentes **alternativas** existentes para la **navegación web** en función de las diferentes tecnologías web que se ejecutan en un cliente.
- Reconocer las **capacidades** de la **ejecución de código en el lado del cliente** de acuerdo a los componentes arquitectónicos de un navegador web.
- Identificar los **principales lenguajes y tecnologías** de programación en **entorno cliente**.
- Conocer las **técnicas de integración** del código con documentos HTML.
- Conocer las **principales características** del lenguaje **JavaScript**.
- Dominar la **sintaxis** básica del lenguaje.
- Comprender y utilizar los distintos tipos de **variables** y **operadores** presentes en el lenguaje **JavaScript**.
- Conocer los diferentes **sentencias condicionales de JavaScript** y saber realizar operaciones complejas con ellas.

# 1. DISEÑO y DESARROLLO web

## DISEÑO WEB

- Creación y organización de contenido: la arquitectura de la información.
- Aspectos importantes:
  - Organización
  - Funcionalidad
  - Accesibilidad



## DESARROLLO WEB

- Nuevas tecnologías.
- Diseño gráfico.
- Programación.
- Diseño de interfaces.
- Usabilidad.
- Recursos

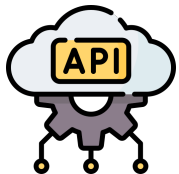


## 2. SITIOS WEB

### LADO DEL SERVIDOR

Contiene el hardware y software del servidor web, elementos de programación y tecnologías:

- Scripts y programas CGI.
- Programas API del servidor
- Módulos de Apache
- Servlets de Java
- Lenguajes de scripting:
  - PHP
  - ASP
  - JavaScript (Node.js)



### LADO DEL CLIENTE

Se refiere a los navegadores web:

- HTML y CSS
- Lenguajes de scripting:
  - JavaScript
  - TypeScript (compilado en JavaScript)
  - VBScript
- Aplicaciones de Ayuda.
- Programas del API del navegador.
- Controles ActiveX.



# 4. CARACTERÍSTICAS DE JAVASCRIPT



¿Qué se puede o no se puede hacer con JavaScript?

Lenguaje interpretado  
en el navegador:  
puede estar  
deshabilitado.

No puede escribir  
ficheros en el  
servidor.

Reacciona a la  
interacción del  
usuario.

Controla múltiples  
ventanas, marcos,  
plugins, applets ...

Pre-procesa datos en  
el cliente.

Puede solicitar  
ficheros al servidor

# 4. CARACTERÍSTICAS DE JAVASCRIPT



## Compatibilidad

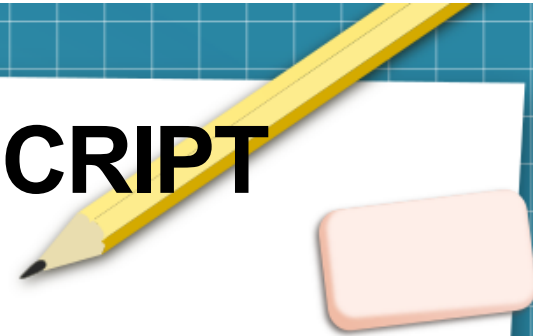
Prácticamente todos los navegadores lo soportan: debemos asegurarnos

Hay incompatibilidades entre navegadores

Algunos dispositivos móviles no pueden ejecutar JavaScript

Puede desactivarse la ejecución de código en el usuario.

# 4. CARACTERÍSTICAS DE JAVASCRIPT



## Seguridad

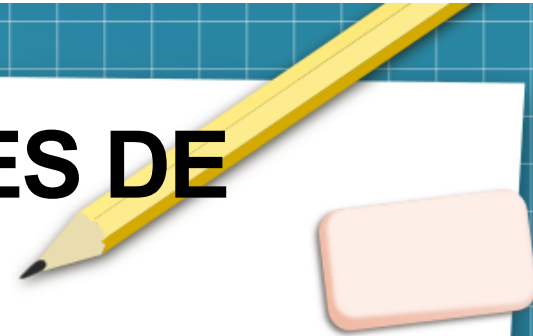
Se ejecuta el código en un “espacio seguro de ejecución”: la web

Scripts restringidos por la política del “mismo origen”

El motor de JavaScript es quien interpreta el código en el navegador: el responsable



# 5. HERRAMIENTAS Y UTILIDADES DE PROGRAMACIÓN



Editor de Texto:

- Edición de código en diferentes lenguajes.
- Sintaxis de colores.
- Verificación de sintaxis.
- Diferencia comentarios del resto de código.
- Genera partes de código automáticas.
- Utilidades adicionales.

Ejemplos de editores:

Visual Studio Code, Aptana Studio, Sublime Text, Eclipse, Netbeans...



## 6. INTEGRACIÓN DE CÓDIGO JAVASCRIPT EN UNA PÁGINA WEB

Etiquetas `<script>` en HTML

`<script>` o ~~`<script type="text/javascript">`~~ **Desuso**  
Código javascript  
`</script>`

Navegador no soportado

`<noscript>`  
Su navegador no soporta JavaScript  
`</noscript>`

## 6. INTEGRACIÓN DE CÓDIGO JAVASCRIPT EN UNA PÁGINA WEB

### Fichero externos

```
<script src="ruta/archivo1.js"></script>  
<script src="ruta/archivo2.js"></script>
```

### Ventajas de usar un fichero externo

- Carga más rápida de páginas.
- Separación entre la capa de diseño y la capa lógica.
- Se puede compartir código entre páginas.
- Facilidad para depuración de errores.
- Modularidad.
- Seguridad.

## 6. INTEGRACIÓN DE CÓDIGO JAVASCRIPT EN UNA PÁGINA WEB

### Protección de código JAVASCRIPT



Incluir mensaje de Copyright

Promocionar el código

Ofuscar el código  
<https://javascript2img.com/>

# 7. FUNDAMENTOS DE JAVASCRIPT



El lenguaje JavaScript desde 1997 se rige por un estándar denominado ECMA, que se encarga de gestionar las especificaciones de este lenguaje y que lo soportan la mayoría de los navegadores.

El estándar de ECMAScript se lanzó en junio de 1997 con la versión 1. La versión que agregó más cambios fue la versión ECMAScript 6 (**ES6**) en junio del 2015 y la última es la versión ECMAScript 2025 (**ES2025**).

[https://www.w3schools.com/js/js\\_2025.asp](https://www.w3schools.com/js/js_2025.asp)

## 8. ENTRADA Y SALIDA EN NAVEGADORES

- **alert():** permite mostrar al usuario mediante una ventana independiente, información literal o una variable.

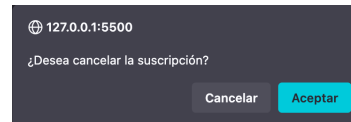
```
alert("Hola mundo");
```

- **console.log():** permite mostrar información en la consola de desarrollo.

```
console.log("Hola mundo");
```

- **confirm():** se activa un cuadro de diálogo que contiene los botones de Aceptar y Cancelar. Al pulsar Aceptar devuelve true y Cancelar devuelve false.

```
let respuesta;  
respuesta=confirm("¿Desea cancelar la suscripción?");  
alert("Ha pulsado " + respuesta);
```



- **prompt():** se activa un cuadro en el que se pide que se introduzca un dato

```
let respuesta;  
respuesta=prompt("Introduzca la provincia");  
alert("La provincia es: " + respuesta);
```



## 9. Sintaxis del lenguaje - COMENTARIOS



Los comentarios pueden hacerse con // para una línea y /\* \*/ para varias líneas.

```
// Esto es un comentario en Javascript de una línea
```

```
/* Esto es un comentario en  
Javascript multilínea */
```

## 9. Sintaxis del lenguaje - VARIABLES

**Variables.** Permiten almacenar distintos valores en cada momento

- **let:** La variable es accesible únicamente en el bloque (block scoped) donde se ha declarado.

```
let x = 10;

if (true) {
  let x = 20; // Esta variable solo existe dentro de este bloque (el if)
  console.log(x); // Imprime 20
}

console.log(x); // Imprime 10, ya que fuera del bloque if, x es 10
```

- **var:** Es accesible por todos los lugares de la función y si es declarada fuera de la función, la variable es accesible para todas las funciones del código.

```
var x = 10;

if (true) {
  var x = 20; // Esta variable es accesible en toda la función
  console.log(x); // Imprime 20
}

console.log(x); // Imprime 20, ya que el `var` dentro del if afecta al `x` fue
```

- **const:** Declara variables locales dentro del bloque y su valor no puede cambiar.
- **Variables sin declarar:** JavaScript permite usar variables no declaradas, es como si se declararan con **var**.

## 9. Sintaxis del lenguaje - VARIABLES



### Crear variables

```
var edad;  
let edad1, edad2, edad3;
```

### Asignar valor a una variable:

```
edad=15;  
nombreApellidos="María";
```

### Crear variable y asignar valor:

```
var edad=15;  
let edad1, edad2, edad3=23;  
const nombre="Alba prieto";
```

## 9. Sintaxis del lenguaje - VARIABLES



- Utilizaremos para la creación de variables, funciones... el estilo CamelCase (lowerCamelCase). Ejemplo// nombreApellidos.
- Formadas por caracteres alfanuméricos y `_`. No se utilizan signos, espacios, %, \$, etc
- No pueden empezar por número y no suelen empezar por mayúscula.
- No tiene asociado un tipo. Podemos cambiar de número a cadena, a boolean, etc.
- No utilizar palabras reservadas

## 9. Sintaxis del lenguaje - “use strict”

- JavaScript Ecma Script 6 o ES6 incorpora el llamado “modo estricto”. Si en algún lugar del código se indica la sentencia “use strict”, indica que ese código se ejecutará en modo estricto, es decir, que es obligatorio declarar las variables antes de su utilización.

```
"use strict";  
pi=3.14;           // Da error
```

( pi no está declarada )

## 9. Sintaxis del lenguaje - Tipo de datos



- ✓ Undefined
- ✓ Boolean
- ✓ Number
- ✓ BigInt
- ✓ String
- ✓ Null
- ✓ Object
- ✓ Symbol
- ✓ Function

## 9. Sintaxis del lenguaje - Tipo de datos primitivos

- ✓ **Undefined:** representa una variable que no se le ha asignado un valor o no ha sido declarada.
- ✓ **Boolean:** representa un valor lógico y puede tener dos valores, ya sean **true** o **false**
- ✓ **Number:** permite representar valores numéricos, 35, -9.25
- ✓ **BigInt:** permite representar valores numéricos que son demasiado grandes para ser representados por el tipo de datos **number**.
- ✓ **String:** representa cadenas de caracteres ("" o "")
- ✓ **Symbol:** representa un valor primitivo único e inmutable.

## 9. Sintaxis del lenguaje - Tipo de datos primitivos



- ✓ **Null:** representa la ausencia intencional de cualquier valor nulo o vacío.
- ✓ **Object:** representa una colección de datos definidos y entidades más complejas.
- ✓ **Function:** es una forma abreviada para funciones. Son objetos con la capacidad de ser ejecutables.

## 9. Sintaxis del lenguaje - Conversiones entre tipos

- **Conversión entre tipos de datos:**

Entero + Float = Float

Número + Cadena = Cadena

- **Conversión de cadenas a números**

`parseInt("32")`

`parseFloat("32.1")`

## 9. Sintaxis del lenguaje - OPERADORES COMPARACIÓN

Sintaxis	Nombre	Tipos de operandos	Resultados
==	Igualdad	Todos	Boolean
!=	Distinto	Todos	Boolean
===	Igualdad estricta	Todos	Boolean
!==	Desigualdad estricta	Todos	Boolean
>	Mayor que	Todos	Boolean
>=	Mayor o igual que	Todos	Boolean
<	Menor que	Todos	Boolean
<=	Menor o igual que	Todos	Boolean

## 9. Sintaxis del lenguaje - OPERADORES ARITMÉTICOS

Sintaxis	Nombre	Tipos de operandos	Resultados
+	Más	Entero, real, cadena	Entero, real, cadena
-	Menos	Entero, real	Entero, real
*	Multiplicación	Entero, real	Entero, real
/	División	Entero, real	Entero, real
%	Módulo	Entero, real	Entero, real
++	Incremento	Entero, real	Entero, real
--	Decremento	Entero, real	Entero, real
+valor	Positivo	Entero, real, cadena	Entero, real
-valor	Negativo	Entero, real, cadena	Entero, real

## 9. Sintaxis del lenguaje - OPERADORES ASIGNACIÓN

Sintaxis	Nombre	Ejemplo	Significado
=	Asignación	x=y	x=y
+=, -=, *=, /=, %=	Operación y asignación	x+=y	x=x+y
<<=	Desplazar bits a la izquierda	x<<=y	x=x<<y
>=, >>=, >>>=	Desplazar bits a la derecha	x>=y	x=x>y
&=	Operación AND bit a bit	x&=y	x=x&y
=	Operación OR bit a bit	x =y	x=x y
^=	Operación XOR bit a bit	x^=y	x=x^y
[]=	Desestructurar asignaciones	[a,b]=[c,d ]	a=c, b=d

## 9. Sintaxis del lenguaje - OPERADORES BOLEANOS

Sintaxis	Nombre	Operandos	Resultados
&&	And	Boolean	Boolean
	Or	Boolean	Boolean
!	Not	Boolean	Boolean

- La operación **AND** solo es true cuando todos los operadores son true.
- La operación **OR** es true siempre que haya un operador true.
- La operación **NOT** cambia el valor del boolean resultado

## 9. Sintaxis del lenguaje - OPERADORES DE OBJETOS

- **Punto:**
  - Objeto.propiedad
  - Objeto.método
- **Corchetes:**
  - Crear un array: `let provincias =[ "Cuenca", "Toledo", "Ciudad Real"]`
  - Enumerar un elemento de un array: `provincias[1]="Guadalajara"`
- **in:**
  - Devuelve true si el objeto tiene la propiedad o método.
- **instanceof:**
  - Devuelve true si es una instancia de un objeto nativo Javascript.  
`aNumeros= new Array(1, 2, 3);`  
`aNumeros instanceof Array; // Devuelve true`

## 9. Sintaxis del lenguaje - OPERADORES MISCELÁNEOS

- **Coma:**

- Expresiones que se evalúan de izquierda a derecha: var nombre, dirección, apellidos
- Operación loop (repetir): for (let i=0, j=0; i<125; i++, j+10)

- **Interrogación (operador condicional):**

- Es la forma reducida de if ... else
- Condición ? expresión si es cierta: expresión si es falso

```
let num1=3, num2=5;
```

```
let resultado= num1>num2 ? num1 = num2; //resultado= 5
```

- **typeof:**

- Devuelve el tipo de valor de una variable o expresión.
- Los tipos son number, string, boolean, object, function, undefined.

```
if (typeof miVariable == "number") alert ("Mi variable es number")
```