

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

# OBJETIVOS

- 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.



# 1.1 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

## 1.2 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.

## 1.3 TECNOLOGÍAS DE PROGRAMACIÓN DEL CLIENTE



## 1.4 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

# 1.4 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.

# 1.4 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



# 1.4 CARACTERÍSTICAS DE JAVASCRIPT

## Limitaciones

No puede modificar o acceder a las preferencias del navegador, ventana principal, impresión

No puede acceder al sistema de ficheros del cliente

No puede capturar datos de un servidor para su retransmisión

No puede enviar e-mail de forma invisible

No puede interactuar directamente con los lenguajes del servidor.

No puede acceder a páginas almacenadas en diferentes dominios.

No puede proteger el origen de las imágenes de la pagina.

## 1.5 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...

## 1.5 HERRAMIENTAS Y UTILIDADES DE PROGRAMACIÓN

### **Navegador Web:**

- ✓ No necesita internet para probar los scripts de JavaScript.
- ✓ Inspector de elementos HTML.
- ✓ Depurador
- ✓ Editor de estilos
- ✓ Otras funciones

### Ejemplos de navegadores

- Google Chrome, Mozilla FireFox, Safari, Opera, Edge, Internet Explores, etc.

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

### Etiquetas <script> en HTML

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

### Navegador no soportado

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

## 1.6 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>`

## 1.6 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

## 1.6 PROTECCIÓN DE CÓDIGO JAVASCRIPT

**El código en Javascript no puede protegerse: está accesible y visible a través de un navegador.**



Incluir mensaje de Copyright

Ofuscar el código  
<https://javascript2img.com/>  
<https://seostudio.tools/es/javascript-obfuscator>

Promocionar el código

## 1.7 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 2024 (**ES2024**).

Documentación oficial → <https://262.ecma-international.org/>

Resumen → [https://www.w3schools.com/js/js\\_2024.asp](https://www.w3schools.com/js/js_2024.asp)



## 1.8 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 (F12 en Chrome) .
  - `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);
```

## 1.9 Manejo de la 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 */
```

## 1.9 Manejo de la Sintaxis del lenguaje – Variables-

### Variables

- Permiten almacenar distintos valores en cada momento

#### **Declaración de variables:**

- **let:** La variable es accesible únicamente en el bloque (block scoped) donde se ha declarado,
- **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.
- **const:** Declara variables locales dentro del bloque y su valor no puede cambiar (constantes).
- **Variables sin declarar:** JavaScript permite usar variables no declaradas, es como si se declararan con **var**

## 1.9 Manejo de la 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 name="Alba prieto";
```

## 1.9 Manejo de la 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
- Utilizar nombres autodescriptivos

## 1.9 Manejo de la Sintaxis del lenguaje - Modo estricto “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. Esto implica, entre otras cosas, que es obligatorio declarar las variables antes de su utilización.

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

## 1.9 Manejo de la Sintaxis del lenguaje - Tipo de datos -

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

## 1.9 Manejo de la 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**. Fundamentales en las estructuras condicionales
- ✓ **Number:** permite representar valores numéricos, 35, -9.25
- ✓ **BigInt:** permite representar valores numéricos enteros que son demasiado grandes para ser representados por el tipo de datos **number**. Para definir un valor como BigInt se agrega una **n** después del valor (let sumaBigInt=1000n+2000n). Si no se añade la **n** se considera Number.
- ✓ **String:** representa cadenas de caracteres (“” o ‘’)
- ✓ **Symbol:** representa un valor primitivo único e inmutable. Se usa para crear identificadores únicos que no pueden ser duplicados evitando así la colisión de nombres en proyectos muy grandes o librerías compartidas.



## 1.9 Manejo de la Sintaxis del lenguaje – Otros tipos de datos-

- ✓ **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.

## 1.9 Manejo de la Sintaxis del lenguaje – Conversiones entre tipos -

- **Conversión entre tipos de datos:**

- ✓ Entero + Float = Float (Al almacenarse en un Number, da igual que sea entero o float)
- ✓ Número + Cadena = Cadena (Cualquier valor sumado a una cadena se convierte en una cadena)

- **Conversión de cadenas a números**

- ✓ `parseInt("32")`
- ✓ `parseFloat("32.1")`

- **Conversión de números a cadenas:**

- ✓ `"32" + 5 // "325"`

## 1.9 Manejo de la Sintaxis del lenguaje – Operadores de 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

## 1.9 Manejo de la 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

## 1.9 Manejo de la Sintaxis del lenguaje – Operadores de 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

## 1.9 Manejo de la Sintaxis del lenguaje – Operadores booleanos -

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

## 1.9 Manejo de la 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
    - ❖ `"write" in document` // devuelve true si write es una propiedad o un método de document
- **instanceof:**
  - ✓ Devuelve true si es una instancia de una clase (nativa de JS o definido por el usuario)
    - ❖ `aNumeros= new Array(1, 2, 3);`  
`aNumeros instanceof Array; // Devuelve true`

## 1.9 Manejo de la 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")
```



## 1.9 Manejo de la Sintaxis del lenguaje – Estructuras de control-

- Instrucciones if/else

```
if (condición) {  
  // bloque de instrucciones que se ejecutan si la condición se  
}  
else{  
  // bloque de instrucciones que se ejecutan si la condición no  
}
```

```
let diaSem;  
diaSem=prompt("Introduce el día de la semana ", "");  
if (diaSem === "domingo")  
{  
  console.log("Hoy es festivo");  
}  
else // Al no tener {}, es un "bloque de una instrucción"  
  console.log("Hoy no es domingo, a descansar!!");
```

## 1.9 Manejo de la Sintaxis del lenguaje - Estructuras de control-

- Instrucciones if/else

```
let edadAna,edadLuis;  
// Convertirmos a entero las cadenas  
edadAna=parseInt(prompt("Introduce la edad de Ana",""));  
edadLuis=parseInt(prompt("Introduce la edad de Luis",""));  
if (edadAna > edadLuis){  
    console.log("Ana es mayor que Luis.");  
}  
else{  
    if (edadAna<edadLuis){  
        console.log("Ana es menor que Luis.");  
    }else{  
        console.log("Ana tiene la misma edad que Luis.");  
    }  
}  
console.log(" Ana tiene "+edadAna+" años y Luis "+ edadLuis);
```

## 1.9 Manejo de la Sintaxis del lenguaje – Estructuras de control-

- **Instrucciones switch**

```
switch (variable) {  
  valor1: // Instrucciones que se van a realizar  
          break; // Rompemos para no ejecutar el resto  
  valor2: // Instrucciones que se van a realizar  
          break;  
  ..... // Más comprobaciones  
  valorn: // Instrucciones a realizar  
          break;  
  default: // Que se hacen en le resto de los casos  
           // No hay break ya que es la última.  
}
```

## 1.9 Manejo de la Sintaxis del lenguaje - Estructuras de control-

- Instrucciones switch

```
let dato = 4;
switch (dato) {
  case 1:
    console.log("el dato es uno");
    break;
  case 2:
    console.log("el dato es dos");
    break;
  case 3:
    console.log("el dato es tres");
    break;
  case 4:
    console.log("el dato es cuatro");
    break;
  default:
    console.log("el dato es diferente");
    break;
}
```

## 1.9 Manejo de la Sintaxis del lenguaje – Estructuras de control-

- Instrucciones switch

```
let dato = 10;
switch (true) {
  case dato < 5:
    console.log("el dato es menor de 5");
    break;
  case dato == 5:
    console.log("el dato es = 5");
    break;
  case dato > 5 && dato <= 10:
    console.log("el dato es mayor de 5 e igual o menor de 10");
    break;
  default:
    break;
}
```

## 1.9 Manejo de la Sintaxis del lenguaje - Bucles -

- Instrucciones for

```
for (expresion inicial; condicion; incremento)
{
    // instrucciones a ejecutar dentro del bucle
}
```

- Ejemplo:

```
for (var i=1; i<20; i++)
{
    //Instrucciones que se repetirán 20 veces
}
```

## 1.9 Manejo de la Sintaxis del lenguaje - Bucles -

- Instrucciones while

```
while(condicion)
{
    // instrucciones a ejecutar dentro del bucle
}
```

- Ejemplo:

```
var i=0;
while (i <=10)
{
    //Instrucciones a ejecutar hasta que i sea mayor que 10 y
    i++;
}
```

## 1.9 Manejo de la Sintaxis del lenguaje - Bucles -

- Instrucciones do... while

```
do {  
    // instrucciones a ejecutar dentro del bucle  
} while (condicion);
```

- Ejemplo:

```
var i=0;  
do {  
    //Instrucciones a ejecutar mientras i sea menor que 3 (2 veces)  
    i++;  
} while (i<3)
```