

## 2.- Programación web. Tecnologías de programación. Fundamentos de JAVASCRIPT

# OBJETIVOS

- Comprender los modelos de ejecución en entornos web
- Conocer las capacidades de los navegadores web.
- Familiarizarse con los lenguajes de programación para clientes web.
- Reconocer las particularidades de la programación de guiones.
- Integrar lenguajes de Marcas y lenguajes de Programación.
- Evaluar herramientas de programación para clientes web
- 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.

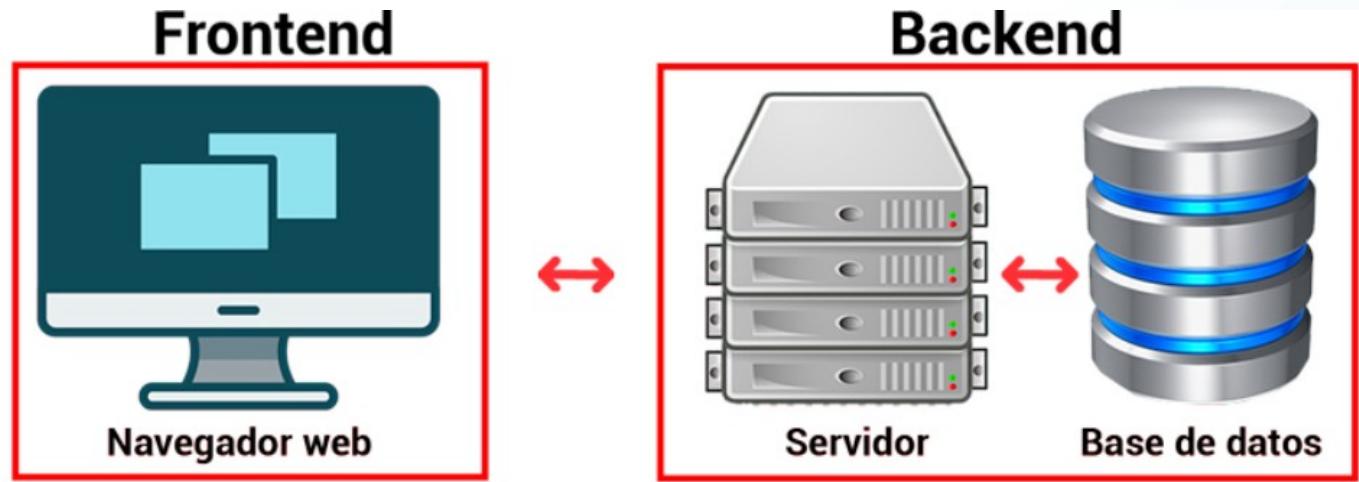


# 1. Introducción

En el desarrollo de aplicaciones web, es fundamental seleccionar las arquitecturas y tecnologías adecuadas para garantizar una experiencia de usuario óptima y eficiente. Este proceso implica identificar y analizar las capacidades de las tecnologías disponibles tanto en el lado del cliente como en el servidor. Los navegadores web modernos ofrecen diversas capacidades y mecanismos de ejecución de código, permitiendo la creación de aplicaciones dinámicas e interactivas.

El aprendizaje de los lenguajes de programación específicos para el desarrollo web, como HTML, CSS y JavaScript, es crucial para aprovechar al máximo estas capacidades. Además, es importante entender las particularidades de la programación de guiones en comparación con la programación tradicional, así como los mecanismos de integración entre los diferentes lenguajes y herramientas.

## 2. Identificación y diferenciación de modelos de ejecución.



## 2. Identificación y diferenciación de modelos de ejecución.

### ● Código en el Servidor (BackEnd)

- ❑ El código se procesa y ejecuta en un servidor antes de que la información se envíe al cliente (navegador web).
- ❑ Aquí se realiza gran parte del trabajo pesado, como la generación de contenido dinámico, acceso a la BD y la aplicación de lógica de negocio.
- ❑ Una vez que el servidor ha generado la respuesta, esta se envía al navegador del usuario.
- ❑ Ejemplos:
  - PHP
  - Node.js
  - Python
  - Ruby
  - ASP.NET

## 2. Identificación y diferenciación de modelos de ejecución.

- **Código en el Cliente (FrontEnd).**

- ❑ El código se ejecuta directamente en el navegador del usuario, después de que la página ha sido cargada.
- ❑ Este modelo permite la interacción directa y dinámica con el contenido web sin necesidad de recargar la página.
- ❑ El navegador interpreta y ejecuta el código que normalmente está escrito en JavaScript y puede manipular el Document Object Model (DOM) para actualizar el contenido de la página en respuesta a las acciones del usuario.
- ❑ Ejemplos:
  - JavaScript
  - React
  - Angular
  - Vue.js

## 2. Identificación y diferenciación de modelos de ejecución.

### ● Análisis de los modelos de ejecución en el Servidor y en el Cliente

#### □ Modelo de ejecución en el Servidor

##### ➤ Ventajas:

- ✓ **Seguridad:** La lógica de negocio y acceso a datos sensibles se manejan en el Servidor, lo que reduce el riesgo de exposición de información sensible al usuario final.

##### ➤ Desventajas:

- ✓ **Carga en el Servidor:** Todas las solicitudes del cliente requieren procesamiento en el Servidor, lo que puede llevar a una mayor carga y necesidad de recursos más potentes en el servidor.
- ✓ **Tiempo de respuesta:** La comunicación entre el cliente y el servidor puede aumentar el tiempo de respuesta, especialmente si la conexión a Internet es lenta o el servidor está muy ocupado.

## 2. Identificación y diferenciación de modelos de ejecución.

### ● Análisis de los modelos de ejecución en el Servidor y en el Cliente

#### □ Modelo de ejecución en el Cliente

##### ➤ Ventajas:

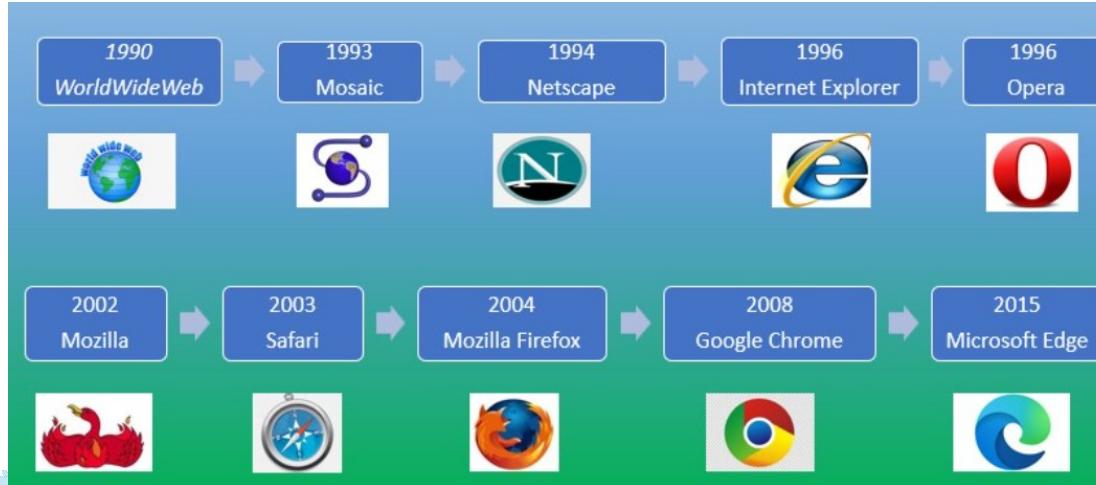
- ✓ **Interactividad:** Permite una experiencia de usuario más dinámica y rápida, ya que el navegador puede actualizar el contenido sin necesidad de recargar de página.
- ✓ **Descarga la carga del Servidor:** Al trasladar parte del procesamiento al cliente, se reduce la carga en el Servidor, lo que puede mejorar el rendimiento.

##### ➤ Desventajas:

- ✓ **Seguridad:** El código y la lógica ejecutados en el Cliente pueden ser más vulnerables a manipulaciones y ataques, ya que el usuario tiene acceso directo al código que se ejecuta en el navegador.
- ✓ **Compatibilidad:** Las diferencias en los navegadores y dispositivos pueden causar problemas de compatibilidad, lo que puede llevar a inconsistencias en el funcionamiento de la aplicación.

### 3. Capacidades de los navegadores Web

#### ● Historia de los navegadores



### 3. Capacidades de los navegadores Web

#### ● Exploración de Mecanismos de ejecución de Código

- ❑ Los navegadores no solo muestran contenido HTML, también interpretan y ejecutan código, lo que permite una amplia gama de interacciones y funcionalidades avanzadas.
- ❑ Los navegadores utilizan motores de JavaScript que interpretan y ejecutan el código JavaScript:

##### ➤ V8

- ✓ Desarrollado por Google y utilizado en el navegador Chrome y en el entorno de ejecución Node.js

##### ➤ SpiderMonkey

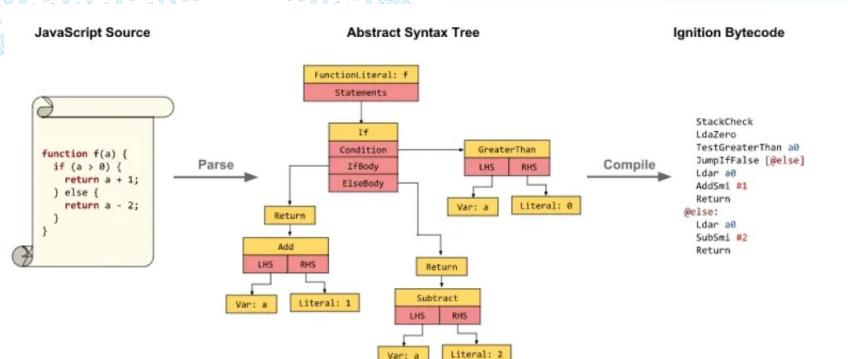
- ✓ Es utilizado por el navegador Firefox, desarrollado por Mozilla.

# 3. Capacidades de los navegadores Web

## ● Exploración de Mecanismos de ejecución de Código

### □ Proceso de interpretación y ejecución de los motores:

- **Lectura y análisis.** El motor de JavaScript lee el código fuente y lo analizan sintácticamente generando un árbol de sintaxis abstracta (AST)
- **Compilación JIT (Just In Time Compiler).** El código JavaScript se compila a un formato intermedio y luego a código máquina nativo, mejorando la velocidad.
- **Ejecución.** El código máquina nativo se ejecuta directamente en el hardware del dispositivo proporcionando una experiencia rápida y eficiente.



### 3. Capacidades de los navegadores Web

- Exploración de Mecanismos de ejecución de Código

- Capacidades del Navegador:

- Manipulación del DOM y manejo de eventos.
    - Comunicación asíncrona (AJAX)
    - Almacenamiento local (LocalStorage y SessionStorage)

### 3. Capacidades de los navegadores Web

#### ● Compatibilidad y Estándares

- ❑ Los navegadores siguen estándares definidos por el W3C para asegurar la compatibilidad y el correcto funcionamiento de las tecnologías web.
- ❑ Hace algunos años, el desarrollo de páginas web era caótico, había que realizar una versión de cada página web para prácticamente cada navegador.
- ❑ Para dar solución a este problema, el **W3C** (World Wide Web Consortium), lanzó una iniciativa en 1997 para lograr la accesibilidad web y que siguieran unas mismas pautas



### 3. Capacidades de los navegadores Web

#### ● Compatibilidad y Estándares

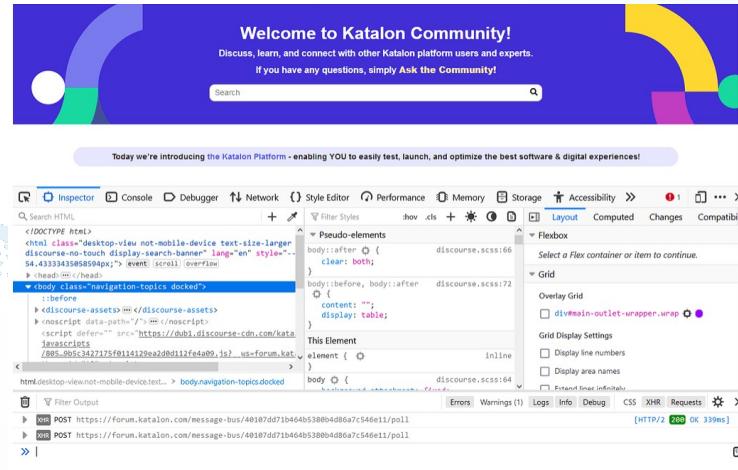
- A raíz de entonces se fueron desarrollando diferentes estándares logrando que cada página se vea correctamente independientemente del navegador o dispositivo
- Algunos estándares Web más conocidos y utilizados son:
  - **HTML**, para definir la estructura de los documentos.
  - **XML**, que sirve de base para un gran número de tecnologías.
  - **CSS**, permite asignar estilos para la representación de documentos.
  - **JavaScript**, permite otorgar dinamismo y funcionalidad.



# 3. Capacidades de los navegadores Web

## ● Herramientas de desarrollo

- ❑ Las herramientas de desarrollo (DevTools) integradas en los navegadores son esenciales para depurar, analizar y optimizar el código.
- ❑ Las principales funcionalidades son:
  - Consola JavaScript
  - Inspector de Elementos
  - Perfilador de Rendimiento
  - Depurador



## 4. 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...

# 5. JavaScript

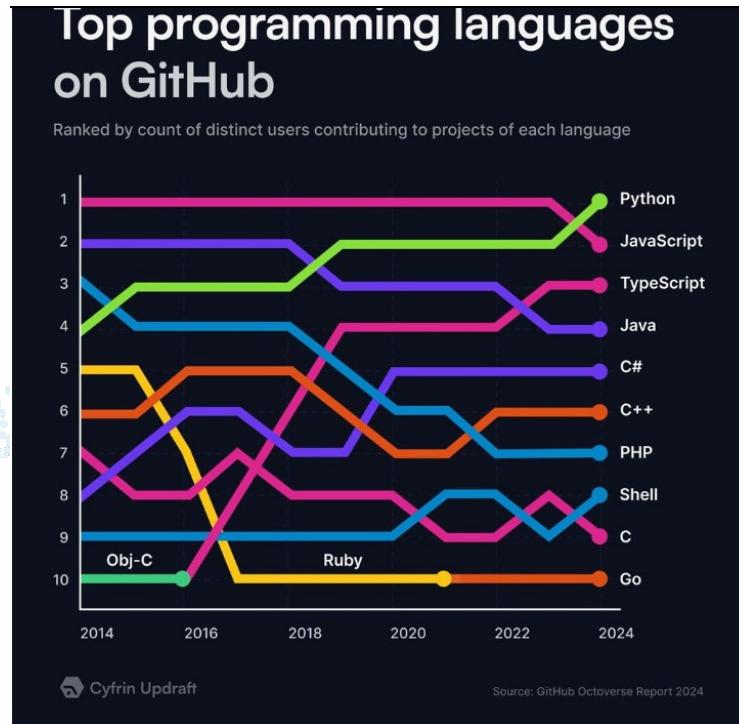
## ● Definición

- ❑ Es un lenguaje de programación interpretado.
- ❑ Se utiliza principalmente para el desarrollo web.
- ❑ Permite agregar interactividad y dinamismo a las páginas web



# 5. JavaScript

## ● Comparativa de JavaScript con otros lenguajes (2024)



# 5. JavaScript

## ● Historia

- ❑ Java y JavaScript son dos lenguajes totalmente diferentes.
- ❑ Creado en 1995 por Brendan Eich en Nestcape en 10 días
- ❑ Originalmente fue llamado LiveScript y finalmente JavaScript.



# 5. JavaScript

## ● Historia

- Estándar **EcmaScript** desarrollado por ECMA Internacional:
  - 1997 (**ES1**) estableciendo una base común para las implementaciones del lenguaje
  - 2009 (**ES5**) introduciendo nuevas características como “strict mode”, JSON nativo y manejo de mejoras para los arrays.
  - 2015 (**ES6**) estableciendo novedades más significativas del lenguaje como nuevas clases, módulos, let y const, funciones arrows...
  - 2025 (**ES16**), última versión publicada en junio de 2025. Nuevo objeto (Iterator). Métodos avanzados en Set. Importación de JSON como módulo nativo, Promise.try(). RegExpt.escape()



# 5. JavaScript

## ● Características principales

- ❑ Es un lenguaje interpretado:
  - Se ejecuta en el navegador sin necesidad de compilación previa.
- ❑ Basado en prototipos:
  - Utiliza prototipos en lugar de clases por herencia.
- ❑ Event-Driven:
  - Responde a eventos como clics, teclas y movimientos de ratón



# 5. JavaScript

## ● Aplicaciones

- Desarrollo Web:

- Crear sitios web interactivos

- Aplicaciones de Una Sola Página (SPA):

- React, Angular, Vue.js

- Desarrollo del lado del Servidor:

- Node.js permite ejecutar JavaScript en el servidor.

- Aplicaciones móviles:

- Framework como React Native, Ionic, NativeScript

## 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>
```

## 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>
```

## 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

## 7. 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://obfuscator.io/>

Promocionar el código

## 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. 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 linea
```

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

## 9. Manejo de la Sintaxis del lenguaje - Variables-

### Variables

- Es un contenedor de información que apunta a un lugar en memoria. Dicha información puede cambiar en el futuro.
- JavaScript es un lenguaje débilmente tipado,

### 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.
- Variables sin declarar:** JavaScript permite usar variables no declaradas, es como si se declararan con **var**

# 9. Manejo de la Sintaxis del lenguaje - Variables-

## Crear variables

```
var edad; //En desuso  
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";
```

## 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

## 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, es decir, que es obligatorio declarar las variables antes de su utilización.

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

## 9. Manejo de la Sintaxis del lenguaje - Tipo de datos -

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

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

- ✓ **Undefined**: representa una variable que no se le ha asignado un valor ni se ha declarada.
- ✓ **Boolean**: representa un valor lógico y puede tener dos valores, **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. 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.

## 9. Manejo de la 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")`
- **Conversión de números a cadenas:**
  - ✓ `"32"+ 5 // "325"`

## 9. Manejo de la Sintaxis del lenguaje - Operadores de comparación-

Sintaxis	Nombre	Tipos de operandos	Resultados
<code>==</code>	Igualdad	Todos	Boolean
<code>!=</code>	Distinto	Todos	Boolean
<code>==&gt;</code>	Igualdad estricta	Todos	Boolean
<code>!=&gt;</code>	Desigualdad estricta	Todos	Boolean
<code>&gt;</code>	Mayor que	Todos	Boolean
<code>&gt;=</code>	Mayor o igual que	Todos	Boolean
<code>&lt;</code>	Menor que	Todos	Boolean
<code>&lt;=</code>	Menor o igual que	Todos	Boolean

## 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

## 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	$[a,b]=[c,d]$ ]	$a=c, b=d$

## 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

## 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
- **instanceof:**
  - ✓ Devuelve true si es una instancia de un objeto nativo Javascript.
    - ❖ aNumeros= new Array(1, 2, 3);  
aNumeros instanceof Array; // Devuelve true

## 9 . Manejo de la Sintaxis del lenguaje- Operadores misceláneos -

- **Coma:**
  - ✓ Expresiones que se evalúan de izquierda a derecha: **let** nombre, dirección, apellidos
  - ✓ Operación loop (repetir): **for** (**let** i=0, j=0; i<125; i++, j+10)
- **? (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 : num1; //resultado= 5
```

- **?? (Nullish Coalescing):**
  - ✓ Devuelve el operando de la derecha solo si el operando de la izquierda es *null* o *undefined*

```
let usuarioEdad;  
let valor = usuarioEdad ?? 18; //valor=18
```

## 9 . Manejo de la Sintaxis del lenguaje- Operadores misceláneos -

- **? (Optional Chaining):**

- ✓ Permite acceder a propiedades anidadas de un objeto sin que se genere un error si alguna parte de la cadena es null o undefined

```
let usuario ={  
    nombre: "Ana",  
};  
  
console.log(usuario.nombre); // "Ana";  
  
console.log(usuario?.edad); //undefined, no da error
```

- **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")
```

## 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!!");
```

## 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);
```

## 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.  
}
```

## 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;
}
```

## 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;
}
```

## 9. Manejo de la Sintaxis del lenguaje- Bucles -

- **Instrucciones for**

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

- Ejemplo:

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

## 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++;
}
```

## 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)
```