

JavaScript: El Motor de la Web Dinámica

Aprende a crear experiencias web interactivas y dinámicas con el lenguaje de programación más popular del mundo.

```
// /  
cIysvecsoprIΔCf  
Iowon') {}  
bspwes posirE{};  
reivade sove' (;  
  rnit fhe L"xxghe(sc");  
  }  
eiresy,  
  ciatomu deE /ny/'{  
    badenhe seumiatifX());  
    jxpheatzon();  
    este'tebecenett());  
    podiveat,rua{}{.  
}  
  moae·fteeai(�tēfndə,):  
  vosLemurbenboñ);  
  msstounine ete(an'y);  
}
```

¿Qué es JavaScript?

JavaScript es un **lenguaje de programación interpretado** que se ejecuta en tiempo real en el navegador del usuario. A diferencia de otros lenguajes compilados, JavaScript permite modificar el contenido de una página web de forma dinámica sin necesidad de recargarla.

Su papel principal es añadir **interactividad y lógica** a las páginas web, transformándolas de documentos estáticos en aplicaciones funcionales completas.

Características clave

- Lenguaje interpretado en tiempo real
- Ejecución en el cliente (navegador)
- Tipo dinámico y flexible
- Compatible con todos los navegadores modernos

El Triángulo del Desarrollo Web



HTML

Estructura y contenido del documento



CSS

Estilos, diseño y aspecto visual



JavaScript

Lógica, interactividad y comportamiento

JavaScript es el elemento dinámico que une HTML (estructura) y CSS (estilos) para crear experiencias web completas. Permite responder a las acciones del usuario, modificar el DOM en tiempo real y comunicarse con servidores de forma asíncrona.



La Historia de JavaScript

1995

Netscape crea Mocha (luego LiveScript), renombrado JavaScript para aprovechar popularidad de Java

2009

Node.js permite ejecutar JavaScript fuera del navegador, en servidores

1

2

3

4

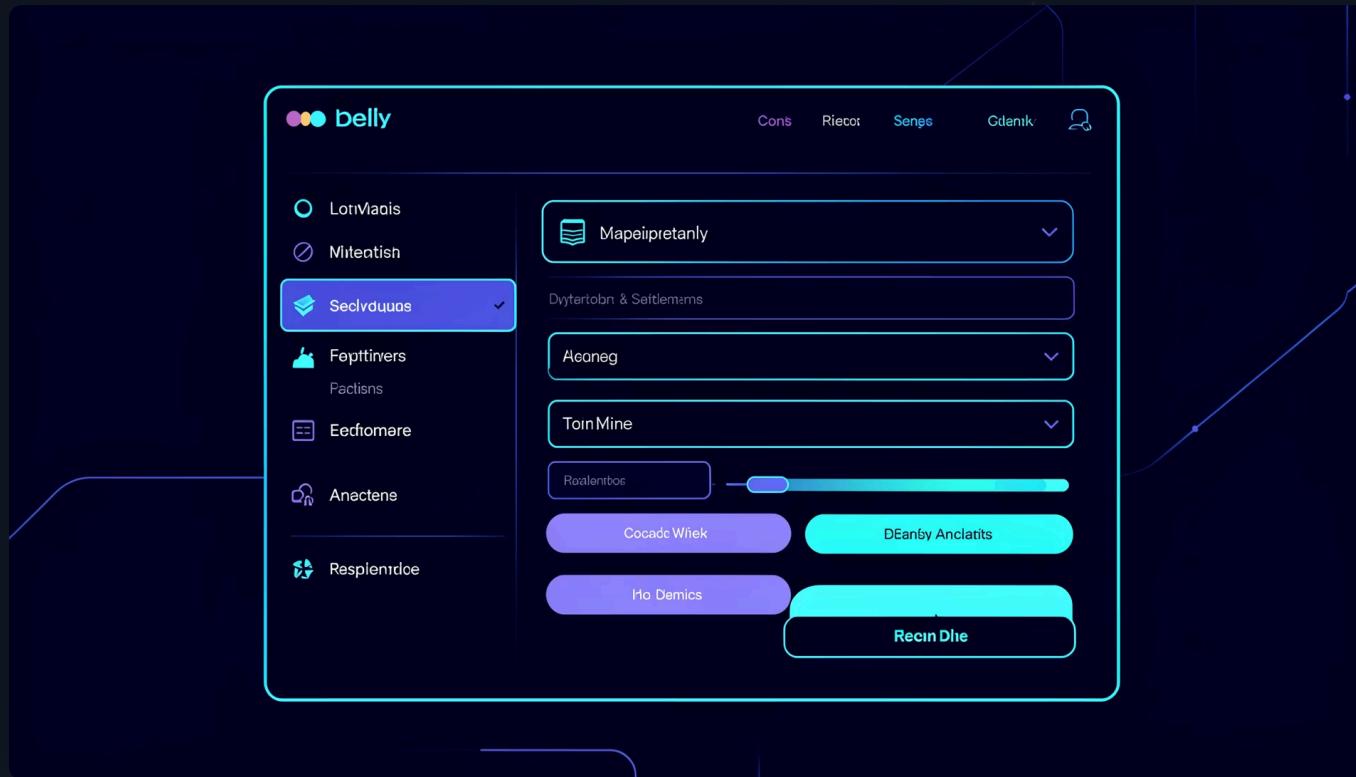
1997

ECMAScript 1 se estandariza para garantizar compatibilidad entre navegadores diferentes

2015+

ECMAScript 6+ introduce modernas características: clases, módulos, promesas y sintaxis avanzada

¿Para Qué Sirve JavaScript en la Práctica?



Comercio electrónico

Actualización del carrito de compras sin recargar página



Redes sociales

Feed infinito, notificaciones instantáneas y mensajes en vivo

Casos de uso cotidiano

- Validación de formularios en tiempo real
- Menús desplegables y navegación dinámica
- Carruseles de imágenes y sliders
- Contadores y relojes actualizados automáticamente
- Carga de contenido sin recargar la página
- API de geolocalización y cámara



Streaming multimedia

Controles de reproducción y gestión de contenido

Fundamentos: Variables y Tipos de Datos

1

Declaración de variables

JavaScript ofrece tres formas: var (función), let (bloque) y const (constante, bloque)

```
let mensaje = "Bienvenido";
const PI = 3.1416;
var usuario = "desarrollador";
```

2

Tipos de datos principales

String, Number, Boolean, Null, Undefined, Symbol y Object.
JavaScript es dinámico: el tipo se determina en ejecución

```
let nombre = "JavaScript";
let edad = 28;
let activo = true;
let datos = null;
```

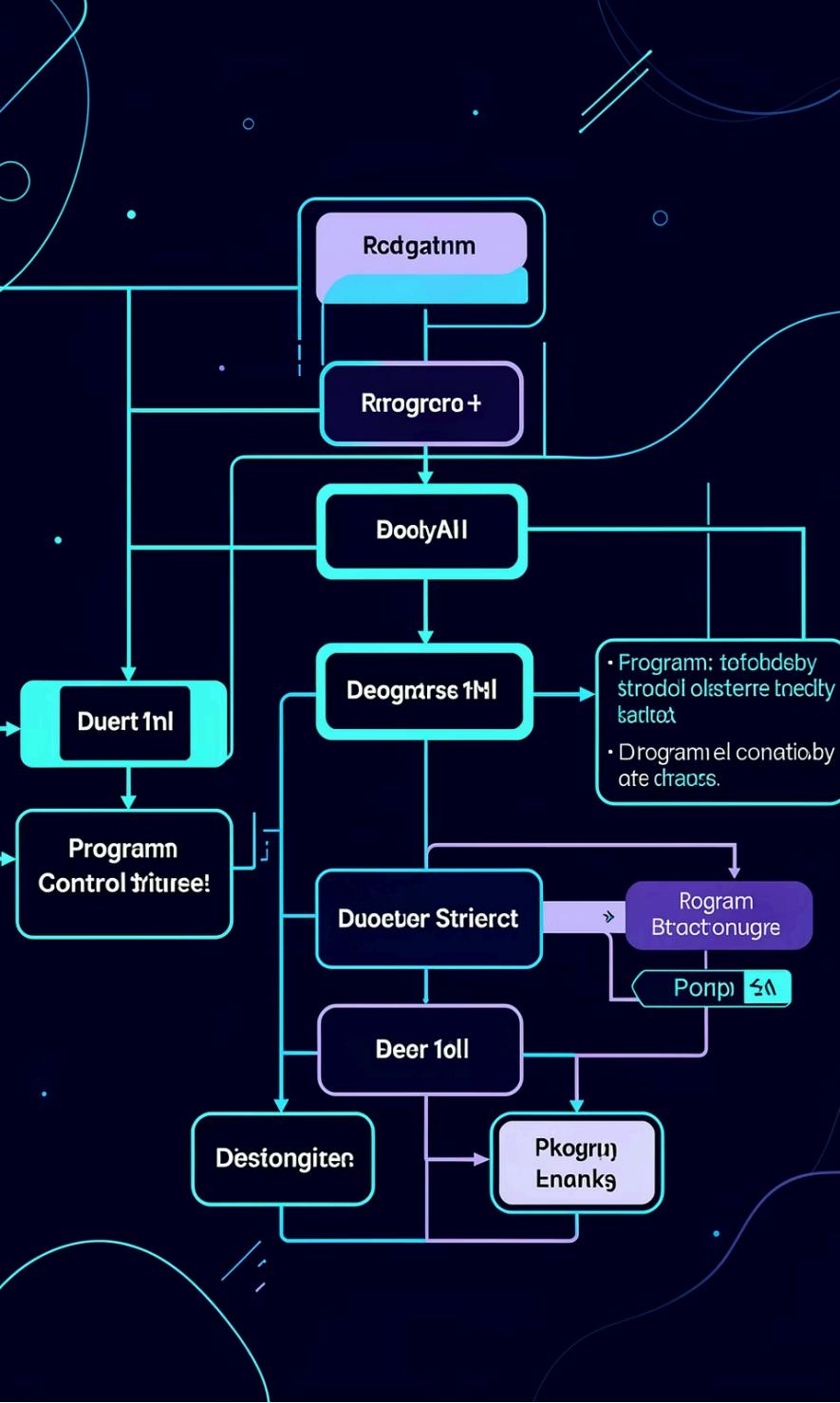
3

Operadores básicos

Aritméticos (+, -, *, /), asignación (=), comparación (==, ===, !=, >, <) y lógicos (&&, ||, !)

```
let suma = 5 + 3;
let mayor = (edad > 18);
let valido = (nombre && edad);
```

Estructuras de Control



Condicionales

Controlan la ejecución basándose en condiciones lógicas

```
if (edad >= 18) {  
    console.log("Mayor de edad");  
} else {  
    console.log("Menor de edad");  
}  
  
switch(dia) {  
    case "lunes":  
        console.log("Día laborable");  
        break;  
    default:  
        console.log("Fin de semana");  
}
```

Bucles

Repite bloques de código múltiples veces

```
for (let i = 0; i < 5; i++) {  
    console.log("Iteración: " + i);  
}  
  
while (activo) {  
    console.log("Ejecutando...");  
    activo = false;  
}  
  
let numeros = [1, 2, 3];  
numeros.forEach(num => {  
    console.log(num * 2);  
});
```

```
javasspniff {
    faysrhct ecaen=U10);
    instfrstnff {
        Javerhertsf {
            fivesnoeh' dvifeal(, 01("t)),
            fmnhtfnçalld ( 08"e)
                (ineaørerv((\eg),
            }
        orserinæun intaver);
        ercalix' and(uøθ:(Dx,-1()));
        siaveer6A1exex());
        >»»
        fctrtaes teed&Z"();
        srttrveølnenf((*1(|));
        time
        fcvsohoøvolcie(()øøarhin), :
        fistineni' e,
        €0vm:
            = yartofeanecrd);
            = \leoeninøsaødr((nø""";="));
            dty"(bph‡2ð(), §),
        }
    }
}
```

Manipulación del DOM y Eventos



Acceso al DOM

JavaScript puede seleccionar elementos HTML usando métodos como getElementById, querySelector o getElementsByTagName

Escucha de eventos

Se registran "listeners" para responder a acciones del usuario: click, submit, change, keypress, scroll y muchos más tipos de eventos

Modificación dinámica

Se altera el contenido, estilos o estructura del documento en tiempo real sin recargar la página, creando experiencias fluidas

Ejemplo Práctico: Contador Interactivo

HTML + JavaScript integrados

Vamos a crear un contador simple que responda a clics de botones:

```
<html>
<body>
  <h2>Contador: <span id="contador">0</span></h2>

  <button onclick="incrementar()"></span>>
    Sumar
  </button>

  <button onclick="restar()"></span>>
    Restar
  </button>

  <script>
    let contador = 0;

    function incrementar() {
      contador++;
      document.getElementById("contador").textContent =
      contador;
    }

    function restar() {
      contador--;
      document.getElementById("contador").textContent =
      contador;
    }
  </script>
</body>
</html>
```

¿Cómo funciona?

1. Inicializamos variable contador en 0
2. Función incrementar() suma 1 al contador
3. Función restar() resta 1 al contador
4. Método getElementById actualiza el contenido del DOM
5. textContent modifica el texto mostrado

Buenas Prácticas y Errores Comunes

✨ Mejores prácticas

- Usar `let` y `const` en lugar de `var`
- Declarar variables con su alcance correcto (block scope)
- Usar `==` para comparaciones estrictas
- Separar lógica en funciones reutilizables
- Añadir comentarios claros en código complejo
- Usar `console.log()` para depuración

⚠️ Errores típicos

- **Hoisting no deseado:** declarar variables con `var` en lugar de `let/const`
- **Comparación floja:** usar `==` en lugar de `===`
- **Sin declaración:** crear variables globales implícitas
- **Redefinición:** intentar reasignar a variables `const`
- **Sin punto y coma:** omitirlos puede causar problemas
- **Undefined:** intentar acceder a propiedades de valores no definidos

Recuerda: La práctica constante y la lectura de documentación son esenciales para dominar JavaScript. Prueba tu código frecuentemente en diferentes navegadores y usa herramientas de desarrollo.