13 de junio de 2022

Especiales JavaScript

Este capítulo resume brevemente las características de JavaScript que hemos aprendido hasta ahora, prestando especial atención a los detalles relevantes.

Estructura de Código

Las declaraciones se delimitan con un punto y coma:

```
alert('Hola'); alert('Mundo');
```

En general, un salto de línea también se trata como un delimitador, por lo que también funciona:

```
alert('Hola')
alert('Mundo')
```

Esto se llama "inserción automática de punto y coma". A veces no funciona, por ejemplo:

```
alert("Habrá un error después de este mensaje.")

[1, 2].forEach(alert)
```

La mayoría de las guías de estilo de código coinciden en que debemos poner un punto y coma después de cada declaración.

Los puntos y comas no son necesarios después de los bloques de código `{...}` y los constructores de sintaxis como los bucles:

```
function f() {
  // no se necesita punto y coma después de la declaración de función
}

for(;;) {
  // no se necesita punto y coma después del bucle
}
```

...Pero incluso si colocásemos un punto y coma "extra" en alguna parte, eso no sería un error. Solo sería ignorado.

Más en: Estructura del código.

Modo estricto

Para habilitar completamente todas las características de JavaScript moderno, debemos comenzar los scripts con `"use strict"`.

```
'use strict';

...
```

La directiva debe estar en la parte superior de un script o al comienzo de una función.

Sin la directiva `"use strict"` todo sigue funcionando, pero algunas características se comportan de la manera antigua y "compatible". Generalmente preferimos el comportamiento moderno.

Algunas características modernas del lenguaje (como las clases que estudiaremos en el futuro) activan el modo estricto implícitamente.

Más en: El modo moderno, "use strict".

Variables

Se pueden declarar usando:

- `let`
- `const` (constante, no se puede cambiar)
- `var` (estilo antiguo, lo veremos más tarde)

Un nombre de variable puede incluir:

- Letras y dígitos, pero el primer carácter no puede ser un dígito.
- Los caracteres `$` y `_` son normales, al igual que las letras.
- Los alfabetos y jeroglíficos no latinos también están permitidos, pero comúnmente no se usan.

Las variables se escriben dinámicamente. Pueden almacenar cualquier valor:

```
let x = 5;
x = "John";
```

Hay 8 tipos de datos:

- `number` tanto para números de punto flotante como enteros,
- `bigint` para números enteros de largo arbitrario,

- `string` para textos,
- `boolean` para valores lógicos: `true/false`,
- `null` – un tipo con el valor único `null`, que significa "vacío" o "no existe",
- `undefined` – un tipo con el valor único `undefined`, que significa "no asignado",
- `object` y `symbol` – para estructuras de datos complejas e identificadores únicos, aún no los hemos aprendido.

El operador `typeof` devuelve el tipo de un valor, con dos excepciones:

```
typeof null == "object" // error del lenguaje
typeof function(){} == "function" // las funciones son tratadas especialment
```

Más en: Variables y Tipos de datos.

Interacción

Estamos utilizando un navegador como entorno de trabajo, por lo que las funciones básicas de la interfaz de usuario serán:

`prompt(question, [default])`

Hace una pregunta `question`, y devuelve lo que ingresó el visitante o `null` si presiona "cancelar".

`confirm(question)`

Hace una pregunta `question`, y sugiere elegir entre Aceptar y Cancelar. La elección se devuelve como booleano `true/false`.

`alert(message)`

Muestra un `message`.

Todas estas funciones son *modales*, pausan la ejecución del código y evitan que el visitante interactúe con la página hasta que responda.

Por ejemplo:

```
let userName = prompt("¿Su nombre?", "Alice");
let isTeaWanted = confirm("¿Quiere té?");

alert( "Visitante: " + userName ); // Alice
alert( "Quiere té: " + isTeaWanted ); // true
```

Más en: Interacción: alert, prompt, confirm.

Operadores

JavaScript soporta los siguientes operadores:

Aritméticos

Los normales: `* + - /`, también `%` para los restos y `**` para aplicar potencia de un número.

El binario más `+` concatena textos. Si uno de los operandos es un texto, el otro también se convierte en texto:

```
1  alert( '1' + 2 ); // '12', texto
2  alert( 1 + '2' ); // '12', texto
```

Asignaciones

Existen las asignaciones simples: `a = b` y las combinadas `a *= 2`.

Operador bit a bit

Los operadores bit a bit funcionan con enteros de 32 bits al más bajo nivel, el de bit: vea la documentación cuando los necesite.

Condicional

El único operador con 3 parámetros: `cond ? resultA : resultB`. Sí `cond` es verdadera, devuelve `resultA`, de lo contrario `resultB`.

Operadores Lógicos

Los operadores lógicos Y `&&` y Ó `||` realizan una evaluación de circuito corto y luego devuelven el valor donde se detuvo (no necesariamente true/false). El operador lógico NOT `!` convierte el operando a tipo booleano y devuelve el valor inverso.

Operador "Nullish coalescing"

El operador `??` brinda una forma de elegir el primer valor "definido" de una lista de variables. El resultado de `a ?? b` es `a` salvo que esta sea `null/undefined`, en cuyo caso será `b`.

Comparaciones

Para verificar la igualdad `==` de valores de diferentes tipos, estos se convierten a número (excepto `null` y `undefined` que son iguales entre sí y nada más), por lo que son iguales:

```
1  alert( 0 == false ); // true
2  alert( 0 == '' ); // true
```

Otras comparaciones también se convierten en un número.

El operador de igualdad estricta `===` no realiza la conversión: diferentes tipos siempre significan diferentes valores.

Los valores `null` y `undefined` son especiales: son iguales `==` el uno al otro y no son iguales a nada más.

Las comparaciones mayor/menor comparan las cadenas carácter por carácter, los demás tipos de datos se convierten a número.

Otros operadores

Hay algunos otros, como un operador de coma.

Más en: Operadores básicos, matemáticas, Comparaciones, Operadores Lógicos, Operador Nullish Coalescing '??'.

Bucles

- Cubrimos 3 tipos de bucles:

```
// 1
while (condition) {
  ...
}

// 2
do {
  ...
} while (condition);

// 3
for(let i = 0; i < 10; i++) {
  ...
}
```

- La variable declarada en el bucle `for(let...)` sólo es visible dentro del bucle. Pero también podemos omitir el `let` y reutilizar una variable existente.
- Directivas `break/continue` permiten salir de todo el ciclo/iteración actual. Use etiquetas para romper bucles anidados.

Detalles en: Bucles: while y for.

Más adelante estudiaremos más tipos de bucles para tratar con objetos.

La construcción “switch”

La construcción “switch” puede reemplazar múltiples revisiones con `if`. “switch” utiliza `===` (comparación estricta).

Por ejemplo:

```
let age = prompt('¿Su Edad?', 18);

switch (age) {
  case 18:

    alert("No funciona"); // el resultado de la petición es un string, no un

  case "18":
    alert("¡Funciona!");
    break;

  default:
    alert("Todo valor que no sea igual a uno de arriba");
}
```

Detalles en: La sentencia "switch".

Funciones

Cubrimos tres formas de crear una función en JavaScript:

1. Declaración de función: la función en el flujo del código principal

```
function sum(a, b) {
  let result = a + b;

  return result;
}
```

2. Expresión de función: la función en el contexto de una expresión

```
let sum = function(a, b) {
  let result = a + b;

  return result;
};
```

3. Funciones de flecha:

```
// la expresión en el lado derecho
let sum = (a, b) => a + b;

// o sintaxis multilínea { ... }, aquí necesita return:
let sum = (a, b) => {
  // ...
  return a + b;
}

// sin argumentos
let sayHi = () => alert("Hello");

// con un único argumento
let double = n => n * 2;
```

- Las funciones pueden tener variables locales: son aquellas declaradas dentro de su cuerpo. Estas variables solo son visibles dentro de la función.
- Los parámetros pueden tener valores predeterminados: `function sum(a = 1, b = 2) {...}`.
- Las funciones siempre devuelven algo. Si no hay `return`, entonces el resultado es `undefined`.

Más: ver Funciones, Funciones Flecha, lo básico.

Más por venir

Esa fue una breve lista de características de JavaScript. Por ahora solo hemos estudiado lo básico. Más adelante en el tutorial encontrará más características especiales y avanzadas de JavaScript.

Lección anterior | Próxima lección

Compartir

Mapa del Tutorial

Comentarios

- Si tiene sugerencias sobre qué mejorar, por favor enviar una propuesta de GitHub o una solicitud de extracción en lugar de comentar.
- Si no puede entender algo en el artículo, por favor explique.
- Para insertar algunas palabras de código, use la etiqueta `<code>`, para varias líneas – envolverlas en la etiqueta `<pre>`, para más de 10 líneas – utilice una entorno controlado (sandbox) (plnkr, jsbin, codepen...)

10 Comentarios

1 Acceder ▼

G

Únete a la conversación...

INICIAR SESIÓN CON

O REGISTRARSE CON DISQUS ?

Nombre

16 • Comparte

Mejores Más recientes Más antiguos

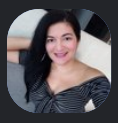
E

Edser Moreno

hace 2 años

Me lo leí todo (los fundamentos) en una semana, muy bien explicado todo, estoy pendiente para contribuir en la traducción de uno que otro pequeño gazapo de gramática que noté. **Reviso lo de las Solicitudes de Extracción (Pull request de git)** y procedo. Gracias.

2 0 Responder



Paola Torres

hace 3 años

Este tutorial es excelente, de una forma muy sencilla explica todos los conceptos de Javascript básico y entrega detalles y particularidades que no había visto ni en otros sitios web ni en clases presenciales.

1 0 Responder

Felipe Arce

hace 3 años

Muy buena explicacion de los fundamentos, ya llevo muchas lecturas y esta fue la mejor

1 0 Responder

O

Oscar Florez

hace 16 días

muy buen curso

0 0 Responder

A

Arlette Chavez

hace 7 meses

Nunca había llegado tan lejos aprendiendo Javascript TTTToTTTT muchas gracias por estas lecciones, he aprendido mucho, mil gracias.

0 0 Responder

Baltasar
hace un año

Muy bueno, me gusta

0 0 Responder

Masakode
hace 2 años

Me ha gustado el resumen. Está muy bien expliado.

0 0 Responder

Gustavo Redondo
hace 2 años

que buen resumen, todo muy claro

0 0 Responder

Rolando Luis Martin
hace 2 años

El nombre del website está mal definido, debería ser "La Biblia de Javascript" y ustedes los apóstoles. Gracias por las horas empleadas en tan coherente obra.

0 0 Responder

Luis Fernando Martinez Jorda
hace 3 años

me la salto!

0 0 Responder