J1

Introducción

SENPAI academy

Javascript

JavaScript es un lenguaje de programación que nació para dinamizar la web, pero que hoy por hoy se usa en “todos lados”.

Si bien se llaman parecidos, Java y JavaScript son dos mundos diferentes.

Es multi-paradigma, es decir que permite varios estilos de programación.

Está basado en el estándar ECMAScript (draft 2022).

Variables y constantes - ¿Cuándo usar var, let y const?

```
// Declaración.
let contador;

// Asignación - dar un valor.
contador = 2;

// Declaración + Asignación
let nombre = 'Senpai';

// Constante
const pi = 3.1415

// Reasignación
nombre = 'Academy';
pi = 5; // Error!
```

Las variables y constantes nos permiten guardar información.

Podemos crearlas utilizando las palabras **var**, **let** y **const**, aunque se aconseja dejar de usar **var**.

Las variables y constantes son prácticamente lo mismo, con la diferencia que la reasignación de una constante nos daría un error.

Tipos de datos: valores primitivos y objetos.

```
/* Valores primitivos */

// String (texto)
const nombre = "Sebastián";

// Number (número)
let edad = 27;

// Boolean (booleano)
let tieneSueño = true;

// null
let laNadaMisma = null;

// undefined
let noEstaDefinido;

/* Objetos */

let auto = {
    marca: 'Volkswagen',
    modelo: 'Beatle', // Fusca
    año: 1962
}

// Array (arreglos)
let profesiones = ['Desarrollador', 'Formador', 'Músico?'];
```

Hay varias diferencias entre los valores primitivos y los objetos, las cuales iremos viendo en el correr del curso.

Los valores primitivos no están compuestos por otros tipos de datos, mientras que los objetos pueden alojar varios.

Otros conceptos a tener en cuenta (pero revisar más adelante): inmutabilidad, herencia, referencias, instancias...

Operadores algebraicos **(+ - / * mod)**.

```
// Teniendo dos variables:
const primerValor = 10;
const segundoValor = 6;

// Suma
const suma = primerValor + segundoValor; // 16

// Resta
const resta = primerValor - segundoValor; // 4

// Multiplicación
const multi = primerValor * segundoValor; // 60

// División
const div = primerValor / segundoValor; // 1.66..

// Módulo
const mod = primerValor % segundoValor; // 4
```

Podemos utilizar operadores algebraicos para calcular valores.

Si bien lo normal es hacerlo con números, también podemos hacerlo con otros tipos de valores, e incluso mezclarlos.

Mezclar tipos de datos en operaciones algebraicas es generalmente un error.

¿Qué resultado tiene la siguiente suma:
“Hola” + “Senpai” + 2020?

Operadores comparativos y lógicos

```
// Teniendo la siguiente variable
const miValor = 10;

const mayorQueDiez = miValor > 10 // False
const menorQueDiez = miValor < 10 // False
const menorOIgualQueDiez = miValor <= 10 // True
const esDiez = primerValor === 10; // True


// NOT ! (negación)
const cuatro = 4; // 4
const esCuatro = cuatro === 4; // True
const noEsCuatro = !esCuatro; // False

// AND &&
const totalCompra = 1500;
const balanceDeCuenta = 12000;
const cuentaBloqueada = false;
const puedeComprar = !cuentaBloqueada && (total < balanceDeCuenta)

// OR ||
const mostrarAyuda = cuentaBloqueada || (balanceDeCuenta < total)
```

Los operadores comparativos siempre devuelven **true** o **false**, y nos permiten comparar valores.

Control de flujo (**if** y **switch** statements)

```
// Condicionales
let mensaje;

if (itemsCarrito > 0) {
    mensaje = 'Tienes ' + itemsCarrito + ' en tu carrito';
} else {
    mensaje = 'Tu carrito está vacío 😢'
}

// Switch
const estado = 'Sin stock';

switch(estado) {
    case 'Sin stock':
        // Deshabilitar botón de comprar
        // Mostrar mensaje de re-stock
    break;
    case 'En stock':
        // Habilitar botón de comprar
        // Mostrar mensaje de unidades disponibles
    break;
    case 'Descontinuado':
        // Ocultar botón de comprar
        // Mostrar mensaje de producto descontinuado
    break;
}
```

Podemos modificar el comportamiento de nuestro código utilizando condicionales o switches.

Se utiliza if/elseif/else en la mayoría de los casos de control de flujo.

Se utiliza switch cuando se evalúa un único valor y este puede tomar varios estados.

Una función salvadora: console.log()

Pueden utilizar console.log() para imprimir en consola todo tipo de cosas.

Por ejemplo, pueden utilizarlo varias veces en el flujo del código, para ir viendo paso a paso lo que sucede (más adelante puede que veamos cómo debuggear en el navegador de manera elegante).

Más información sobre console.log (y otros métodos del objeto Console): **link**

¡GRACIAS!

SENPAI
academy