Programación Imperativa

Módulo 2. Intro a JS

Sesión 4: Conociendo el entorno de desarrollo: Node y JavaScript

Agosto 16 de 2021

Node.js
Es un entorno de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome. Permite ejecutar JavaScript por fuera de un navegador web.

Variable
Un espacio de memoria donde se almacena un dato que podemos reutilizar a futuro.

Estándar: Forma correcta de escribir el código:
- camelCase
- snake_case
- kebab_case

Bloque de ejecución:
Todo el código que se encuentra dentro de las llaves.

Declarar variables
var: Se ignoran los bloques de código y la variable es global
let: Pertenece únicamente a un bloque de ejecución
const: No se puede cambiar el valor una vez asignado. No se puede redefinir en ningún otro lugar (?).

let y const son accesibles únicamente dentro del bloque de código en el que fueron declaradas.

Tipos de datos:
- Number: Número entero o decimal
- Strings:
- Booleanos
- NaN: Not a number. Ej “a”*2
- null: Valor vacío o desconocido
- undefined: Implícito si la variable no ha sido inicializado.
- Objeto literal
- Array

Sincrónico

- No usar “var”. Usar en cambio “let”, ya que el ámbito de las variables es siempre local.
- Existen 5 tipos de datos primitivos: number, string, boolean, NaN y undefined, symbol(?).
- Tipos de datos complejos: object y array
- typeof: Devuelve el tipo de dato.

```
let vacio = null
typeOf vacio // Devuelve object


 typeOf Infinity // Devuelve NaN
```

- Revisar el objeto Math

```
console.log(Math) // Muestra todos los métodos/propiedades del objeto Math

Math.round(20.49) // Devuelve 20
Math.round(20.50) // Devuelve 21
```

Zoom en VSC: Ctr +, Ctr -

Sesión 5. Trabajando con funciones

Agosto 18 de 2021

Operadores:

De Asignación:

let edad = 35;

Aritméticos:

10 + 15
10 – 15
10 * 15
10 / 15
15++
15--
15%5: Módulo

De Concatenación:

nombre + ` ` + apellido

Funciones
Una función es un bloque de código que nos permite realizar una tarea en particular. El uso de funciones estiliza el código y lo hace escalable.

```
function sumar(a,b) {
	return a + b;
}
```

Funciones declaradas
Son aquellas que se declaran usando la estructura básica. Pueden recibir un nombre, escrito a continuación de la palabra reservada **function**.

```
function sumar(a,b) {
	return a + b;
}
```

Funciones expresadas
Son aquellas que se asignan como valor de una variable. La función en sí no tiene nombre, es una función anónima.

```
let z = function(a,b) {
        return a +b;
}
```

Invocar una función:

```
sumar(3,4);
imprimirResultados();
```

Parámetros: Variables que se escriben cuando se define la función
Argumentos: Valores que se envían cuando se invoca la función

Valores por defecto en las funciones:

```
function saludar(nombre = "visitante", apellido = "anónimo") {
        return "Hola " + nombre + " " + apellido;
}

saludar(); // Retorna "Hola visitante anónimo"
```

Scope local
Las variables que se definen dentro de una función tienen scope local.

Scope global
Las variables que se definen por fuera de cualquier función tienen scope global.

Preguntas:
- Las variables que se definen dentro de una función deberían ser let, var o const? ¿O es indiferente?

Sincrónico

```
numero = 5
console.log(numero)
console.log(numero++)
console.log(numero) // Imprime 5, 5, 6

numero = 5
console.log(numero)
console.log(++numero)
console.log(numero) // Imprime 5, 6, 6
```

Punto y coma en JavaScript:
No es obligatorio

Literal templates

Utilizan comillas francesas (alt + 96):

Ej:
console.log(`Mi nombre es ${nombre} ${apellido}`)

Lo que va entre llaves es código de JavaScript

Consultar hoisting
Diferencias entre funciones declaradas y expresadas.

String.toUpperCase()
Math.PI

Clase 6.

Agosto 19 de 2021

Sincrónico

Los tipos primitivos se pasan como valor.

```
function promedioDeTresNumeros (x, y, z) {
	let suma = sumar(x,y);
	suma = sumar(suma,z);
	return dividir(suma,3);
}
```

Sesión 7. Operando lógicamente.

Agosto 23 de 2021.

Operadores de comparación
Devuelven un valor true o false

Igualdad simple y estricta

Comparación simple. (==)
Permite comparar dos valores.

Comparación estricta. (===)
Compara el valor y el tipo de dato.

Desigualdad simple y estricta

Desigualdad simple (!=)
Desigualdad estricta (!==)

Otras comparaciones
Mayor que (>)
Mayor o igual que (>=)
Menor que (<)
Menor o igual que (<=)

Operadores lógicos
Permiten unir sentencias de código

Conjunción (&&)
Disyunción (||)
Negación (!)

Resumen Operadores:
- Aritméticos
- Lógicos
- De comparación
- De concatenación.

Tipos de datos primitivos:
- Undefined
- Null
- Boolean
- Number (incluyendo Infinity y NaN)
- String

Truthy y Falsy

Falsy:
- false
- 0
- “”
- null
- undefined
- NaN

Truthy
- ‘0’
- ‘false’
- []
- {}
- function () {}

Reglas:
- false, 0 y cadenas vacías son todas equivalentes.
- null y undefined son equivalentes a ellos mismos y entre ellos, pero nada más.
- NaN no es equivalente a nada, incluido otro NaN.

Sincrónico

Sesión 8. Controlando el flujo de la aplicación.

Agosto 25 de 2021.

Condicionales

```
if (condicion1) {
	// Código si condicion1 es verdadera
} else if (condicion2) {
	// Código si condicion 2 es verdadera
} else {
	// Código si condicion1 y condicion2 son falsas
};
```

If ternario

test ? Código si test es true: código si test es false;

Ambos casos son requeridos. Ej:

test ? Código si test es true: “”;

Ej:

```
let max = x > y ? x: y;
console.log(max);
```

Switch

No requiere operador de comparación.

```
switch (expresion) {
	case caso1:
		// Código para el caso 1
		break;
	case caso2:
		// Código para el caso 2
		break;
	case caso3:
		// Código para el caso 2
		break;
	default:
		// Código si los demás casos son false.
};
```

Agrupamiento de casos:

```
switch (expresion) {
	case caso1:
	case caso2:
		// Código para los casos 1 y 2
		break;
	case caso3:
		// Código para el caso 2
		break;
};
```