Carrera: Tecnología en Desarrollo de Software

Nivel: I I I Semestre

Nombre del Estudiante: Segura Pesantes Edson Gabriel

Asignatura: DISEÑO WEB

Docente: ING. ERNESTO ROBLES

2024 2025

1. Conversión implícita y explícita de tipos en JavaScript

JavaScript es un lenguaje débilmente tipado, lo que significa que puede convertir valores entre tipos automáticamente (conversión implícita) o mediante funciones explícitas.

- Conversión implícita: JavaScript cambia el tipo automáticamente según el contexto, por ejemplo, al sumar un número con una cadena, el número se convierte en cadena.

```
let resultado = 5 + "10";  // "510"
console.log(resultado);
```

- Conversión explícita: El programador usa funciones para transformar tipos, como

```
let num = Number("123");  // convierte cadena a número
console.log(num);         // 123

let str = String(456);    // convierte número a cadena
console.log(str);         // "456"
```

2. Ejemplos de estructuras condicionales y repetitivas

Las estructuras condicionales permiten tomar decisiones según condiciones, y las repetitivas permiten repetir bloques de código.

- Condicional if - else:

```
let edad = 18;
if (edad >= 18) {
    console.log("Mayor de edad");
} else {
    console.log("Menor de edad");
}
```

- Condicional switch:

```
let color = "rojo";
switch (color) {
    case "rojo":
        console.log("Color rojo");
        break;
    case "azul":
        console.log("Color azul");
        break;
    default:
        console.log("Color desconocido");
}
```

```
let i = 0;
while (i < 3) {
    console.log("Número: " + i);
    i++;
}
```

3. Arreglos y objetos

- Declaración y acceso:

```
// Arreglo
let frutas = ["manzana", "pera", "banana"];
console.log(frutas[1]);  // "pera"

// Objeto
let persona = { nombre: "Ana", edad: 30 };
console.log(persona.nombre);  // "Ana"
```

- Métodos comunes de arreglos:

```
let numeros = [1, 2, 3, 4, 5];

// map: crea un nuevo arreglo multiplicando por 2 cada elemento
let dobles = numeros.map(n => n * 2);
console.log(dobles);  // [2,4,6,8,10]

// filter: crea un nuevo arreglo con números mayores a 3
let mayores = numeros.filter(n => n > 3);
console.log(mayores);  // [4,5]

// concat: une dos arreglos
let masNumeros = numeros.concat([6, 7, 8]);
console.log(masNumeros);  // [1,2,3,4,5,6,7,8]
```

- Acceso a propiedades de objetos:

```
let libro = {
    titulo: "Aprendiendo JS",
    autor: "Luis",
    paginas: 200
};

console.log(libro.titulo);       // "Aprendiendo JS"
console.log(libro["autor"]);     // "Luis"
```

```
// Declaración de un arreglo
const numeros = [1, 2, 3, 4, 5];

// Uso de map
const alCuadrado = numeros.map(n => n * n);
console.log(alCuadrado); // [1, 4, 9, 16, 25]

// Uso de filter
const mayoresQueTres = numeros.filter(n => n > 3);
console.log(mayoresQueTres); // [4, 5]

// Uso de concat
const otrosNumeros = [6, 7];
const todos = numeros.concat(otrosNumeros);
console.log(todos); // [1, 2, 3, 4, 5, 6, 7]

// Objeto y acceso a propiedades
const persona = {
  nombre: "Edson",
  edad: 18
};
console.log(persona.nombre); // Edson
```

4. Tipos de funciones

- Función declarada:

```
function saludar(nombre) {
    return "Hola " + nombre;
}
console.log(saludar("Juan"));
```

- Función flecha:

```
const multiplicar = (a, b) => a * b;
console.log(multiplicar(4, 5));
```

- Función callback (función como argumento)

```
function ejecutarFuncion(callback) {
    console.log("Ejecutando callback:");
    callback();
}

ejecutarFuncion(() => console.log("¡Callback ejecutado!"));
```

```javascript
// Función declarada
function saludar(nombre) {
  console.log("Hola " + nombre);
}
saludar("Edson");

// Función anónima
const despedir = function(nombre) {
  console.log("Chao " + nombre);
};
despedir("Edson");

// Función flecha
const sumar = (a, b) => a + b;
console.log(sumar(3, 4)); // 7

// Callback
function operar(a, b, callback) {
  return callback(a, b);
}
const multiplicar = (x, y) => x * y;
console.log(operar(5, 3, multiplicar)); // 15
```

5. Eventos comunes para llamar funciones

En JavaScript, los eventos permiten que el navegador ejecute funciones al ocurrir acciones del usuario:

Ejemplo:

```javascript
// Crear un objeto que representa a un estudiante
let estudiante = {
  nombre: "María",
  edad: 17,
  calificaciones: [8, 9.5, 10, 7.5]
};

// Función para calcular el promedio
function calcularPromedio(calificaciones) {
  let suma = 0;
  for (let i = 0; i < calificaciones.length; i++) {
    suma += calificaciones[i];
  }
  return suma / calificaciones.length;
}

// Usamos la función
let promedio = calcularPromedio(estudiante.calificaciones);

// Evaluamos el resultado usando una estructura condicional
if (promedio >= 9) {
  console.log(estudiante.nombre + " tiene un excelente promedio: " + promedio);
} else if (promedio >= 7) {
  console.log(estudiante.nombre + " tiene un buen promedio: " + promedio);
} else {
  console.log(estudiante.nombre + " necesita mejorar. Promedio: " + promedio);
}
```

6. Buenas prácticas en código

- Nombrado significativo: Usa nombres que describan la función o variable, por ejemplo, calcularEdad en vez de ce.

- Comentarios útiles: Explica la intención del código, no lo obvio.

- Funciones reutilizables: Crea funciones pequeñas que hagan solo una tarea para usar varias veces.

```
// Calcula el área de un rectángulo
function calcularArea(base, altura) {
    return base * altura;
}

// Uso
let area = calcularArea(10, 5);
console.log("Área:", area);
```

Bibliografía

• Mozilla Developer Network (MDN). (s.f.). *JavaScript Guide*. Mozilla. Recuperado el [fecha de consulta], de https://developer.mozilla.org/es/docs/Web/JavaScript/Guide

• W3Schools. (s.f.). *JavaScript Tutorial*. Recuperado el [fecha de consulta], de https://www.w3schools.com/js/

• Haverbeke, M. (2018). *Eloquent JavaScript* (3.ª ed.). No Starch Press. ISBN: 9781593279509

• JavaScript.info. (s.f.). *The Modern JavaScript Tutorial*. Recuperado el [fecha de consulta], de https://javascript.info/

• freeCodeCamp. (s.f.). *JavaScript Algorithms and Data Structures Certification*. Recuperado el [fecha de consulta], de https://www.freecodecamp.org/learn