Lenguaje de Marcas: Javascript

Iván Nieto Ruiz

il.nietoruiz@edu.gva.es

Scope en JavaScript

¿Qué es el Scope?
El **Scope** o **alcance** en JavaScript define dónde pueden accederse las variables dentro del código.

JavaScript tiene tres tipos principales de scope:

- **Global**
- **Local o de función**
- **De bloque**

Scope Global
Variables declaradas fuera de cualquier función o bloque pueden ser accedidas desde cualquier parte del programa.

```
var globalVar = "Soy una variable global";

function mostrarGlobal() {
        console.log(globalVar);
}
mostrarGlobal();
```

Scope en JavaScript

Scope de Función

Variables declaradas dentro de una función solo existen dentro de esa función.

```
function funcionScope() {
    let localVar = "Solo existo aquí";
    console.log(localVar);
}
funcionScope(); ❌ Error
```

Scope de Bloque

- Variables declaradas con `let` o `const` dentro de un bloque ({}) solo existen dentro de ese bloque.

```
if (true) {
    let blockVar = "Solo existo en este bloque";
    console.log(blockVar);
}
console.log(blockVar); ❌ Error
```

Objetos en JavaScript

¿Qué son los Objetos?

- Son estructuras de datos que agrupan **propiedades** y **métodos** en **una sola entidad.**
- Se representan con llaves **{}** y contienen pares **clave-valor**.

```
const persona = {
    nombre: "Juan",
    edad: 30,
    saludar: function() {
        console.log("Hola, soy " + this.nombre);
    }
};

console.log(persona.nombre); // "Juan"
persona.saludar(); // "Hola, soy Juan"
```

- Se acceden con **objeto.propiedad** o **objeto["propiedad"]**
- `this` dentro de un método **hace referencia al propio objeto.**

Desde **ES6**, podemos escribir métodos en un objeto de forma más corta:

```
const persona = {
    nombre: "Juan",
    edad: 30,
    saludar() { // Forma simplificada
        console.log(`Hola, soy ${this.nombre}`);
    }
};
```

Manipulación de Objetos

Agregar y Modificar Propiedades

```
persona.apellido = "Pérez";
persona.edad = 31;
console.log(persona);
```

Eliminar Propiedades

```
delete persona.edad;
console.log(persona);
```

Podemos modificar, agregar o eliminar propiedades dinámicamente.

Recorrer un Objeto

```
for (let clave in persona) {
    console.log(`${clave}: ${persona[clave]}`);
}
```

Manipulación de Objetos

Arrays de Objetos

```
const estudiantes = [
        { nombre: "Ana", edad: 20 },
        { nombre: "Luis", edad: 22 },
        { nombre: "Marta", edad: 19 }
];
```

map() - Transformar datos

```
const nombres = estudiantes.map(est => est.nombre);
console.log(nombres); // ["Ana", "Luis", "Marta"]
```

.filter() - Filtrar datos

```
const mayoresDe20 = estudiantes.filter(est => est.edad > 20);
console.log(mayoresDe20);
```

Manipulación de Objetos

```
const estudiantes = [
        { nombre: "Ana", edad: 20 },
        { nombre: "Luis", edad: 22 },
        { nombre: "Marta", edad: 19 }
];
```

reduce() - Acumular valores

```
const sumaEdades = estudiantes.reduce((acc, est) => acc + est.edad, 0);
console.log(sumaEdades); // 61
```

forEach() - Iterar sin retornar valor

```
estudiantes.forEach(est => console.log(est.nombre));
```

Uso del Spread Operator (. . .)

El Spread Operator (. . .) facilita copias y fusiones de arrays y objetos.

Copiar Arrays

```
const numeros = [1, 2, 3];
const copiaNumeros = [...numeros];
console.log(copiaNumeros); // [1, 2, 3]
```

Fusionar Objetos

```
const usuario = { nombre: "Ana", edad: 25 };
const datosAdicionales = { ciudad: "Madrid", ocupacion: "Desarrolladora" };
const usuarioCompleto = { ...usuario, ...datosAdicionales };
console.log(usuarioCompleto);
```

¿Preguntas?