Sistemas Distribuidos

Bloque I

Introducción a los Sistemas Distribuidos: caracterización y arquitecturas.

Tema 1.5:

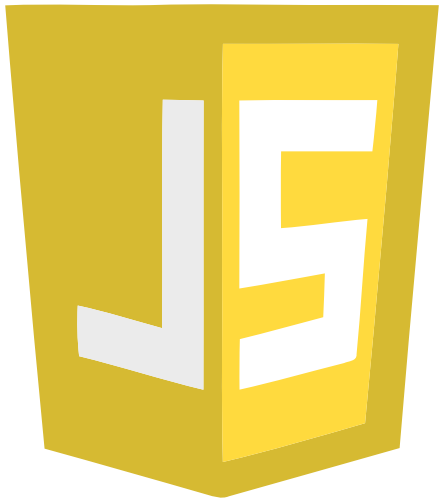
JavaScript

Nicolás Rodríguez
nicolas.rodriguez@urjc.es

Índice

Definición y origen

- JavaScript es un lenguaje de programación interpretado, de alto nivel, que se utiliza principalmente para scripts en páginas web.
- Fue creado por Brendan Eich en 1995 y originalmente se llamaba "Mocha", luego "LiveScript", y finalmente "JavaScript".



- A pesar de su nombre, JavaScript no está relacionado con Java y fue desarrollado independientemente.

Índice

Características principales

- Es un lenguaje de programación dinámico que soporta estilos de programación orientada a objetos, imperativa y funcional.
- Se ejecuta principalmente en el navegador, aunque también se puede usar en el servidor (por ejemplo, Node.js).
- Es conocido por su capacidad para agregar interactividad a las páginas web, mejorar la experiencia del usuario y crear aplicaciones web modernas.

Características principales

- JavaScript es uno de los tres lenguajes fundamentales en el desarrollo web, junto con HTML y CSS.
- Ha jugado un papel crucial en la evolución de la web, desde páginas estáticas hasta aplicaciones web dinámicas y complejas.
- Su popularidad y adopción han crecido enormemente, y ahora es una herramienta indispensable para cualquier desarrollador web.

Características principales

- Un script JavaScript típico se compone de declaraciones que se ejecutan de arriba hacia abajo.
- Cada declaración suele terminar con un punto y coma (;), aunque este es opcional debido al "Automatic Semicolon Insertion" en JavaScript.
- Declaración de variables:
  - Palabras clave var, let y const para la declaración de variables.
  - var para variables con scope de función
  - let para variables con scope de bloque
  - const para valores constantes.

```
let mensaje = "Hola, mundo!";
console.log(mensaje);
```

Índice

Control de flujo

```
let edad = 20;
if (edad < 18) {
        console.log("Eres menor de edad.");
} else if (edad < 60) {
        console.log("Eres adulto.");
} else {
        console.log("Eres adulto mayor.");
```

```
let nombres = ["Ana", "Luis", "Carlos", "Marta"];
for (let i = 0; i < nombres.length; i++) {
        console.log(nombres[i]);
}
```

```
let numero = 8;
while (numero > 1) {
        console.log(numero);
        numero = numero / 2;
}
```

```
let dia = "Martes";
switch (dia) {
        case "Lunes": console.log("Inicia la semana de trabajo.");
        break;
case "Martes":
case "Miércoles":
case "Jueves":
        console.log("En medio de la semana laboral.");
        break;
case "Viernes":
        console.log("Casi es fin de semana!");
        break;
case "Sábado":
case "Domingo":
        console.log("Es fin de semana, a disfrutar!");
        break;
default: console.log("No es un día válido.");
```

Índice

Tipos de datos

- Primitivos:
  - Number: Representa tanto enteros como flotantes. Ejemplo: let edad = 25;.
  - String: Cadenas de caracteres, ya sea en comillas simples o dobles. Ejemplo: let nombre = "Alice";.
  - Boolean: Representa valores de verdad (true o false). Ejemplo: let estaActivo = true;.
  - Undefined: Indica una variable que no ha sido asignada. Ejemplo: let resultado;.
  - Null: Representa una ausencia intencional de cualquier valor de objeto. Ejemplo: let respuesta = null;.
- Complejos:
  - Object: Colecciones de propiedades.
    - Ejemplo: let persona = {nombre: "Alice", edad: 25};.
  - Array: Lista ordenada de datos.
    - Ejemplo: let numeros = [1, 2, 3, 4, 5];.

Índice

Operadores

- Aritméticos: Usados con valores numéricos para realizar operaciones matemáticas comunes, como +, -, *, /, y % (módulo).
- Asignación: Asignan un valor a una variable. El más simple es =, pero también incluyen operadores compuestos como +=, -=, *=, y /=.
- Comparación: Comparan dos valores y retornan un booleano. Incluyen == (igualdad), === (igualdad estricta), != (desigualdad), !== (desigualdad estricta), >, <, >=, <=.
- Lógicos: Utilizados para determinar la lógica entre variables o valores. Incluyen && (y), || (o), ! (no).
- String: Concatenación (+)

Operadores

- La diferencia entre igualdad (==) e igualdad estricta (===) en JavaScript es una distinción importante y es fundamental :
- Igualdad (==)
  - Descripción: La igualdad, o "igualdad abstracta", compara dos valores por su igualdad después de convertir ambos valores a un tipo común.
  - Conversión de Tipo: Si los dos valores tienen tipos diferentes, JavaScript intenta convertirlos a un tipo común antes de hacer la comparación, lo que se conoce como "coerción de tipo". Ejemplo:
    - 0 == '0' evalúa como true porque el string '0' se convierte en el número 0 antes de la comparación.
    - null == undefined también es true ya que ambos se consideran ausencia de valor.
- Igualdad Estricta (===)
  - Descripción: La igualdad estricta, o "igualdad de tipo", compara tanto el valor como el tipo de los dos operandos, sin realizar la conversión de tipo.
  - Sin Conversión de Tipo: Si los dos valores tienen tipos diferentes, la comparación es automáticamente false. Ejemplo:
    - 0 === '0' es false porque aunque el valor numérico es el mismo, los tipos son diferentes (número vs. string).
    - null === undefined es false ya que null y undefined son tipos diferentes.

Índice

Funciones

- Las funciones son fundamentales en JavaScript para organizar y reutilizar código.
- Existen las funciones anónimas, que no tienen nombre y generalmente se usan como argumento de otras funciones.
- Las arrow functions (introducidas en ES6) tienen una sintaxis más corta para escribir funciones.

```
function saludo(nombre) {
                return `Hola, ${nombre}!`; }
console.log(saludo("Alice"));
```

```
document.getElementById('miBoton’).
addEventListener('click', function() {
 alert('¡El botón fue pulsado!'); });
```

```
const suma = (a, b) => a + b;
 console.log(suma(5, 3));
```

Funciones

- Existen también las closures functions, como funciones que recuerdan el entorno en el que fueron creadas, lo que permite técnicas avanzadas como encapsulación y fabricación de funciones.

```
function crearSaludo(saludo) {
            return function(nombre) {
                        return `${saludo}, ${nombre}!`;
                        }
            }
const saludoPersonal = crearSaludo("Hola");
console.log(saludoPersonal("Alice"));
```

Índice

Objetos

- Un objeto es una colección de propiedades, y una propiedad es una asociación entre un nombre (o clave) y un valor.
- El valor de una propiedad puede ser una función, en cuyo caso la propiedad es conocida como un método.

```
let persona = {
        nombre: "Alice",
        edad: 25,
        saludar: function() {
                console.log(`Hola, mi nombre es ${this.nombre}`);
        }
 };
persona.saludar();
```

Listas

- Colección “ordenada” de valores, que pueden ser de cualquier tipo, incluidos otras listas y objetos.
- Métodos comunes para manipulación de lista: push, pop, shift, unshift, map, filter, reduce.

```
let frutas = ['manzana', 'banana', 'cereza’];
frutas.push('naranja’);
console.log(frutas);
// ['manzana', 'banana', 'cereza', 'naranja']
```

```
let frutas = ["manzana", "banana", "cereza"];
console.log(frutas[1]);
// Accede al segundo elemento: banana
```

```
let numeros = [2, 3, 4]; numeros.unshift(1);
console.log(numeros);
// [1, 2, 3, 4]
```

```
let numeros = [1, 2, 3, 4, 5];
let cuadrados = numeros.map(num => num * num);
console.log(cuadrados);
// [1, 4, 9, 16, 25]
```

Índice

DOM y BOM

- El DOM es una interfaz de programación que representa los documentos HTML y XML, como una estructura de árbol, donde cada nodo es un objeto que representa una parte del documento.
- JavaScript puede utilizar el DOM para manipular el contenido, la estructura y el estilo de los documentos web.
- El BOM es una representación del navegador que incluye objetos para trabajar con todo lo que no forma parte del documento HTML en sí, como window, navigator, screen, location, history.
- El BOM permite interactuar con el navegador, incluyendo manipulación de ventanas, detección de características del navegador y más.

DOM y BOM

- Ejemplo 1: Cambiando el contenido de un elemento

```
// Cambiando el contenido de un párrafo
document.getElementById('miParrafo').textContent = 'Texto actualizado!';
```

- Ejemplo 2: Añadiendo un nuevo elemento

```
// Creando un nuevo elemento y añadiéndolo al DOM
let nuevoDiv = document.createElement('div');
nuevoDiv.textContent = '¡Soy un nuevo div!';
document.body.appendChild(nuevoDiv);
```

- Ejemplo 3: Modificando estilos

```
// Cambiando el color de fondo de un elemento
document.getElementById('miDiv').style.backgroundColor = 'lightblue';
```

DOM y BOM

- Ejemplo 4: Añadiendo y quitando clases CSS

```
// Añadiendo una clase CSS a un elemento
document.getElementById('miElemento').classList.add('mi-clase-css’);
// Quitando una clase CSS
document.getElementById('miElemento').classList.remove('otra-clase-css');
```

- Ejemplo 5: Manejador de eventos

```
// Añadiendo un evento click a un botón
let miBoton = document.getElementById('miBoton');
miBoton.addEventListener('click', function() {
        alert('¡Botón pulsado!’);
});
```

- Ejemplo 6: Obteniendo el valor de un campo de formulario

```
// Obteniendo el valor de un campo de texto
let valor = document.getElementById('miCampoTexto').value;
console.log(valor);
```

Índice

Eventos

- Los eventos en JavaScript son acciones o sucesos que ocurren en la página web, como clics, pulsaciones de teclas, movimientos del mouse, etc.
- JavaScript permite responder a estos eventos mediante el uso de funciones.
- Evento1: primer argumento es el tipo de evento y el segundo es la función que se ejecuta cuando ocurre el evento.

```
document.getElementById('miBoton').addEventListener('click', function() {
console.log('Botón pulsado');
```

- Evento2: Prevención del comportamiento predeterminado de eventos usando event.preventDefault().

```
document.getElementById('miEnlace').addEventListener('click',
function(event) { event.preventDefault();
```

Eventos

- Ejemplo 1: Cambiar el estilo de un elemento en un evento

```
document.getElementById('miBoton').addEventListener('mouseover', function() {
        this.style.backgroundColor = 'lightblue'; });
document.getElementById('miBoton').addEventListener('mouseout', function() {
this.style.backgroundColor = ''; });
```

- Ejemplo 2: Validar un formulario antes de enviar

```
document.getElementById('miFormulario').addEventListener('submit',
function(event) {
        let campo = document.getElementById('miCampoTexto').value;
        if (campo.length < 3) {
                alert('El texto ingresado es demasiado corto.');
                event.preventDefault();
        }
});
```

Ejercicio 1

- Desarrollar una página web que permita al usuario añadir elementos a una lista y verlos reflejados en tiempo real en la página. Se pide:
  - Estructura HTML:
    - Crea un archivo HTML con un formulario que incluya un campo de texto (input) y un botón (button) para añadir elementos a la lista.
    - Incluye un elemento ul vacío donde se añadirán los nuevos elementos de la lista (li).
  - JavaScript para manejar eventos:
    - Script que “espere” el evento 'click' en el botón de añadir.
    - Cuando se haga clic en el botón, el script deberá leer el valor del campo de texto y añadirlo como un nuevo elemento li dentro del ul.
  - Validación:
    - Asegúrate de que el campo de texto no esté vacío antes de añadir un elemento a la lista. Si está vacío, muestra una alerta o mensaje de error al usuario.
    - Limpiar el campo de texto, después de añadir un elemento a la lista, limpia el campo de texto para que esté listo para la siguiente entrada.

Ejercicio 2

- Desarrollar una página web que permita al usuario cambiar el tema de color de la página (por ejemplo, de claro a oscuro) utilizando JavaScript y el DOM. Se pide:
  - Estructura HTML:
    - Crea un archivo HTML con un botón o interruptor (switch) que el usuario pueda hacer clic para cambiar el tema de color de la página.
    - Incluye varios elementos en la página, como un encabezado, un párrafo y un área de contenido, para que el cambio de tema sea evidente.
  - CSS para tema:
    - Define dos temas de color en tu archivo CSS: uno claro y otro oscuro. Puedes hacerlo definiendo clases que cambien propiedades como background-color y color.
  - JavaScript para cambiar temas:
    - Escribe un script que escuche el evento 'click' en el botón de cambio de tema.
    - Cuando se haga clic en el botón, el script deberá alternar entre las clases de tema claro y oscuro en los elementos relevantes de la página.

Sistemas Distribuidos

Bloque I

Introducción a los Sistemas Distribuidos: caracterización y arquitecturas.

Tema 1.5:

JavaScript

Nicolás Rodríguez
nicolas.rodriguez@urjc.es