Clase 1: Nivelación JS

Sitio:	Centro de E-Learning - UTN.BA	Imprimido por:	Nilo Crespi
Curso:	Curso de Backend Developer - Turno Noche	Día:	Friday, 23 de January de 2026, 10:10
Libro:	Clase 1: Nivelación JS		

Descripción

Objetivos de la clase:

- Repasar los conceptos fundamentales de JavaScript que servirán de base para comprender TypeScript.
- Comprender la sintaxis básica de JavaScript, incluyendo variables, operadores y estructuras de control de flujo.
- Familiarizarse con los conceptos de funciones, eventos y objetos en JavaScript.
- Aprender a utilizar las herramientas de depuración y consola para identificar y solucionar errores en el código.

Tabla de contenidos

1. Introducción

En esta clase, repasaremos los conceptos fundamentales de JavaScript que hemos visto en el módulo de Front-End. Este repaso nos permitirá reforzar la base necesaria para avanzar hacia TypeScript con mayor confianza.

Volveremos a trabajar con la sintaxis básica de JavaScript, incluyendo variables, operadores y estructuras de control de flujo, conceptos esenciales para la programación en este lenguaje. También revisaremos funciones, eventos y objetos, así como el uso de herramientas de depuración para identificar y corregir errores en el código.

Además, realizaremos ejercicios prácticos que nos ayudarán a aplicar estos conocimientos en situaciones reales, permitiéndonos consolidar lo aprendido y mejorar nuestra lógica de programación.

¡Comencemos!

2. Variables y Funciones

Variables básicas

Una variable es un espacio en la memoria de la computadora que nos permite almacenar información para luego recuperarla y operarla.

Sus tipos básicos son:

- Números
- Strings
- Booleanos

Variables

```
var nombre = 'Juan';

let edad = 25;

const PI = 3.14159;
```

Funciones

Una función un **conjunto de instrucciones** que realiza una tarea y se define con la palabra function y luego el nombre que le queramos poner.

Las funciones pueden **devolver un resultado**. La sentencia **return** finaliza la ejecución de la función y especifica un valor para ser devuelto.

```
function sumar(a, b) {

  return a + b;

}

let resultado = sumar(3, 4); // resultado será igual a 7

let resultado = sumar(3, 4); // resultado será igual a 7
```

3. Operadores

Operadores

Los operadores nos ayudan a armar nuestras condiciones. Tenemos de dos tipos:

- **lógicos:** and, or, not
- **comparación:** mayor, menor, mayor igual, menor igual, igual, estrictamente igual. desigual, estrictamente desigual.

```
let x = 5;

let y = 3;


let suma = x + y; // suma será igual a 8

let suma = x + y; // suma será igual a 8

let resta = x - y; // resta será igual a 2

let esMayor = x > y; // esMayor será igual a true
```

4. Condicionales y Bucles

Condicionales

Las i**nstrucciones condicionales** se usan para realizar las diferentes acciones en el código según una condición que se evalúa como verdadera o falsa:

- If / if else / if else if
- Switch

```
let edad = 18;



if (edad >= 18) {
  console.log('Eres mayor de edad');
} else {
  console.log('Eres menor de edad');
}



for (let i = 0; i < 5; i++) {
  console.log(i);
}


let i = 0;

while (i < 5) {

  console.log(i);
  i++;
}
```

Bucles

Los bucles (loops) son utilizados para realizar **tareas repetitivas** con **base en una condición**. Las condiciones típicamente devuelven true o false al ser evaluados. El bucle continuará ejecutándose hasta que la condición devuelva false. Conocemos tres tipos de bucles:

- While
- Do...while
- For

```
// Uso de ciclos
for (let i = 0; i < 10; i++) {
  console.log(i);
}
```

```
let nombres = ["Juan", "Pedro", "María"];
for (let nombre of nombres) {
  console.log(nombre);
}
```

5. Objetos

Objetos literales

Un objeto es un conjunto de propiedades en donde cada propiedad está compuesta de una llave y un valor.

Son objetos literales cuyas propiedades están **declaradas textualmente** en el código.

Los objetos en JavaScript nos ayudan agrupar información.

DOM

Es la **estructura de objetos** que genera el **navegador** cuando se carga un documento y se puede alterar mediante Javascript para cambiar dinámicamente los contenidos y aspecto de la página.

Podríamos decir que es la forma en que JavaScript interpreta nuestro código HTML.

6. ¡A practicar!

Actividades

Actividad de depuración de código:

Se te proporcionará un código en Javascript con errores. Utiliza herramientas de depuración y la consola de tu entorno de desarrollo para identificar y corregir los errores en el código.

Una vez que hayas corregido los errores, ejecuta el código y verifica que funcione correctamente.

1.

```
Código a corregir:
// Código con errores
function suma(a, b) {
return a + b;
}
var resultado = suma(2, 3);
console.log(resultado);
console.log("El resultado de la suma es: " + resultado);
```

2.

Utiliza la consola de tu entorno de desarrollo para seguir el valor de la variable a medida que se ejecuta el código.

Registra el valor de la variable en cada iteración y al final de la ejecución del código.

```
// Código a seguir
let contador = 0;
for (let i = 0; i < 10; i++) {
contador += i;
}
console.log("El valor del contador es: " + contador);
```

3.

- Identificar y corregir los errores de tipo en el código.
- Una vez que hayas corregido los errores, ejecuta el código y verifica que funcione correctamente.

```
let numero = "2";
let resultado = numero + 3; → let resultado = parseInt(numero) + 3;
console.log("El resultado es: " + resultado);
```

7. Espacio de práctica

Espacio de práctica

1. Escribe una función que tome un número como parámetro y devuelva el doble de ese número. Por ejemplo, si la función recibe el número 5, debe devolver 10.
2. Escribe una función que tome un array de números como parámetro y devuelva la suma de todos los números en el array. Por ejemplo, si la función recibe el array [1, 2, 3, 4], debe devolver 10.
3. Escribe una función que tome un string como parámetro y devuelva el mismo string pero con todas las letras en mayúscula. Por ejemplo, si la función recibe el string "hola mundo", debe devolver "HOLA MUNDO".
4. Escribe una función que tome un objeto como parámetro y devuelva el número de propiedades que tiene el objeto. Por ejemplo, si la función recibe el objeto {nombre: "Juan", edad: 30, ciudad: "Buenos Aires"}, debe devolver 3.
5. Escribe una función que tome un número como parámetro y devuelva un array con todos los números enteros desde 1 hasta el número pasado como parámetro. Por ejemplo, si la función recibe el número 5, debe devolver el array [1, 2, 3, 4, 5].