¡Les damos la bienvenida!

¿Comenzamos?

Esta clase va a ser

- grabada

Principios básicos de Javascript

Temario

00

Principios de programación Backend

- ✓ Programación web
- ✓ Distintas maneras de probar Javascript
- ✓ Tipos de datos en Javascript

01

Principios básicos de Javascript

- ✓ Tipos de datos y variables en Javascript
- ✓ Javascript y ES6
- ✓ Funciones en Javascript

02

Funcionalidades de los lenguajes ECMAScript

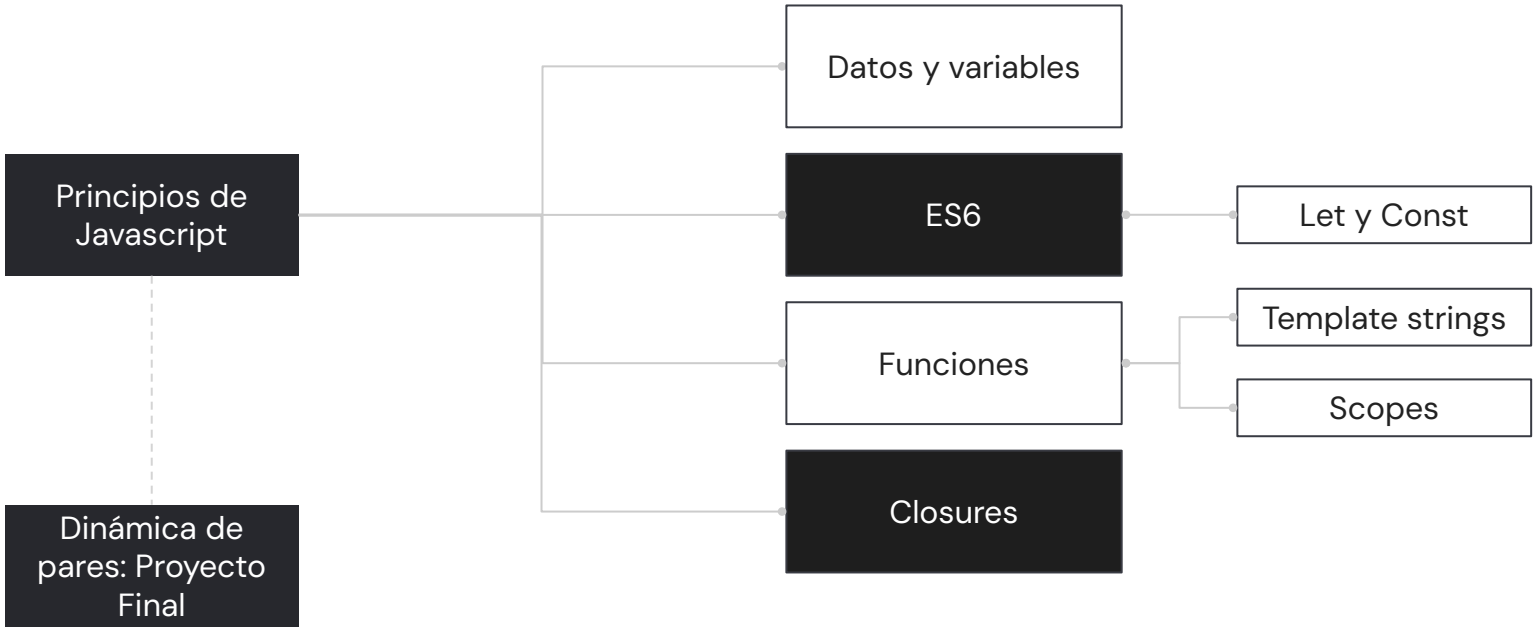
- ✓ Nuevas funciones de ECMAScript
- ✓ Funciones más utilizadas en backend

CODERHOUSE

Objetivos de la clase

- Familiarizarse con las estructuras y conceptos fundamentales al programar utilizando Javascript
- Conocer los nuevos elementos de lenguaje aportados por ES6

MAPA DE CONCEPTOS



Tipos de datos y variables en Javascript

		Nombre	Descripción
Tipos de datos en Javascript	**Tipos primitivos**	String	Cadenas de texto
		Number	Valores numéricos
		Boolean	True/false
		Null	Tipo especial, contiene null
		Undefined	Tipo especial, contiene undefined.
	Tipos objeto	Object	Un objeto es una colección de propiedades clave-valor.
		Array	Serie de elementos o formación tipo vector o matriz.
		Function	En JavaScript, las funciones también son objetos. Es decir, se pueden almacenar en variables, pasarlas como argumentos y/o retornarlas.
		Objetos especiales	Objeto global (Infinity, NaN, etc.)
			Objeto prototipo
			Otros

Para pensar

Una variable es un **espacio de memoria** apartado por la computadora, para poder **guardar un dato.**

¿Verdadero o falso?

Contesta mediante el chat de Zoom

Para pensar

Una variable **no puede cambiar su valor,** a pesar de que el programa lo necesite. Por consiguiente, no puede reutilizarse.

¿Verdadero o falso?

Contesta mediante el chat de Zoom

Algunos cambios importantes de ES6

Let y const

`let y const` son dos formas de declarar variables en JavaScript introducidas en ES6 que limitan el ámbito de la variable al bloque en que fue declarada (antes de ES6 esto no era así).
Es posible que se encuentren con ejemplos y código en internet utilizando la palabra reservada “var” para crear variables. Esta es la manera en que se hacía antes de ES6 ¡y no se recomienda su uso!

CODERHOUSE

Scopes

El scope define el alcance de una variable o constante a un cierto contexto. Esto permite utilizar el mismo nombre para diferentes variables, sin confundir al computador.

El **scope global** afectará a todo el nivel del archivo en el que trabajemos, mientras que el **scope local** afectará a la función o bloque en el que esté declarado.

Mutabilidad y const

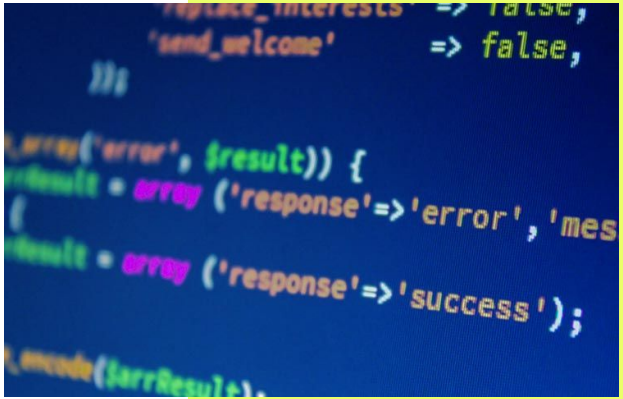
- ✓ Mientras que con `let` una variable puede ser reasignada, con `const` no es posible.
- ✓ Si se intenta reasignar una constante se obtendrá un error tipo TypeError.
- ✓ Pero que no se puedan reasignar no significa que sean inmutables.
- ✓ Si el valor de una constante es algo "mutable", como un array o un objeto, se pueden cambiar los valores internos de sus elementos.

NO REASIGNABLE ≠ INMUTABLE

Funciones en Javascript

¿Qué es una función?

Son bloques de instrucciones que trabajan sobre un scope interno (conocido como scope local). Pueden encontrarse en su sintaxis básica o en su notación flecha.



CODERHOUSE

Ejemplo en vivo

- Realizar una función con estructura básica. Se destacarán los elementos y un caso de uso.
- Realizar la misma función con estructura flecha. Se destacarán las diferencias.

Template Strings

Template strings

- ✓ Funcionan como un **superset** de una string.
- ✓ Permite **incrustar** información dentro de ella, evitando la concatenación.
- ✓ Reconoce los saltos de línea, para mantener el formato.
- ✓ Pero más importante aún es que no presenta el límite de caracteres de una string normal (Permitiendo hacer estructuras más complejas, como plantillas)

```
`texto de cadena de caracteres`

`línea 1 de la cadena de caracteres
 línea 2 de la cadena de caracteres`

`texto de cadena de caracteres ${expresión} texto adicional`
```

CODERHOUSE

HANDS ON LAB

Funciones

¿Cómo lo hacemos? Definiremos la función “mostrarLista”, la cual recibirá un arreglo con elementos como parámetro.

- ✓ Si la lista está vacía, devolver un mensaje indicando “Lista vacía”.
- ✓ Si la lista cuenta con elementos, mostrarlos 1 por 1 en consola. Finalizar el proceso devolviendo la longitud de la lista (Utilizar template strings)
- ✓ Invocar la función con los casos de prueba.

Se espera una duración de 10 minutos.

Break

¡10 minutos y volvemos!

Closures en Javascript

Closures

Una cláusula o closure es una función que puede mantener variables declaradas de manera interna, además de contar con una función que pueda acceder a ambos ámbitos, tanto el suyo como el de su función padre, logrando así un efecto de **variable privada.**

Este efecto de encapsulamiento se utilizaba con anterioridad debido a la falta de la implementación de **clases** en Javascript, sin embargo, al introducirse las clases, el cambio fue muy notorio, dejando a las closures en desuso, o bien en casos de uso muy limitados e insuficientes para códigos enterprise.

Uso de clases en Javascript

Clases

Una clase es una representación de una entidad. Nos sirve para modelar múltiples cosas como: un auto, una persona, o bien cosas más abstractas como: un administrador de archivos, un conector a base de datos, etc.

Las clases funcionan como **moldes**, por lo que una vez definida una de éstas, podemos crear múltiples objetos con la misma forma y con las mismas funcionalidades. A éstas se les llaman **instancias.**

HANDS ON LAB

Creación de una clase contador

¿Cómo lo hacemos? **Se creará una clase que permitirá llevar cuentas individuales según cada responsable.**

- ✓ Definir clase Contador
- ✓ La clase se creará con un nombre, representando al responsable del contador.
- ✓ El contador debe inicializarse en 0
- ✓ Debe existir una variable estática que funcione como **contador global** de todas las instancias de contador creadas.

Se espera una duración de 10 minutos.

HANDS ON LAB

Creación de una clase contador

- ✓ Definir el método getResponsable, el cual debe devolver el responsable de dicho contador.
- ✓ Definir el método contar, el cual debe incrementar, tanto su cuenta individual, como la cuenta global.
- ✓ Definir el método getCuentaIndividual, el cual debe devolver sólo la cuenta individual del contador
- ✓ Definir el método getCuentaGlobal, el cual debe devolver la variable estática con el conteo global.
- ✓ Realizar prueba de individualidad entre las instancias.

¿Preguntas?

Opina y valora esta clase

Resumen
de la clase hoy

- ✓ Repaso sobre tipos de datos y variables en Javascript
- ✓ Javascript y ES6
- ✓ Funciones en Javascript
- ✓ Uso de clases en Javascript

Muchas gracias.