



***¡LES DAMOS LA
BIENVENIDA!***

¿Arrancamos?

***RECUERDA PONER A GRABAR LA
CLASE***



CODER HOUSE

PRESENTACIÓN DE ESTUDIANTES

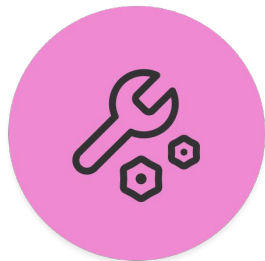


Por encuestas de Zoom:

1. País
2. Conocimientos previos de Backend
3. ¿Por qué elegiste el curso?

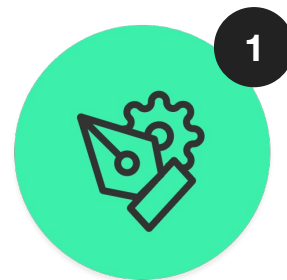
DESAFÍOS Y ENTREGABLES

Son actividades o ejercicios que se realizan durante la cursada, para enfocarse en la práctica.



Desafíos genéricos

Ayudan a poner en práctica los conceptos y la teoría vista en clase. No deben ser subidos a la plataforma.



Desafíos entregables

Relacionados completamente con el **Proyecto Final**. Deben ser subidos obligatoriamente a la plataforma hasta 7 días luego de la clase para que sean corregidos.

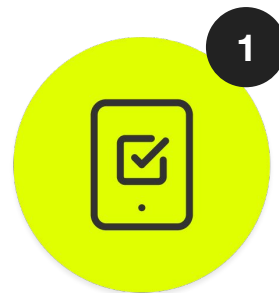
DESAFÍOS Y ENTREGABLES

Son actividades o ejercicios que se realizan durante la cursada, para enfocarse en la práctica.



Desafíos complementarios

Desafíos que complementan a los entregables. Son optativos y, de ser subidos a la plataforma a tiempo y aprobados, suman puntos para el top 10.



Entregas del Proyecto Final

Entregas con el estado de avance de tu **proyecto final** que deberás subir a la plataforma a lo largo del curso y hasta 7 días luego de la clase, para ser corregidas por tu docente o tutor/a.



PROYECTO FINAL

El Proyecto Final se construye a partir de los **desafíos** que se realizan clase a clase. Se va creando a medida que el estudiante sube los desafíos entregables a nuestra plataforma.

El objetivo es que cada estudiante pueda utilizar su Proyecto Final como parte de su portfolio personal.

El **proyecto final** se debe subir a la plataforma la ante-última o última clase del curso. *En caso de no hacerlo tendrás 20 días a partir de la finalización del curso para cargarlo en la plataforma. Pasados esos días el botón de entrega se inhabilitará.*

¿CUÁL ES NUESTRO PROYECTO FINAL?

CODER HOUSE

E-commerce backend

Proyecto
Final



Desarrollarás el **backend de una aplicación de e-commerce** para poder vender productos de un rubro a elección.

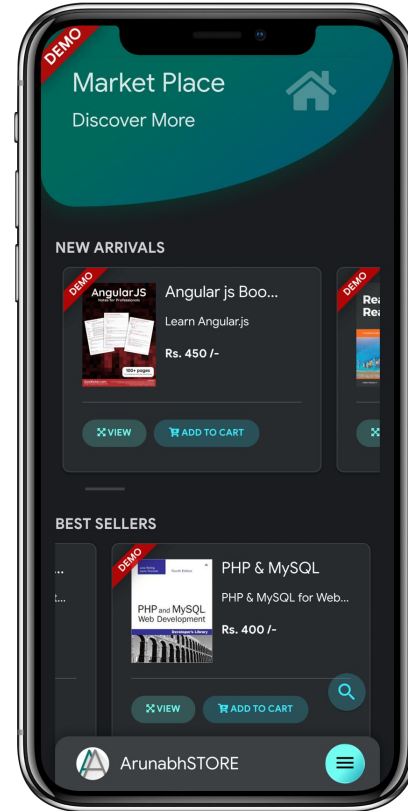
El servidor se basará en un diseño en capas, orientado a MVC y su código contendrá modernas estructuras de programación del poderoso lenguaje ECMAScript.

El formato del servidor será un proyecto consistente en una estructura de archivos, carpetas, configuraciones y base de datos, listas para ejecutarse bajo el entorno Node.js.

EJEMPLO

- [Market Place](#) by @arunabharjun
([Repositorio](#) en GitHub)

Proyecto
Final



CODER HOUSE

¡IMPORTANTE!

Los desafíos y entregas de proyecto se deben cargar hasta siete días después de finalizada la clase. Te sugerimos llevarlos al día.

LUNES 16/03 20:30HS

10. Estrategia de contenido para Twitter y LinkedIn

DESAFÍO - EXPIRA EL 23/03/2020

Crear publicaciones para Twitter

👍

HOY 20:30

📁

Tenes tiempo hasta el 23/03/2020

↑

ENTREGAR

Algunas recomendaciones

Este curso tiene un nivel avanzado, lo cual implica un gran desafío. Queremos que tengas éxito y sobre todo, **disfrutes del camino.** Por eso traemos algunas recomendaciones a continuación.

La práctica hace al maestro

A lo largo del curso encontrarás nuevos conceptos cuya dificultad irá incrementando gradualmente. Te sugerimos **ser constante con la práctica**. Organiza en tu semana momentos para repasar y practicar los temas vistos, a fin de corroborar que los hayas comprendido o anotar aquello que necesites consultar.



¡Anímate a investigar!

En clase abordaremos los temas necesarios para que puedas crear un proyecto que cumpla con los requerimientos planteados en la consigna. No obstante, sabemos que el mundo de la programación es inmenso. Te sugerimos **amigarte con la investigación** para encontrar aquello que tu proyecto necesita. Es una buena práctica para tu futuro profesional.



W3schools.com



CODER HOUSE

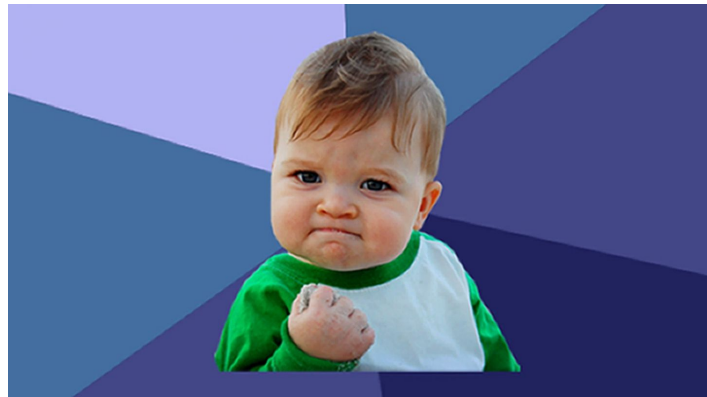
Amígate con los after classes

Además de las clases de asistencia obligatoria, organizaremos **after classes** para repasar y profundizar los temas vistos, así como abrir espacios de consulta. Si bien la asistencia es optativa, te sugerimos que los tengas presentes como recurso para mejorar tu experiencia. En caso de no poder asistir, recuerda que quedarán grabados.



Más vale pájaro en mano que cien volando

Queremos que logres realizar el mejor Proyecto Final posible. Por eso es importante que primero te centres en los requisitos o piezas mínimas solicitados en la consigna, para luego apuntar hacia ideas más ambiciosas. Esto te permitirá **alcanzar las metas paso a paso** y superar tus propios límites.



¡Arrancamos!



Clase 1. Programación Backend

Principios de programación Backend



OBJETIVOS DE LA CLASE

- Entender la diferencia entre programación Front end y Back end
- Comprender las nociones básicas para programar utilizando Javascript y el MERN Stack

CRONOGRAMA DEL CURSO

Clase 1



**Principios de
programación
Backend y Javascript**

Clase 2



**Manejo de Archivos en
Javascript:
programación
sincrónica y
asincrónica**

Clase 3



Servidores Web

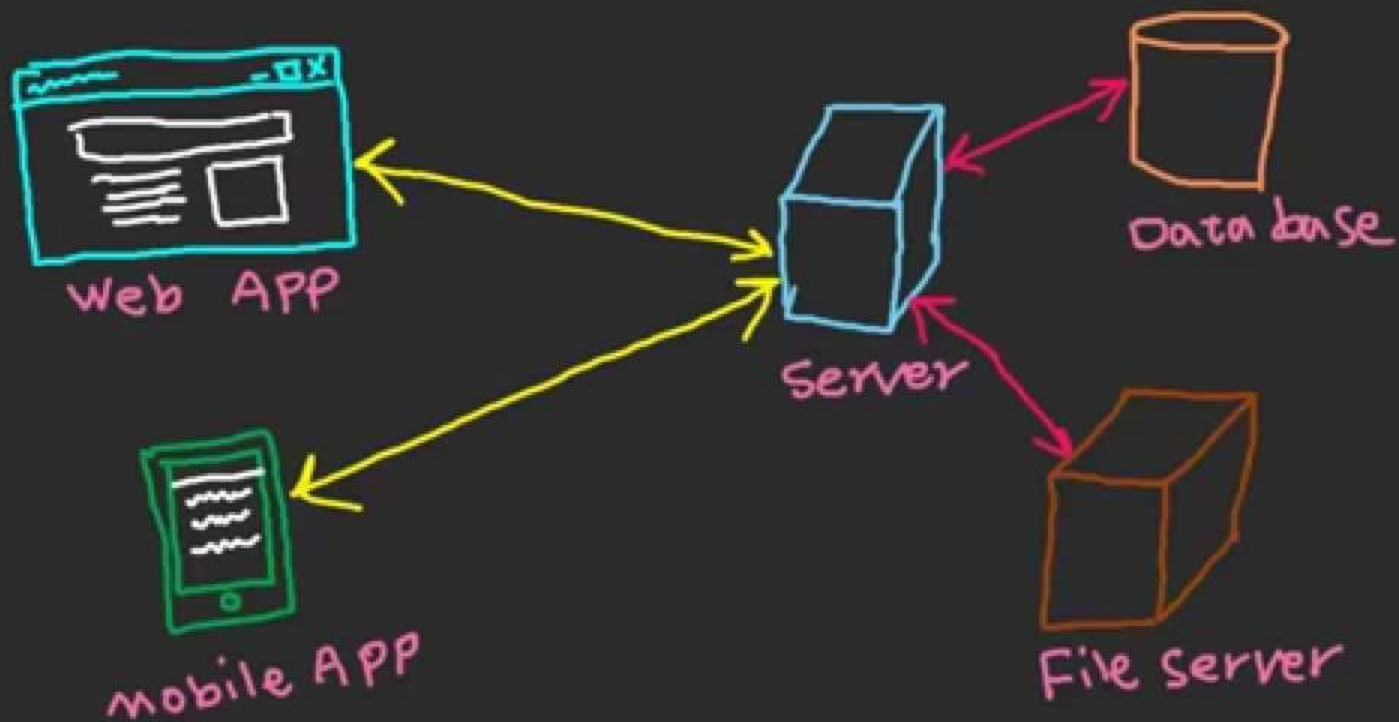
Programación Web

***¿Qué es Front end y Back
end?***

Concepto

El Back end y el Front end son dos partes fundamentales de la programación de una aplicación web. Al **Front end** se lo conoce como el **lado del cliente** y al **Back end** como el **lado del servidor**

Front-End / Back-End



Front End



- Es la parte de una aplicación que **interactúa con los usuarios**
- Conocida como “el lado del cliente”.
- Todo **lo que vemos en la pantalla** cuando accedemos a un sitio web o aplicación: tipos de letra, colores, efectos del mouse, teclado, movimientos, desplazamientos, efectos visuales y otros elementos que permiten navegar dentro de una página web.

Back End



- Aplicaciones que viven en el servidor
- A menudo se le denomina “el **lado del servidor**”.
- Consiste en un servidor, una aplicación y una base de datos.
- Se **toman** los datos, se **procesa** la información y se **envía** al usuario

Visto de otra manera...



MERN Stack

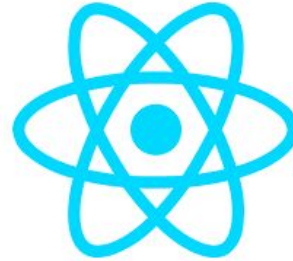
Componentes



MongoDB



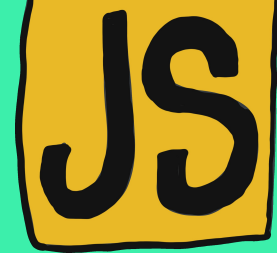
ExpressJS



ReactJS



NodeJS



Concepto

El stack MERN utiliza **JavaScript como único lenguaje**, por ello no tendremos dificultades al familiarizarnos con cualquiera de estas tecnologías, las cuales son MongoDB, Express, React y Node.js.

La ventaja que encontramos al utilizar este stack en específico es que **nos permite profundizar** en un solo lenguaje de programación, logrando así enfocar y reforzar nuestros conocimientos y con ello ser más productivos.

MERN Stack



- **MongoDB** base de datos no relacional
- **ExpressJS** framework para crear servidores en NodeJS
- **ReactJS** librería para desarrollar interfaces de usuario
- **NodeJS** entorno de ejecución de Javascript



MongoDB

MongoDB (del inglés *humongous*, "enorme") es un **sistema de base de datos NoSQL** orientado a documentos y de código abierto. Permite almacenar Big Data gestionando altos volúmenes de información sin degradar su performance en las búsquedas.

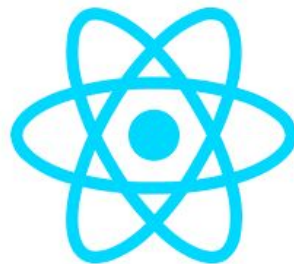
Su diseño brinda **alta escalabilidad y disponibilidad**.



ExpressJS

Express.js es un **framework** para Node.js que sirve para ayudarnos a crear **aplicaciones web en menos tiempo**, ya que nos proporciona funcionalidades como el enrutamiento, opciones para gestionar sesiones y cookies, entre otras funciones

ReactJS

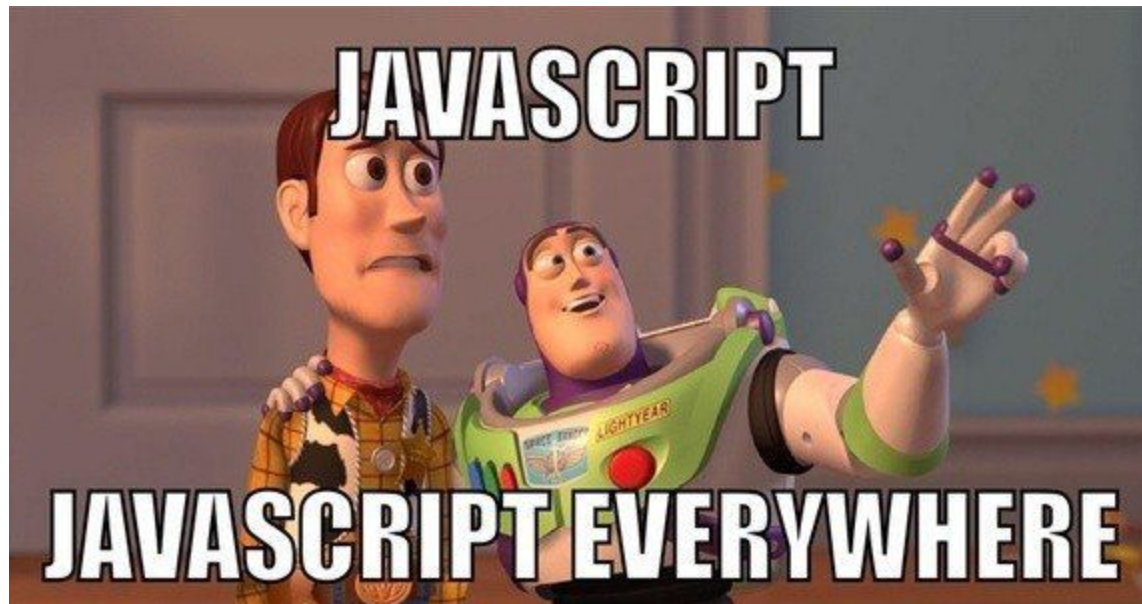


Es una **librería Javascript** de código abierto diseñada para crear **interfaces de usuario** con el objetivo de facilitar el desarrollo de aplicaciones web en una sola página (SPA). Es mantenido por Facebook y la comunidad de software libre.



NodeJS

Es un **entorno en tiempo de ejecución multiplataforma** para la **capa del servidor** (pero no limitándose a ello) basado en JavaScript. Es de código abierto, **asíncrono**, con E/S de datos en una **arquitectura orientada a eventos** y basado en el motor V8 de Google.



Distintas maneras de probar Javascript

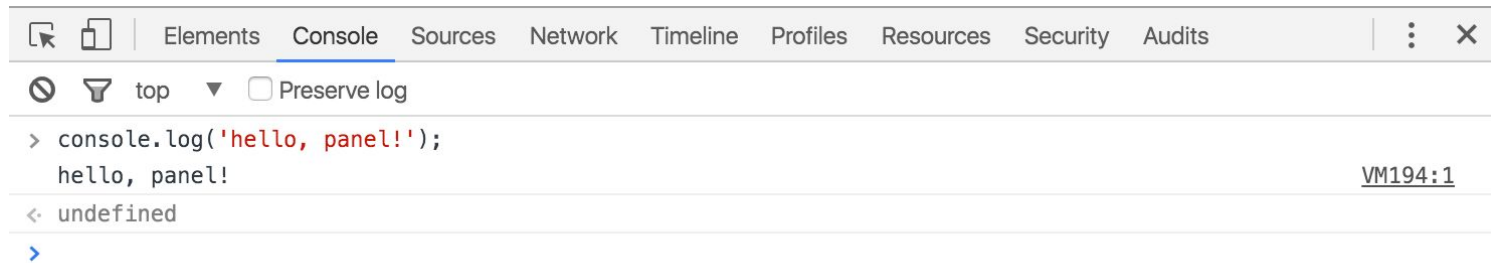
Javascript en el frontend

La Consola Web

Al trabajar en el Front End de una aplicación web, una de las herramientas que más utilizamos es la **Consola Web**

La Consola Web

1. Muestra información asociada con los **logs** de la página web: solicitudes de red, JavaScript, CSS, errores de seguridad y advertencias.
2. También muestra advertencias y mensajes informativos explícitamente generados por Javascript en **tiempo de ejecución** dentro del contexto de la página.
3. Permite **interactuar con la página** ejecutando expresiones Javascript en el contexto de la misma.



La información que se muestra en la consola puede ser extremadamente útil cuando intentamos descubrir cómo resolver un problema.

Javascript Console Object

En JavaScript, *console* es un **objeto** que proporciona acceso a la consola de depuración del navegador.

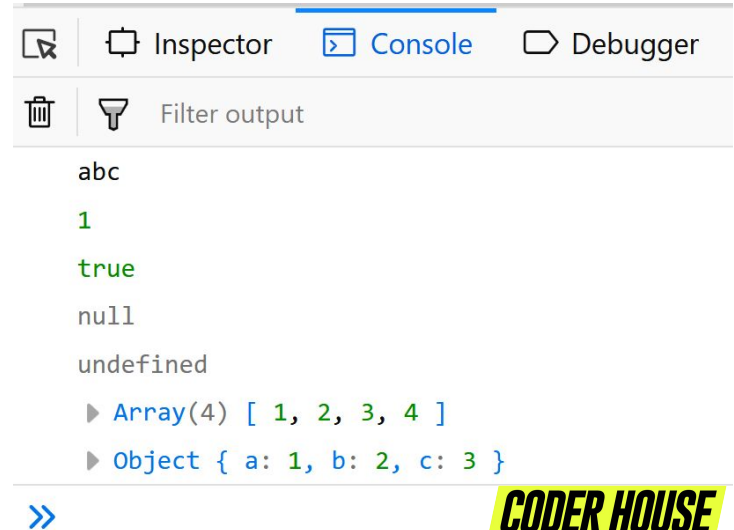
Este objeto nos proporciona varios **métodos**. Veamos los principales:

```
console.log()  
console.error()  
console.warn()  
console.clear()
```

console.log()

Se utiliza para imprimir la salida en la consola. Podemos poner cualquier tipo de dato dentro del log ().

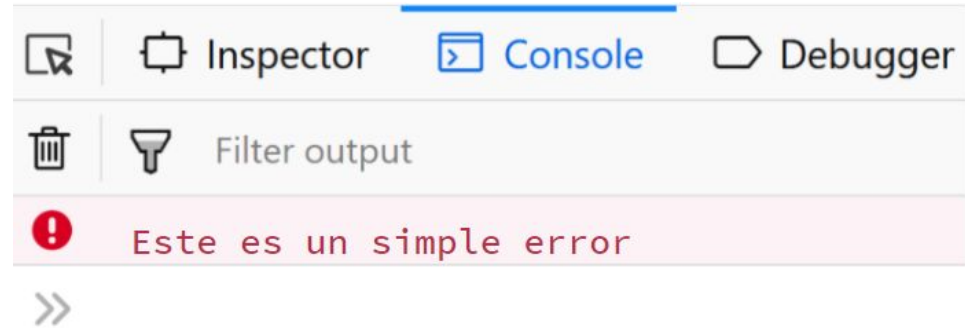
```
// console.log() método
console.log('abc');
console.log(1);
console.log(true);
console.log(null);
console.log(undefined);
console.log([1, 2, 3, 4]); // array
console.log({a:1, b:2, c:3}); // objeto
```



console.error()

Se utiliza para imprimir mensajes de error en la consola. Resulta útil para probar código. Por default, el mensaje se resaltará en rojo.

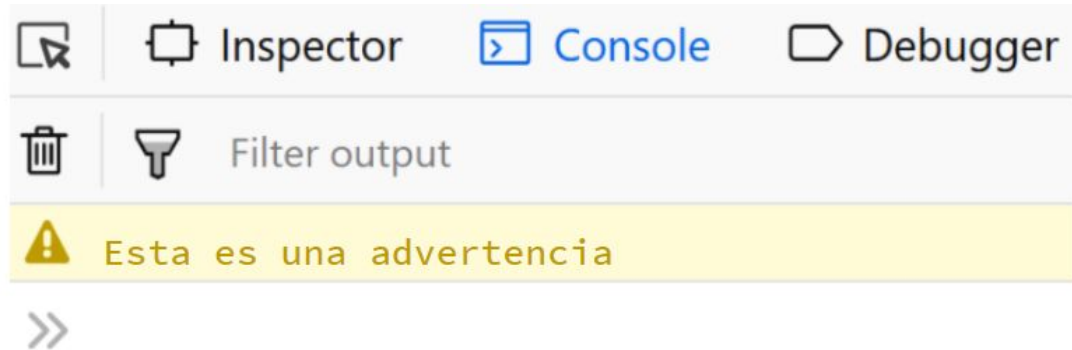
```
// console.error() método  
console.error('Este es un simple  
error');
```



console.warn()

Se utiliza para imprimir mensajes de advertencia en la consola. Por default, el mensaje se resaltará en amarillo.

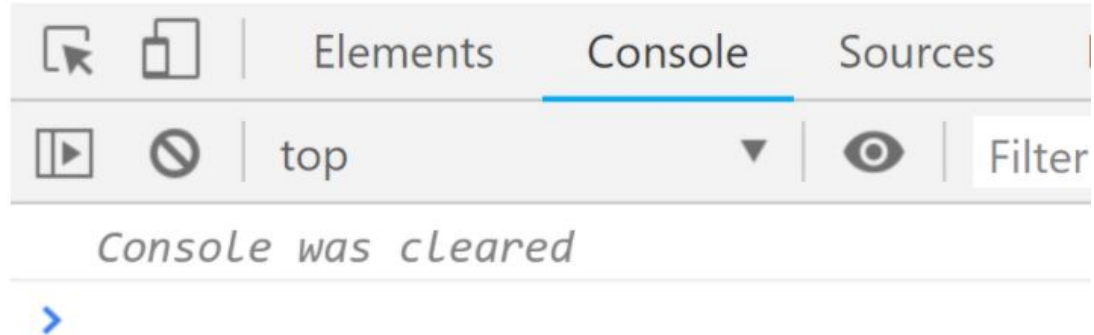
```
// console.error() método  
console.warn('Esta es una  
advertencia');
```



console.clear()

Se utiliza para limpiar la consola. En el caso de Chrome, aparecerá un mensaje dando aviso. En el caso de Firefox, no aparecerá nada.

```
// console.clear() método  
console.clear();
```





Probemos la Consola Web!

Javascript en el backend

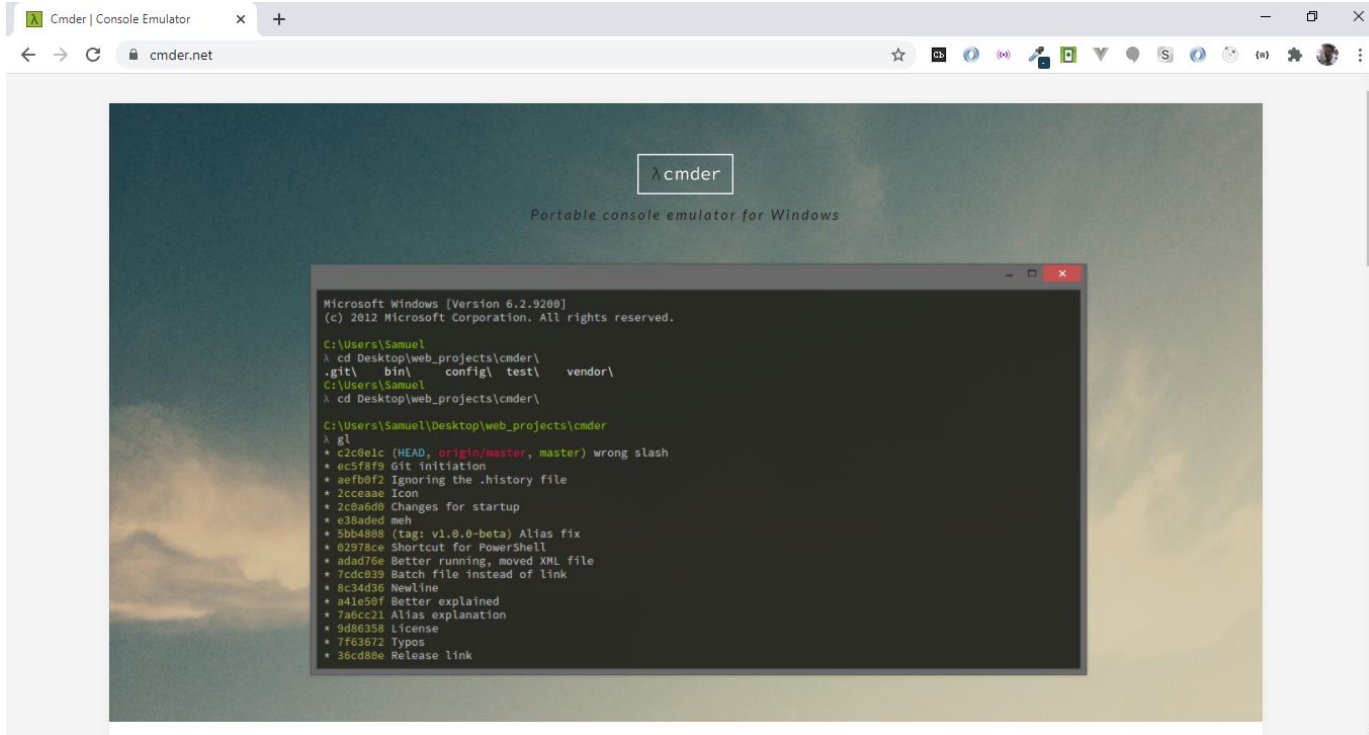
La Consola CLI

Al trabajar en el Back End de una aplicación web, una de las herramientas que más utilizaremos será la **Consola CLI**

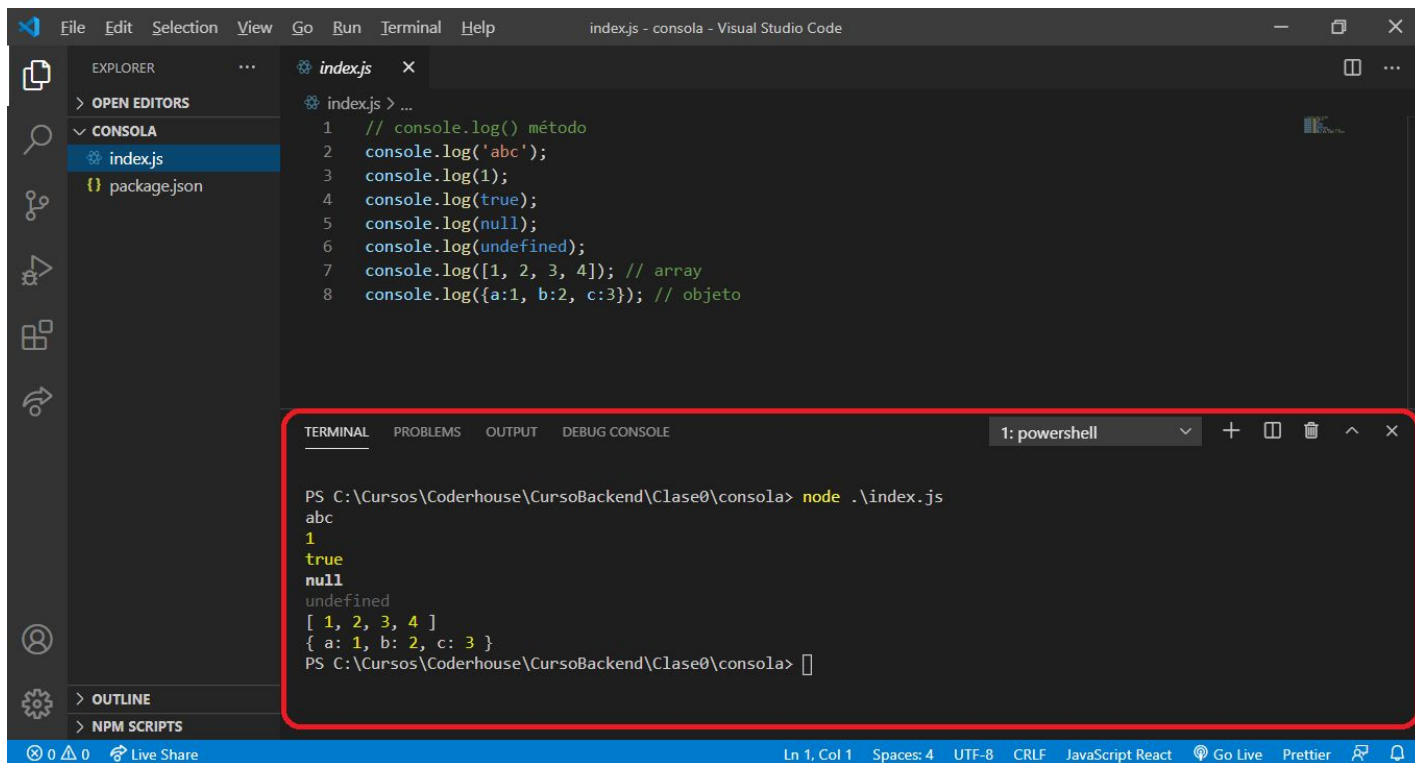
La Consola CLI:

1. Es una interfaz de línea de comandos (**C**ommand **L**ine **I**nterface)
2. Permite ejecutar nuestro programa de servidor
3. Muestra información asociada con los **logs** de los procesos de backend: son mensajes de debug que enviamos desde nuestro programa con el objeto **console**.
4. También muestra advertencias y mensajes informativos explícitamente generados por Javascript en **tiempo de ejecución** dentro del contexto del programa del servidor.

Ejemplo de consola CLI : <https://cmdr.net/>



Ejemplo de consolas CLI : **Terminal** en Visual Studio Code



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left shows a project with files `index.js` and `package.json`. The `index.js` file is open in the editor, containing the following JavaScript code:

```
1 // console.log() método
2 console.log('abc');
3 console.log(1);
4 console.log(true);
5 console.log(null);
6 console.log(undefined);
7 console.log([1, 2, 3, 4]); // array
8 console.log({a:1, b:2, c:3}); // objeto
```

At the bottom, the TERMINAL panel is active, showing a PowerShell session. The command `node .\index.js` has been executed, resulting in the following output:

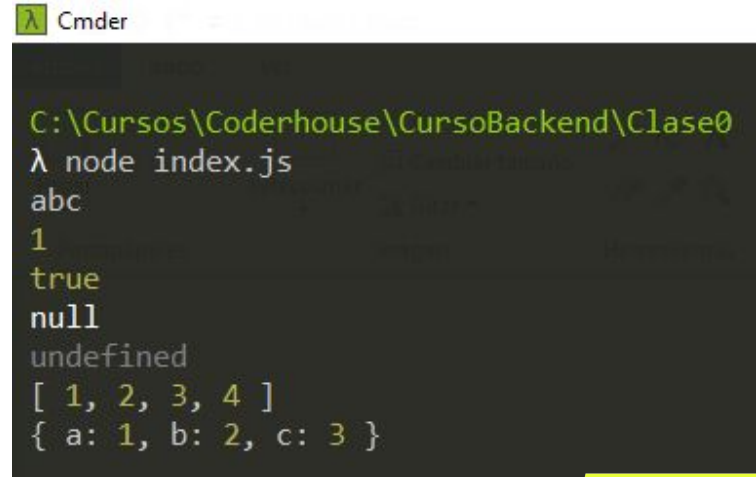
```
PS C:\Cursos\Coderhouse\CursoBackend\Clase0\consola> node .\index.js
abc
1
true
null
undefined
[ 1, 2, 3, 4 ]
{ a: 1, b: 2, c: 3 }
PS C:\Cursos\Coderhouse\CursoBackend\Clase0\consola>
```

The terminal output matches the values logged in the JavaScript file. The terminal window title is `1: powershell`.

console.log() en Backend

Al igual que en Front End, se utiliza para imprimir un resultado, contenido de una variable ó mensaje desde nuestro programa de servidor en la consola CLI.

```
// console.log() método
console.log('abc');
console.log(1);
console.log(true);
console.log(null);
console.log(undefined);
console.log([1, 2, 3, 4]); // array
console.log({a:1, b:2, c:3}); // objeto
```



A screenshot of a terminal window titled 'Cmder'. The terminal shows the directory path 'C:\Cursos\Coderhouse\CursoBackend\Clase0' and the command 'λ node index.js'. Below the command, the output of the console.log() calls is displayed: 'abc', '1', 'true', 'null', 'undefined', '[1, 2, 3, 4]', and '{ a: 1, b: 2, c: 3 }'.

```
λ Cmder
C:\Cursos\Coderhouse\CursoBackend\Clase0
λ node index.js
abc
1
true
null
undefined
[ 1, 2, 3, 4 ]
{ a: 1, b: 2, c: 3 }
```



Probemos la Consola CLI!



BREAK

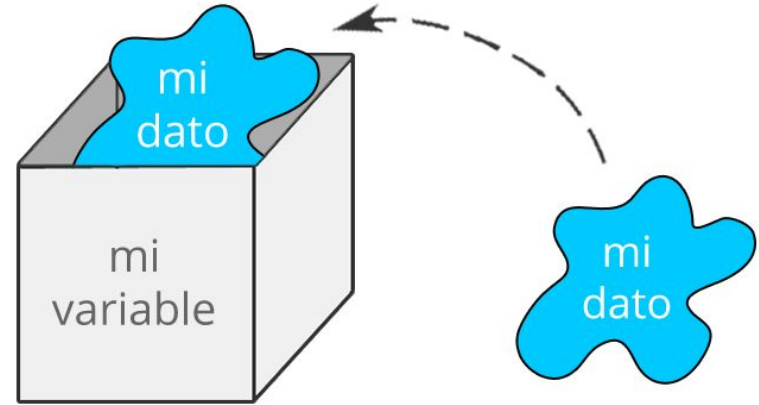
¡5/10 MINUTOS Y VOLVEMOS!

Tipos de datos en Javascript

Variables y tipos de datos

Variable: es un espacio reservado para almacenar un dato que puede ser usado o modificado tantas veces como se desee.

Tipo de dato: es el atributo que especifica la clase de dato que almacena la variable.



Tipos de datos

- **Tipo Primitivos:** Incluyen a las cadenas de texto (String), variables booleanas cuyo valor puede ser true o false (Boolean) y números (Number). Además hay dos tipos primitivos especiales que son Null y Undefined. La copia es por valor.
- **Tipo Objeto:** Incluyen a los objetos (Object), a los arrays (Array) y funciones. La copia es por referencia.

TIPOS DE DATOS EN JAVASCRIPT		NOMBRE	DESCRIPCIÓN
	TIPOS PRIMITIVOS	String	Cadenas de texto
		Number	Valores numéricos
		Boolean	true ó false
		Null	Tipo especial, contiene null
		Undefined	Tipo especial, contiene undefined
	TIPOS OBJETO	Tipos predefinidos de JavaScript	Date (fechas) RegExp (expresiones regulares) Error (datos de error)
		Tipos definidos por el programador / usuario	Funciones simples Clases
		Arrays	Serie de elementos o formación tipo vector o matriz. Lo consideraremos un objeto especial que carece de métodos.
		Objetos especiales	Objeto global
			Objeto prototipo
			Otros

Javascript y ES6

EcmaScript 6

ES6 o EcmaScript 2015, fue una enorme revisión que surgió en el año 2015 y trajo -dentro de varias polémicas- enormes avances en el mundo de la programación JavaScript.

Entre sus mayores innovaciones se encuentra la declaración de variables con **let** y **const**, la introducción de **clases** al lenguaje, y los *template strings*.

Variables en Javascript

Recordemos...

- Una variable es un **contenedor dinámico** que nos permite **almacenar valores**.
- Los valores pueden ser diversos **tipos de datos**, según la variable.
- Tal como lo indica su nombre, el **valor de la variable puede cambiar**, permitiéndonos crear programas que funcionen independientemente del valor de la variable.

Distintas maneras de crear variables en Javascript

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!



Let

Un **bloque** en JavaScript se puede entender como “**lo que queda entre dos llaves**”, ya sean definiciones de funciones o bloques if, while, for y loops similares. Si una variable es declarada con let en el ámbito global o en el de una función, la variable pertenece al ámbito global o al ámbito de la función respectivamente.

```
let i = 0;
function foo() {
  i = 1;
  let j = 2;
  if(true) {
    console.log(i); // 1
    console.log(j); // 2
  }
}
foo();
```

```
function foo() {
  let i = 0;
  if(true) {
    // Sería otra variable i
    // sólo para el bloque if
    let i = 1;
    console.log(i); // 1
  }
  console.log(i); // 0
}
foo();
```

Ejemplo Let

Aquí la variable `i` es global y la variable `j` es local.

Pero si declaramos una variable con **let dentro un bloque**, que **a su vez está dentro de una función**, la **variable pertenece solo a ese bloque**.

```
function foo() {  
  if(true) {  
    let i = 1;  
  }  
  // ReferenceError:  
  // i is not defined  
  console.log(i);  
}  
foo();
```

Ejemplo Let

Fuera del **bloque** donde se declara con `let`, la **variable no** está **definida**.

Const

Al igual que con `let`, el **ámbito** (scope) para una **variable** declarada con **`const`** es el **bloque**.

Sin embargo, **`const`** además **prohíbe la reasignación de valores** (const viene de *constant*).

```
const i = 0;  
  
// TypeError:  
// Assignment to constant variable  
i = 1;
```

Ejemplo Const

Si se intenta reasignar una constante se obtendrá un **error**.

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

Ejemplo Mutabilidad

Por ejemplo, una constante se puede asignar a un objeto con determinadas propiedades. Aunque la constante no se pueda asignar a un nuevo valor, **sí se puede cambiar el valor de sus propiedades.**

```
const user = { name: 'Juan' };  
user.name = 'Manolo';  
console.log(user.name); // Manolo
```

Esto sería posible

```
const user = 'Juan';  
//TypeError: Assignment to constant  
user = 'Manolo';
```

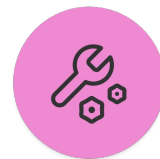
Esto NO sería posible



Datos y Variables

Tiempo aproximado: 5 minutos

Datos y variables



- 1) Definir variables variables que almacenen los siguiente datos:
 - Un nombre: “pepe”
 - Una edad: 25
 - Un precio: \$99.90
 - Los nombres de mis series favoritas: “Dark”, “Mr Robot”, “Castlevania”
 - Mis películas favoritas, en donde cada película detalla su nombre, el año de estreno, y una lista con los nombres de sus protagonistas.
- 2) Mostrar todos esos valores por consola
- 3) Incrementar la edad en 1 y volver a mostrarla
- 4) Agregar una serie a la lista y volver a mostrarla



***¿YA CONOCES LOS BENEFICIOS QUE TIENES
POR SER ESTUDIANTE DE CODERHOUSE?***

CODER HOUSE



***TE INVITAMOS A QUE COMPLEMENTES
LA CLASE CON LOS SIGUIENTES
CODERTIPS***



VIDEOS Y PODCASTS

- [Aprende Programación Web y construye el futuro de nuestra humanidad](#) | **Coderhouse**
- [Desarrollo freelance](#) | **Coderhouse**
- [Desarrollo profesional](#) | **Coderhouse**



VIDEOS Y PODCASTS

- [CoderNews](#) | **Coderhouse**
- [Serie de Branding](#) | **Coderhouse**
- [Serie para Emprendedores](#) | **Coderhouse**
- [Serie Aprende a Usar TikTok](#) | **Coderhouse**
- [Serie Finanzas Personales](#) | **Coderhouse**
- [CoderConf](#) | **Coderhouse**

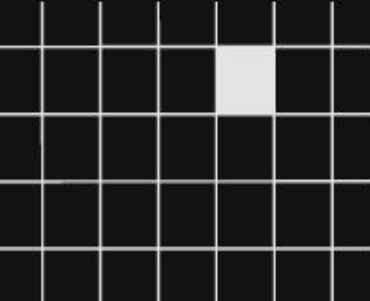
¿PREGUNTAS?





¡MUCHAS GRACIAS!

Resumen de lo visto en clase hoy:

- Conceptos de programación en Javascript
 - Novedades de ES6
- 



OPINA Y VALORA ESTA CLASE

#DEMOCRATIZANDO LA EDUCACIÓN

CODER HOUSE