

# {JS}

**Clase 39**



# ◀ Índice ▶

## Índice

API

---

Fetch

---

Async y Await

---

**API**

# API

<https://jsonplaceholder.typicode.com/>

race

myfakeapi.com

# API

Listar => GET /todos/1

Crear => POST /todos

Modificar => PUT /todos/1

Borrar => DELETE /todos/1

**Fetch**

# Fetch

Fetch será utilizado para pedir o enviar datos a un servidor desde nuestro código javascript



# Fetch

Fetch siempre devuelve una  
promesa resuelta  
(cuidado con esto)

# Fetch

Fetch necesita dos parámetros  
url y un objeto con opciones que  
veremos posteriormente

# Fetch

Podemos usar then, catch y  
todo lo aprendido con las  
promesas

# Fetch

Hagamos nuestro primer fetch

**async y await**

# async y await

La declaración de  
función async define una función  
asíncrona que devuelve un objeto, lo  
cual permite a un programa correr  
una función sin congelar todo la  
compilación

# async y await

Dada que la finalidad de las funciones `async/await` es simplificar el comportamiento del uso síncrono de promesas, se hace más fácil escribir promesas.

# async y await

Nos evita tener que encadenar con `.then`  
si no que usamos un sistema más  
tradicional y lógico ante nuestros ojos



# async y await

“Bloqueamos” hasta que son resueltas  
para continuar dentro de una función

# async y await

Lo que hace await es detener la ejecución y no continuar. Se espera a que se resuelva la promesa, y hasta que no lo haga, no continua. A diferencia del `.then()`, aquí tenemos un código bloqueante.

# async y await

Sobre todo viene a resolver  
el callback hell

# async y await

## Problemos

**◀ Despedida ▶**

Email

**bienvenidosaez@gmail.com**

Instagram

**@bienvenidosaez**

Youtube

**youtube.com/bienvenidosaez**