

# Arquitectura de Software

Tecnología - 2022

A dark blue diagonal gradient bar that starts from the bottom left corner and extends towards the top right corner, covering the lower half of the slide.

# Repasemos

- ¿Qué es javascript?
- Características del lenguaje
- Variables y tipos de datos
- Operaciones
- Funciones

**ECMAScript**

**JS**

# Closures

- Confuso porque su uso es “invisible”
- Identificar cuándo lo estamos usando
- ¿Por qué son importantes tenerlas en mente?

*Una función que utiliza un valor declarado fuera de su contexto*

# Closures



```
let count = 1
function contador() {
  console.log(count)
}
contador() // imprime 1
```



```
function miFuncion() {
  let count = 1
  function contador() {
    console.log(count)
  }
  contador() // imprime 1
}
```

tenemos una función que utiliza un valor que fue declarado fuera de su contexto: **un closure**.

# Sincronismo

- Característica del lenguaje
- Single-thread lenguaje
- Comportamiento asincrónico
  - ¿Cómo logramos esto?: Promises, callbacks, etc.

JavaScript  
Sync vs Async



# Callbacks

- Funciones reciben funciones como parámetro
- ¿Por qué usar funciones callbacks?
- Callback Hell

```
function saludarAlumnos(saludo, callback){  
  var miCita = "Les recomendamos " + saludo;  
  callback(miCita); // 2  
}  
  
function logSaludo(cita){  
  console.log(cita);  
}  
  
saludarAlumnos("repasar javascript!", logSaludo); // 1
```

# Promises

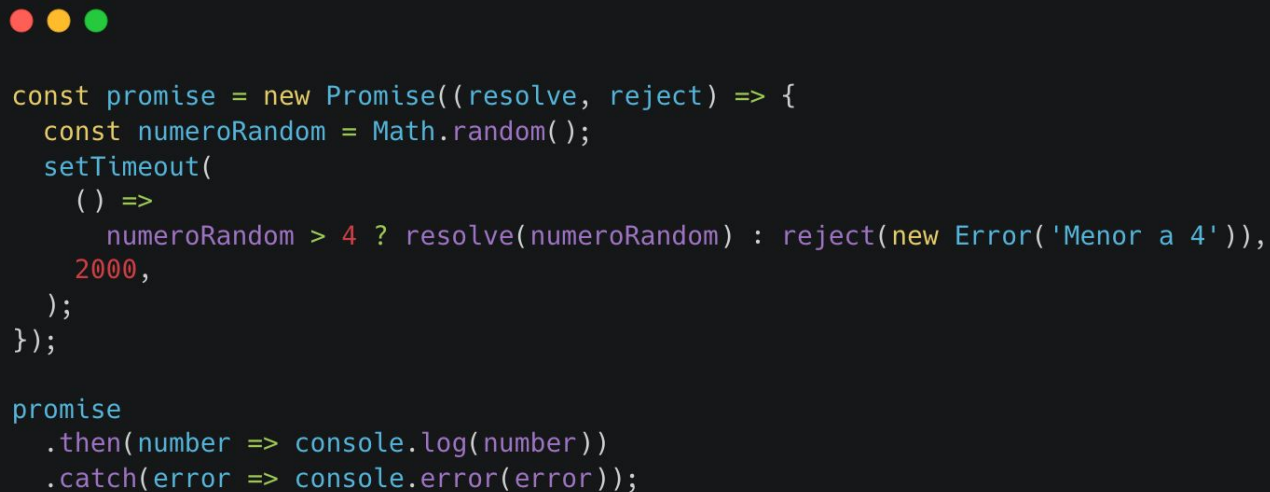
*Una promesa es un objeto que representa un valor que puede que esté disponible «ahora», en un «futuro» o que «nunca» lo esté.*

¿Qué soluciona una promesa?

Estados de una promesa:

- Pendiente
- Resuelta
- Rechazada

# Promises



```
const promise = new Promise((resolve, reject) => {  
  const numeroRandom = Math.random();  
  setTimeout(  
    () =>  
      numeroRandom > 4 ? resolve(numeroRandom) : reject(new Error('Menor a 4')),  
    2000,  
  );  
});  
  
promise  
  .then(number => console.log(number))  
  .catch(error => console.error(error));
```



# Async Await

- Mejor sintaxis
- Mejor comprensión
- ES6

async/await

**JS**

# Async Await

```
const promise = new Promise((resolve, reject) => {
  const numeroRandom = Math.round(Math.random() * 8);
  setTimeout(
    () =>
      numeroRandom > 4 ? resolve(numeroRandom) : reject(new Error("Menor a 4")),
    2000
  );
});

async function ejecutarPromesa() {
  try {
    const result = await promise;
    console.log(result);
  } catch (error) {
    console.log(error);
  }
}f, seed, [])
}
```

# Clases


- Introducidas en ES6
- *Funciones especiales*



```
class Rectangulo {  
  constructor(alto, ancho) {  
    this.alto = alto;  
    this.ancho = ancho;  
  }  
}
```

# Objetos

- Colección de propiedades
- Entidad independiente



```
let miCurso = new Object();  
miCurso.nombre = 'Arquitectura de software';  
miCurso.id = 'N7A';  
miCurso.cantidadAlumnos = 20;
```