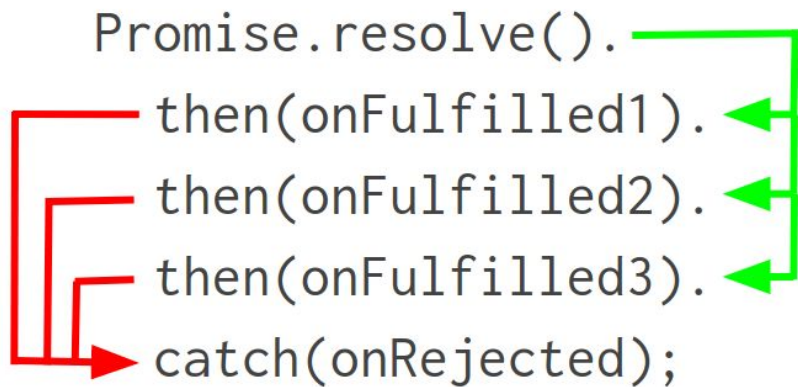


Encadenamiento de Promesas

DEV.F
DESARROLLAMOS(PERSONAS);

dev

```
Promise.resolve().  
  then(onFulfilled1).  
  then(onFulfilled2).  
  then(onFulfilled3).  
  catch(onRejected);
```



Encadement de Promesas

Una necesidad común es el ejecutar dos o más operaciones asíncronas seguidas, donde cada operación posterior se inicia cuando la operación previa tiene éxito, con el resultado del paso previo.

Logramos esto creando una cadena de objetos promises.

Ejemplo de cómo luce un encadement



```
hazAlgo().then(function(resultado) {  
    return hazAlgoMas(resultado);  
})  
.then(function(nuevoResultado) {  
    return hazLaTerceraCosa(nuevoResultado);  
})  
.then(function(resultadoFinal) {  
    console.log('Obtenido el resultado final: ' + resultadoFinal);  
})  
.catch(falloCallback);
```

Resumen de Promesas

- Es usado para interacciones asíncronas
- Se compone de dos aspectos:
 - **Resolve:** Se ejecuta cuando el objetivo de la promesa se efectuó de manera correcta
 - **Reject:** Se ejecuta cuando el objetivo de la promesa ocasionó un error o no se llegó a cumplir.
- Se utiliza la palabra reservada ***“Promise”***
- En las promesas se tienen siempre tres estados:
 - **Pendiente:** Estado inicial de la promesa
 - **Resuelta:** Todo se ejecutó correctamente
 - **Rechazada:** Hubo un problema

async await

- Introducido en **ES06** para mejorar la legibilidad de las promesas
- Nos libramos del encadenamiento de `.then()`
- `async` vuelve a las funciones una promesa y permite usar la palabra reservada `await`
- `await` espera a que las promesas se resuelvan
- Podemos agregar un bloque `try - catch`

```
1  async function getPokemons() {  
2    const response = await fetch(' https://pokeapi.co/api/v2/pokemon');  
3    const pokemons = await response.json();  
4    console.log('pokemons: ', pokemons);  
5  }
```

async await

Example 1.11

```
async function test() {  
  try {  
    const bad = undefined;  
    bad.x;  
    const p = Promise.reject(new Error('Oops!'));  
    await p;  
  } catch (error) {  
    // "cannot read property 'x' of undefined"  
    console.log(err.message);  
  }  
}
```