Curso de JavaScript avanzado

Promesas y asincronía

Trabajando con el objeto Promise

OpenWebinars

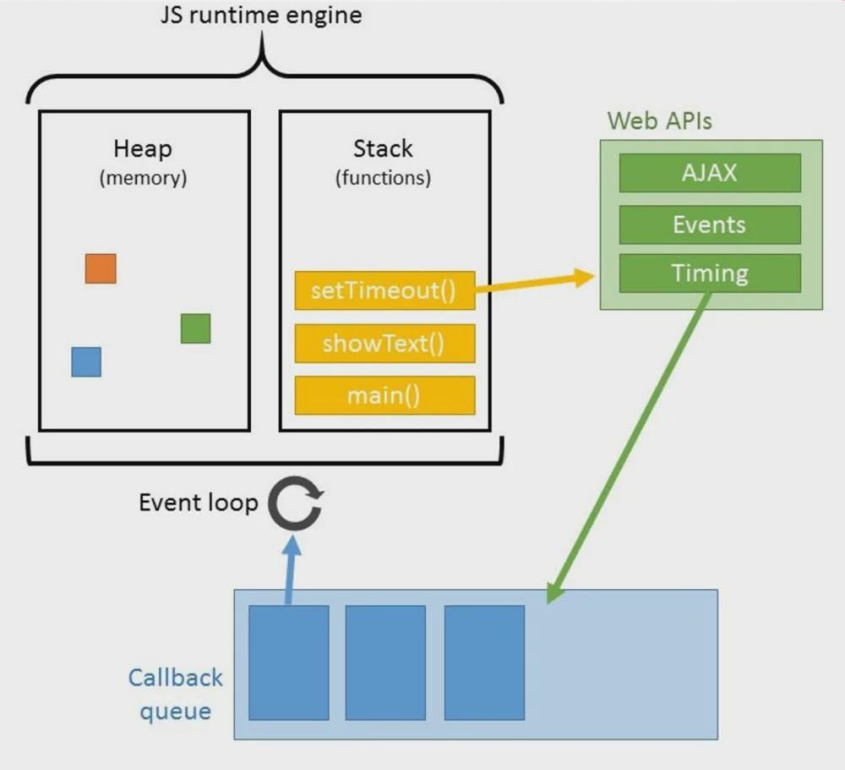
ÍNDICE

- La asincronía en JavaScript
- Estructura de la promesa
- Uso y consideraciones
- Funciones de ayuda

La asincronía en JavaScript

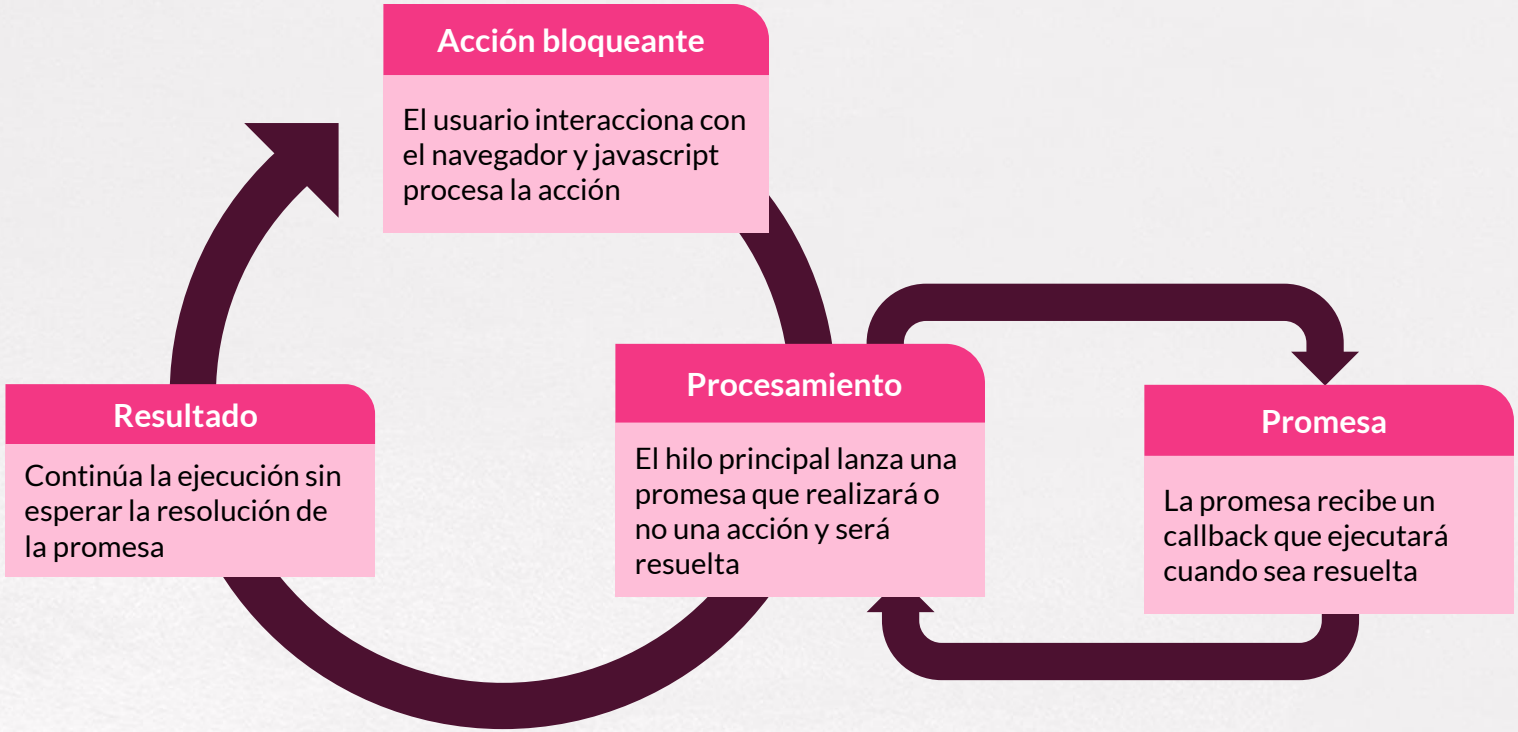
- Javascript es un lenguaje de hilo único, por tanto **es síncrono por definición**
- Esto significa que sólo dispone de **una única pila de llamadas y heap**
- Esto es perjudicial en muchos casos para l**a interacción con el usuario**

La asincronía en Javascript

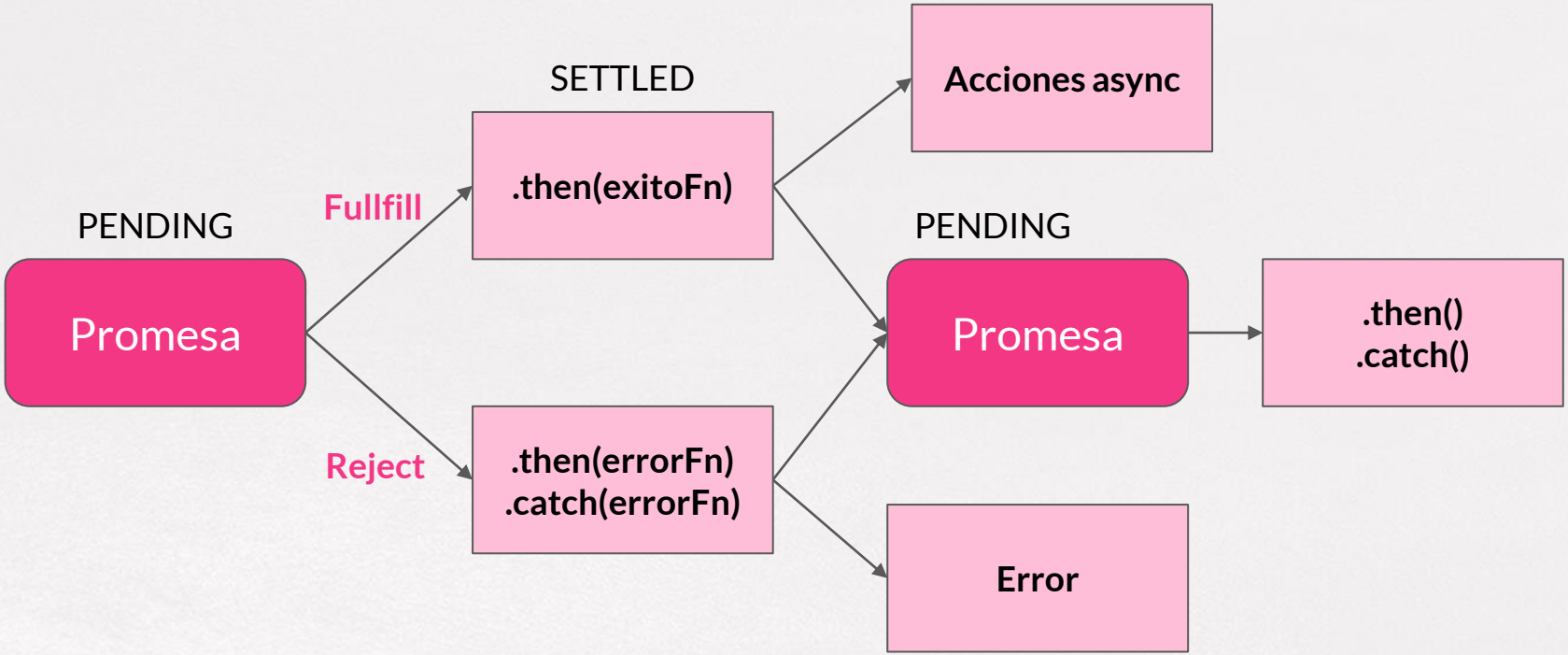


- La asincronía con las web APIs **se resuelve gracias al ciclo de eventos (Event loop)**
- Cada llamada a una API de navegador **genera un callback**
- Los callback son procesados sólo cuando **la pila de llamadas está vacía**

La asincronía en Javascript



Estructura de la promesa



Estructura de la promesa

- Una promesa puede encontrarse en tres posibles estados: **pendiente (pending), cumplida (fulfilled) o rechazada (rejected)**

- Un promesa resuelta en **estado cumplida** ejecutará el manejador o callback **en el bloque then()**

- Un promesa resuelta en **estado rechazada** ejecutará el manejador o callback **en el bloque catch()** donde recibirá como entrada **error** la razón por la que cambió a rechazada

Uso y consideraciones

```
function compruebaNombre(nombre) {
    return new Promise(function(resolve, reject) {
      if (name === 'Pablo') {
          resolve("¡Bien! Te llamas Pablo");
      } else {
          reject("Un momento ¡Tú no eres Pablo!");
      }
    });
  }

compruebaNombre('Pablo')
  .then(response => console.log(response))
  .catch(error => console.log(error))

// Esto imprime '¡Bien! Te llamas Pablo'
// por consola
```

Uso y consideraciones

- Una promesa recibe dos manejadores o callbacks, **el de resolución y el de rechazo**
- Es posible encadenar todos los **then() o catch()** que queramos, ya que una promesa se devuelve a sí misma
- También podemos usar **finally()** que se ejecutará en una promesa **siempre al final e independientemente del estado en el que se encuentre**

Funciones de ayuda

- **Promise.all(iterable) ->** Recibe un iterable de promesas y devuelve una promesa cuyo valor es un array de valores si tiene éxito o la primera razón de fallo si alguna falla
- **Promise.race(iterable) ->** Recibe un iterable de promesas y devuelve una promesa cuyo valor es el de la primera resuelta si tiene éxito o la primera razón de fallo si alguna falla
- **Promise.reject(razón) ->** Devuelve una promesa rechazada con la razón que especifiquemos
- **Promise.resolve(valor)** -> Devuelve una promesa resuelta con el valor que especifiquemos

PARA RESUMIR

- Javascript es un lenguaje de hilo único, y por tanto síncrono, **que gestiona la asincronía mediante el ciclo de eventos y las promesas**
- Una promesa es un objeto que define **una tarea asíncrona que puede ejecutarse en cualquier momento**
- Es posible crear nuestras promesas **para realizar tareas asíncronas**, e incluso es posible gestionar varias promesas al unísono