JavaScript Asincrono

Le Promise

Paolo Caramanica

Sommario

- Le Promise
  - Il conetto di Promise
  - Come è fatta una promise
  - Gli stati di una promise
  - Live Demo
- La chaining e le Promise API
  - Promise chaining
  - Promise API
  - Live Demo
- Approfondimenti
  - Promisification
  - Async/Await
- Fetch
  - Chiamate AJAX con fetch
  - Live Demo

Le Promise

Introduzione e primi concetti

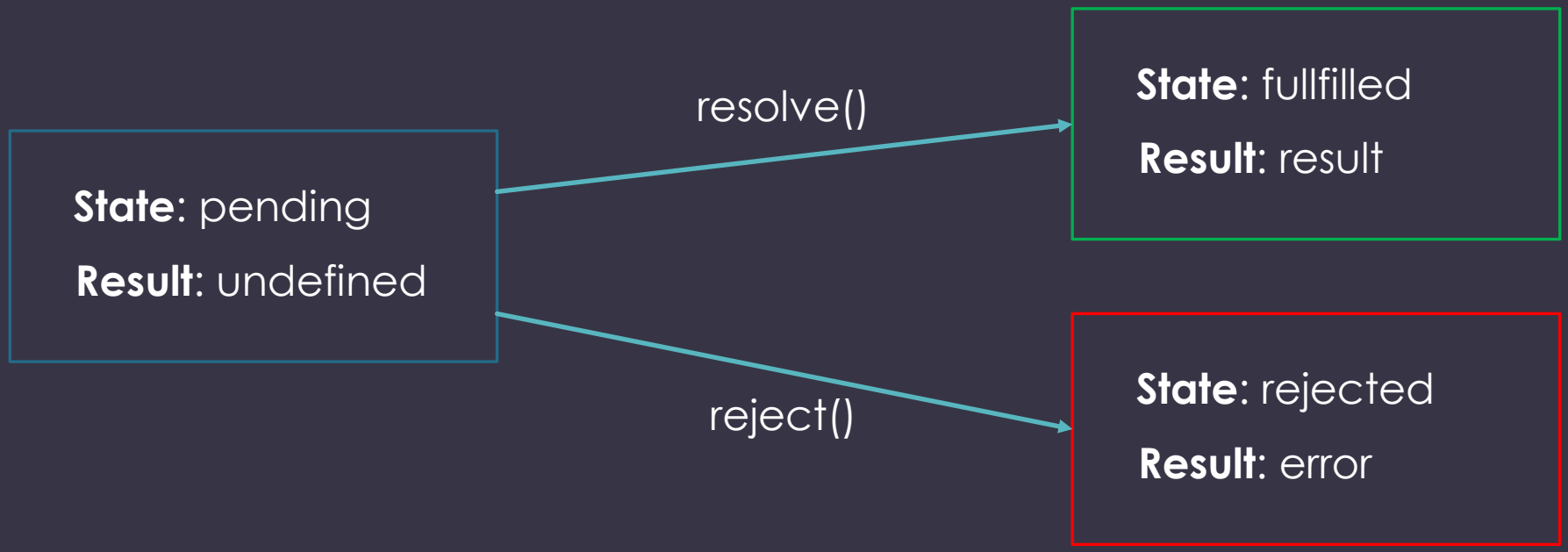
Il concetto di promise

- Una funzione asincrona:
  - Non restituisce un risultato con return.
  - Quando ha terminato l'elaborazione, chiama una funzione di calllback, a cui passa il risultato.
- Con l'utilizzo delle promise, una funzione asincrona:
  - Restituisce, in modo sincrono, un oggetto (la promise), che può essere pensato come un placeholder per il risultato.
  - Tale oggetto è inizialmente in uno stato di **pending**.
  - Appena terminata l'elaborazione asincrona, cambierà stato e conterrà il risultato.

Come è fatta una promise

- E' un oggetto, il cui costruttore accetta come parametro una funzione (**executor**).
- L'executor, a sua volta, accetta come parametri altre due funzioni, denominate **resolve** e **reject.**
- L'executor esegue qualche elaborazione (normalmente asincrona):
  - Se l'elaborazione termina con successo, chiama la funzione resolve passandole come parametro il risultato (lo stato della promise cambia in **fullfilled**).
  - Se l'elaborazione termina con un errore, chiama la funzione reject, passandole come parametro l'errore stesso (lo stato della promise cambia in **rejected**).
- L'oggetto viene creato in modo sincrono e mette a disposizione dei metodi (**then**, **catch**, **finally**) per indicare le funzioni da eseguire ai cambiamenti di stato.

Gli stati di una promise



Live demo

Chaining e Promise API

I vantaggi delle promise

Promise chaining

- Se abbiamo più funzioni asincrone da chiamare in sequenza:
  - La prima restituisce una promise.
  - Il then accetta come parametro la funzione da chiamare appena risolta e restituisce ancora una promise.
  - Il then successivo accetta come parametro la funzione da chiamare appena risolta e restituisce un'ulteriore promise.
  - …
- Ogni then restituisce una promise, sulla quale si può di nuovo chiamate then.
- Il codice è più lineare e si evita la callback hell.

Promise API

- La classe Promise mette a disposizione dei metodi statici, chiamati comunemente **Promise API**:
  - **Promise.all**: accetta come argomento un array di promise e ritorna una nuova promise, la quale
    - Viene risolta se tutte le promise nell'array sono risolte.
    - Viene rigettata se almeno una di esse viene rigettata.
  - **Promise.an**y: accetta come argomento un array di promise e ritorna una nuova promise, la quale
    - Viene risolta non appena una delle promise viene risolta.
    - Viene rigettata se tutte le promise sono rigettate.

Live demo

Approfondimenti

Ulteriori concetti sulle promise

Promisification

- Data una funzione asincrona implementata tramite le callback, può essere utile averne un'implementazione equivalente con le promise:
  - Questo richiede di modificarla o riscriverla.
  - Per funzioni complesse, può essere oneroso.
- Grazie alla **promisification**:
  - La funzione non viene modificata.
  - Si crea una seconda funzione che utilizza quella data e che restituisce una promise.

Async/Await

- **async**:
  - Usata nella definizione di una funzione, fa in modo che questa restituisca sempre una promise.
- **await**:
  - Mette 'in pausa' l'esecuzione di una funzione, finché una promise non sia stata risolta (o rigettata).
  - Può essere usata solo all'interno di funzioni dichiarate con async.

Live demo

Fetch

Utilizzo delle promise nelle chiamate AJAX

Chiamate AJAX con fetch

- **La funzione fetch:**
  - Effettua una richiesta di rete in modo asincrono e restituisce una promise.
  - Se viene risolta, il risultato è sotto forma di oggetto response, che mette a disposizione metodi utili (ad esempio per la conversione in JSON).
  - Permette di effettuare chiamate AJAX, avvantaggiarsi della chaining ed evitando così la callback hell.

Live demo

Contatti

Email: paolocaramanica@gmail.com

Linkedin: https://it.linkedin.com/in/paolo-caramanica-436942a/it-it

Facebook: https://it-it.facebook.com/paolo.caramanica

Twitter: https://twitter.com/PaoloCaramanica

GitHub: https://github.com/paolocaramanica

Q&A