Callback Hell and Promises

Jogesh K. Muppala

THE DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

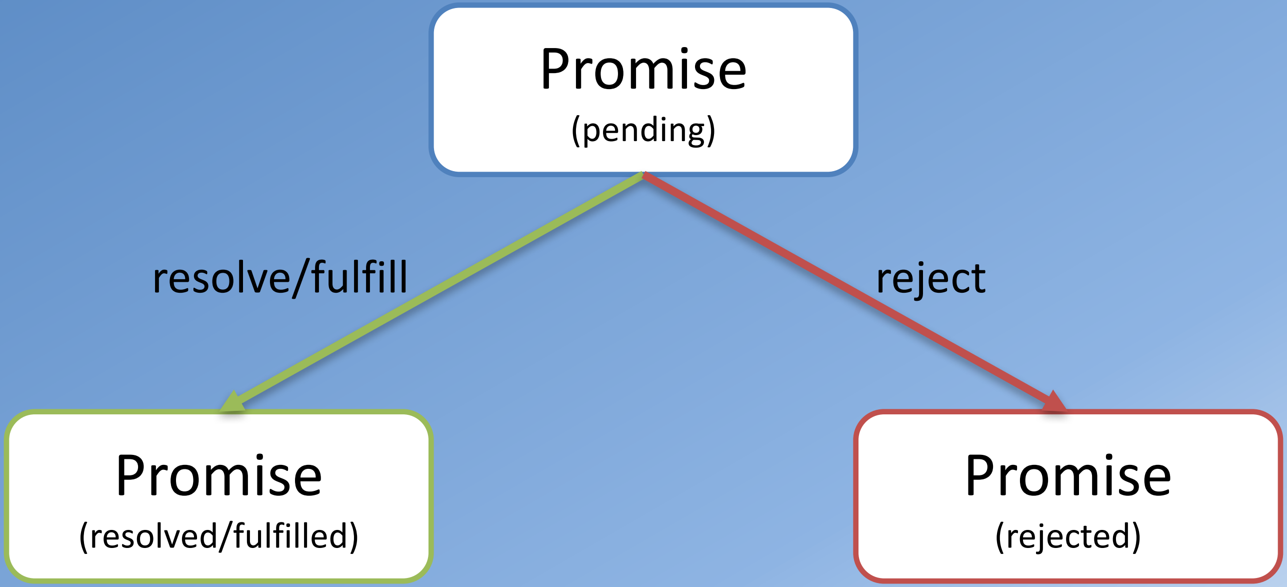
Callback Hell

- Heavily nested callback code
  - Results from our tendency to write code top to bottom
  - Pyramid of doom
- Can use promises to tame it
  - Tries to preserve the top-down appearance of the code

Promise

- Promise is a mechanism that supports asynchronous computation
- Proxy for a value not necessarily known when the promise is created:
  - It represents a value that may be available now, or in the future, or never

Promise



```
new Promise ( function (resolve, reject) { . . . } );
```

Why Promises?

- Solves the callback hell problem
- Promises can be chained
- Can immediately return:
  - Promise.resolve(result)
  - Promise.reject(error)

Consuming Promises

- Consumers of promise are notified of the fulfillment or rejection of the promise
  - Register the callbacks to handle fulfillment and rejection with the .then()
    - can be chained
  - Use .catch() for handling errors
- Example

```
promise
        .then( () => { })
        .catch( () => { });
```