

Promises

Jogesh K. Muppala



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

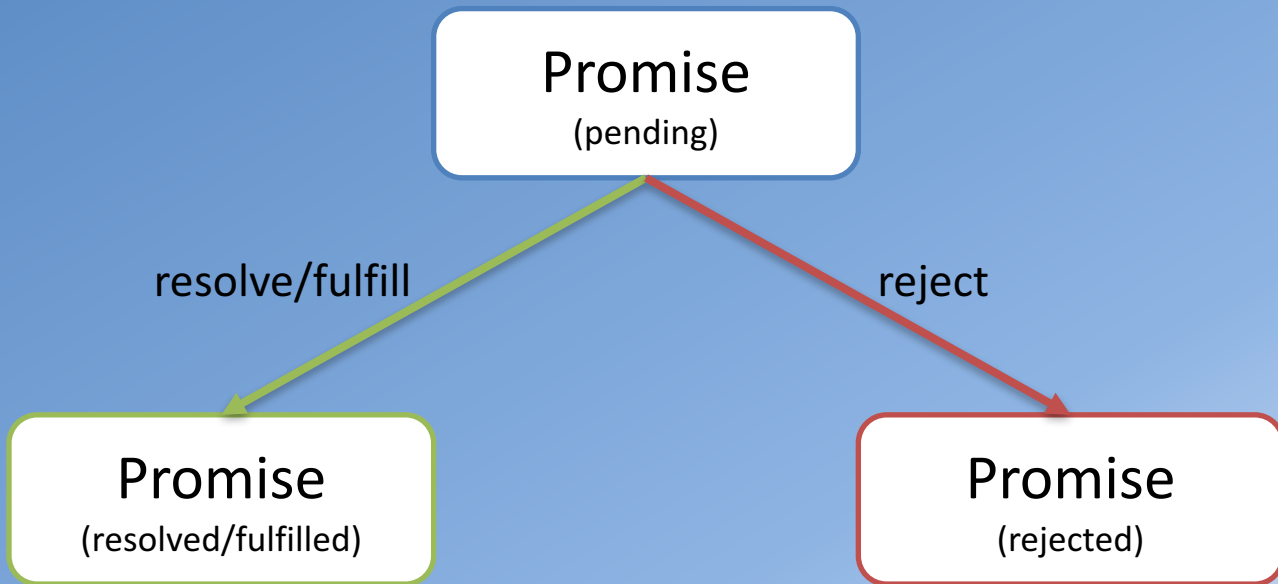


香港科技大學
THE HONG KONG UNIVERSITY OF
SCIENCE AND TECHNOLOGY

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) { . . . } );
```

Promise Example

```
...
getDishes()
    .then ( function(result) {
    })
    .catch ( function(error) {
    });

getDishes(): Promise<Dish[]> {
    return new Promise (
        function(resolve, reject) {
            // do something
            if (successful) {
                resolve(result);
            }
            else {
                reject(error);
            }
        }
    );
}
```

The diagram illustrates the internal structure of a Promise function. It shows how the `getDishes()` method call is mapped to the `return new Promise` statement. The `.then` method is mapped to the `if (successful)` block, and the `.catch` method is mapped to the `else` block. The `resolve(result)` and `reject(error)` calls are shown as the actions taken within these blocks.

Why Promises?

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