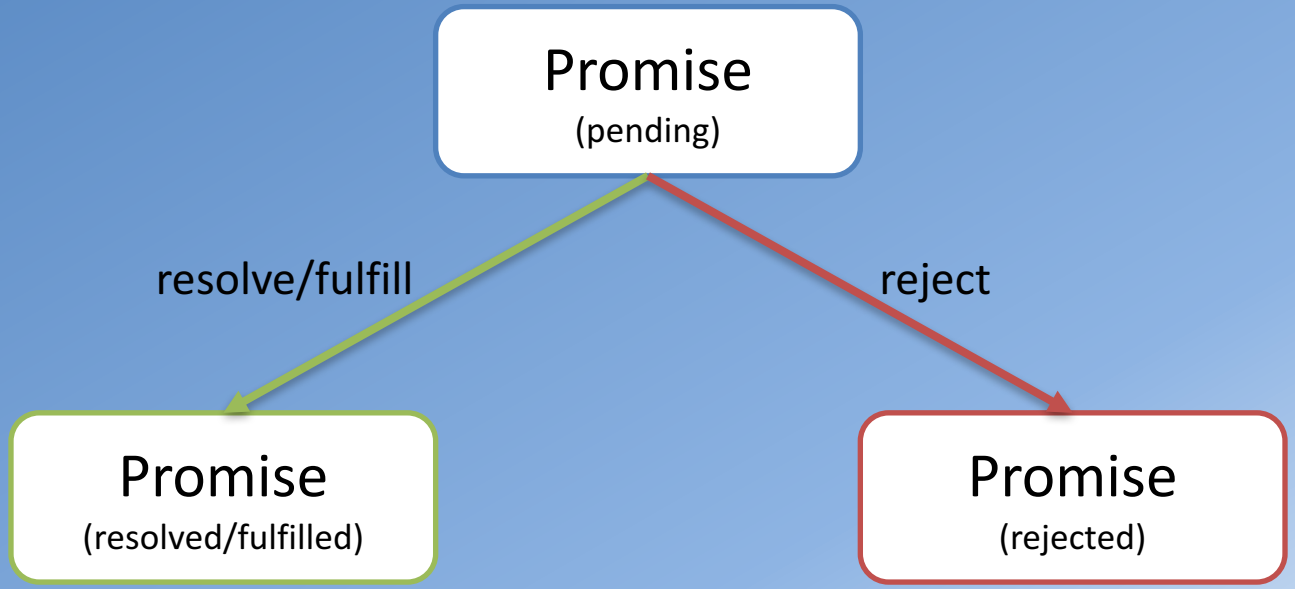
# Promise

- 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);
        }
    });
}
```

# Why Promises?

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