The core idea behind promises is that a promise represents the result of an asynchronous operation. A promise is in one of three different states:

* pending - The initial state of a promise.
* fulfilled - The state of a promise representing a successful operation.
* rejected - The state of a promise representing a failed operation.

Once a promise is fulfilled or rejected, it is immutable (i.e. it can never change again).

Constructing a promise

Once all of the APIs return promises, it should be relatively rare that you need to construct one by hand. In the meantime, we need a way to polyfill existing APIs. For example:

**function** readFile(filename, enc){

**return** **new** Promise(**function** (fulfill, reject){

fs.readFile(filename, enc, **function** (err, res){

**if** (err) reject(err);

**else** fulfill(res);

});

});

}

We use new Promise to construct the promise.

We give the constructor a factory function which does the actual work.

This function is called immediately with two arguments. The first argument fulfills the promise and the second argument rejects the promise. Once the operation has completed, we call the appropriate function.

A promise can be:

* **fulfilled** - The action relating to the promise succeeded
* **rejected** - The action relating to the promise failed
* **pending** - Hasn't fulfilled or rejected yet
* **settled** - Has fulfilled or rejected