I promise you, you won't regret

By: Simon Escobar
Twitter: @sescob27
Github: sescobb27

```
fs.readdir(source, function (err, files) {
  if (err) {
    console.log('Error finding files: ' + err)
  } else {
    files.forEach(function (filename, fileIndex) {
      console.log(filename)
      gm(source + filename).size(function (err, values) {
        if (err) {
          console.log('Error identifying file size: ' + err)
        } else {
          console.log(filename + ' : ' + values)
          aspect = (values.width / values.height)
          widths.forEach(function (width, widthIndex) {
            height = Math.round(width / aspect)
            console.log('resizing ' + filename + 'to ' + height + 'x' + height)
            this.resize(width, height).write(dest + 'w' + width + '_' + filename, function(err) {
              if (err) console.log('Error writing file: ' + err)
            })
          }.bind(this))
        }
      })
    })
  }
})
```

Task

Fetch all E&C collections and then fetch all their related products, how would you do that?(Async)

Sa

```
{
  data: [
    {
      "type": "collections",
      "id": "56c4c6a3d800f2110034f326",
      "attributes": {
        "code": "SP14",
        "title": "Spring 2014",
      },
      "relationships": {
        "products": {
          "data": [{
            "type": "products",
            "id": "56c4c6a3d800f2110034f339"
          }, {
            "type": "products",
            "id": "56c4c6a3d800f2110034f333"
          }, {
            "type": "products",
            "id": "56c4c6a3d800f2110034f340"
          }],
          "links": {
            "self": "/collections/56c4c6a3d800f2110034f326/relationships/products"
          }
        }
      }
    }
  ]
}
```

Show async solution

Promises

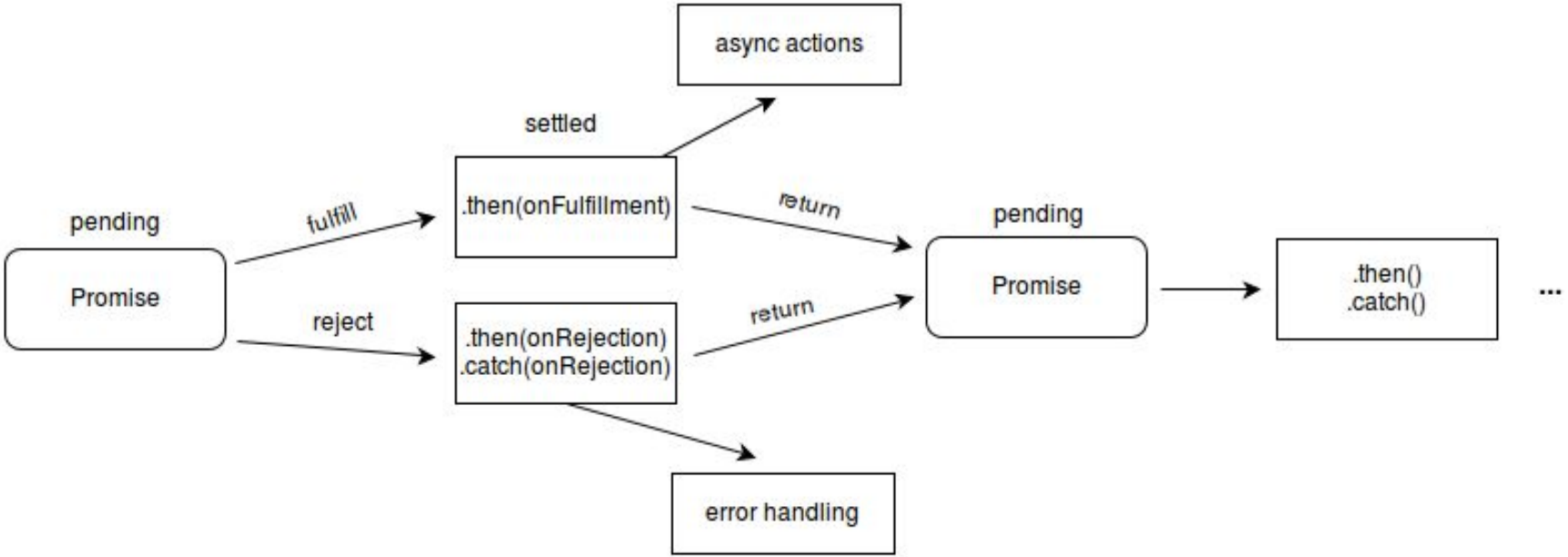
The Promise object is used for deferred and asynchronous computations. A Promise represents an operation that hasn't completed yet, but is expected in the future.

States

A Promise is in one of these states:

- pending: initial state, not fulfilled or rejected.
- fulfilled: meaning that the operation completed successfully.
- rejected: meaning that the operation failed.

Promise Flow

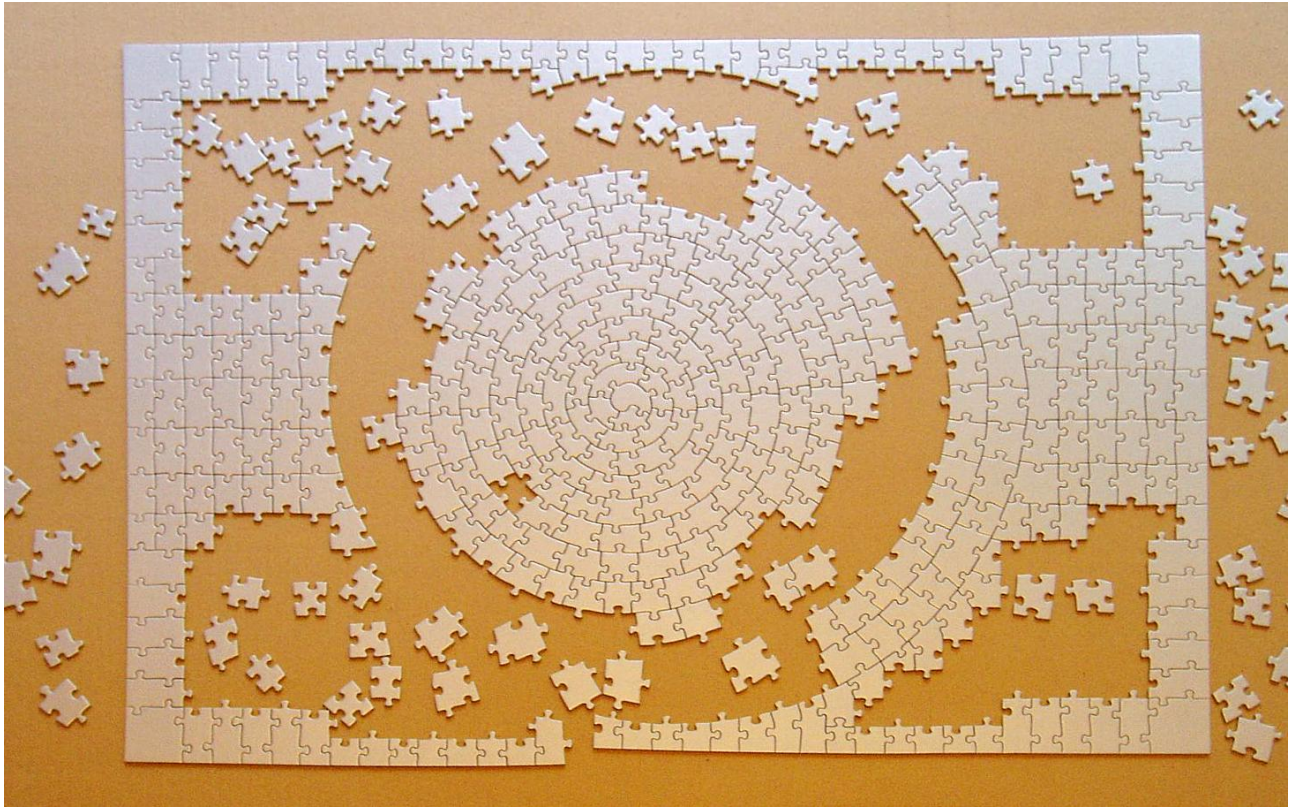


Methods

```
Promise.catch(onRejected)

Promise.then(onFulfilled, onRejected)
```

So, What's the special thing?



Let's see

```
'use strict';

let promiseNumber = new Promise((resolve, reject) => {
  resolve(5);
});

let promiseString = new Promise((resolve, reject) => {
  resolve("Hello World");
});
```

NOTE

It is ok to return non-promise values from non-async code.

```
promiseNumber
  .then((number) => {
    console.log(number);
    return promiseString;
  })
  .then((str) => {
    console.log(str);
  });
```

```
let promiseError = new Promise((resolve, reject) => {
  reject(new Error("Validation Error"));
});

promiseNumber
  .then((number) => {
    console.log(number);
    return promiseError;
  })
  .then(() => {
    return promiseString;
  })
  .then((str) => {
    console.log(str);
  })
  .catch((error) => {
    console.error(error);
  });
```

```
Promise.all([
    promiseNumber,
    promiseString
  ])
  .then((result) => {
    console.log('promiseNumber: ', result[0]);
    console.log('promiseString: ', result[1]);
  });
```

Gotcha: Make sure you return a promise if the code is async.

```
asyncWithPromise1()
  .then(() => {                    // happens first
    return asyncWithPromise2();    // happens after the first finishes
  })
  .then(() => {
    return asyncWithPromise3();    // happens after the second finishes
  })
  .then(() => {
    console.log('finish');
  });
```

Show Promise solution

What!!!!!???

They aren’t pretty enough but they are pispitas

A+ Promises

An open standard for sound, interoperable JavaScript promises—by implementers, for implementers.

https://github.com/promises-aplus/promises-spec

Libs

https://github.com/kriskowal/q

https://github.com/tildeio/rsvp.js

https://github.com/petkaantonov/bluebird

Why a lib? (Q.js)

```
Q.allSettled(promises)
```

```
Q.any(promises)
```

```
Q.fcall(() => 5)
```

```
Q.defer()
```

```
And more ... much more
```

Elizabeth & Clarke - Real Example

https://gist.github.com/sescobb27/b4e567c9411c8d1b1809

https://gist.github.com/sescobb27/822594e2ef51f3920110

Tips

Never broke the contract (always return the same)

Always handle the onReject section

Always prefer composable promises over a very large one

Promises works both in browser and server-side (RSVP - Ember.js) (Q - Angular.js)

I prefer to wrap async behaviour and return a promise instead of using callbacks

THANKS

Let me know if i broke my promise.