API Reference

by Forbes Lindesay (with many examples taken from MDN (https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Global_Objects/Promise))

Methods

Promise.all(iterable)

Returns a Promise that waits for all promises in the `iterable` to be fulfilled and is then fulfilled with an array of those resulting values (in the same order as the input).

Show Example
Show Polyfill

Promise.denodeify(fn, length)

Non Standard

Some promise implementations provide a `.denodeify` method to make it easier to ineroperate with node.js code. It will add a `callback` to any calls to the function, and use that to fullfill or reject the promise.

If the function returns a Promise, the state of that Promise will be used instead of the callback.

If `length` is specified and more arguments are passed than `length`, the remaining arguments will not be passed down to fn.

Show Example
Show Polyfill

Promise.race(iterable)

Returns a promise that resolves or rejects as soon as any of the promises in `iterable` have been resolved or rejected (with the corresponding reason or value).

Show Example

Show Polyfill

Promise.reject(reason)

Returns a promise that is rejected with the given `reason`.

Show Example
Show Polyfill

Promise.resolve(value)

Returns a promise that is resolved with the given `value`.

If the `value` is a promise, then it is unwrapped so that the resulting promise adopts the state of the promise passed in as `value`. This is useful for converting promises created by other libraries.

Show Example
Show Polyfill

Prototype Methods

Promise.prototype.catch(onRejected)

Equivalent to calling `Promise.prototype.then(undefined, onRejected)`

Show Example
Show Polyfill

Promise.prototype.done(onFulfilled, onRejected)

Non Standard

Calls `onFulfilled` or `onRejected` with the fulfillment value or rejection reason of the promise (as appropriate).

Unlike `Promise.prototype.then` it does not return a Promise. It will also throw any errors that occur in the next tick, so they are not silenced. In node.js they will then crash your process (so it can be restarted in a clean state). In browsers, this will cause the error to be properly logged.

Note that `promise.done` has not been standardised. It is supported by most major promise libraries though, and is useful for avoiding silencing errors by accident. I recommend using it with the following polyfill (#[a(href="/polyfills/promise-done-" + versions.promise + ".min.js") minified] / #[a(href="/polyfills/promise-done-" + versions.promise + ".js") unminified]):

```
<script src="https://www.promisejs.org/polyfills/promise-done-#{versions.promise}.min.js"></script>
```

Show Example
Show Polyfill

Promise.prototype.finally(onResolved)

Non Standard

Some promise libraries implement a (non-standard) `.finally` method. It takes a function, which it calls whenever the promise is fulfilled or rejected. It can be pollyfilled with:

Show Example
Show Polyfill

Promise.prototype.nodeify(callback, ctx)

Non Standard

Some promise implementations provide a `.nodeify` method to make it easier to ineroperate with node.js code. If the `callback` parameter is not a function, it simply returns the promise without any changes. If the callback is a function, it is called and `undefined` is returned.

Show Example
Show Polyfill

Promise.prototype.then(onFulfilled, onRejected)

Calls `onFulfilled` or `onRejected` with the fulfillment value or rejection reason of the promise (as appropriate) and returns a new promise resolving to the return value of the called handler.

If the handler throws an error, the returned Promise will be rejected with that error.

If the `onFulfilled` handler is not a function, it defaults to the identify function (i.e. `function (value) { return value; }`).

If the `onRejected` handler is not a function, it defaults to a function that always throws (i.e. `function (reason) { throw reason; }`).

Show Example

Further Reading

- Promises Introduction (/) - explains why Promises are necessary and why we can't just use callbacks.
- Patterns (/patterns/) - patterns of promise use, introducing lots of helper methods that will save you time.
- MDN (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise) - The mozilla developer network has great documentation on promises.

← introduction (/)

patterns → (/patterns/)

Developed by @ForbesLindesay (http://www.forbeslindesay.co.uk)
Can you make this better? Please fork it on GitHub (https://github.com/ForbesLindesay/promisejs.org)