p-map

> *Map over promises concurrently*

Useful when you need to run promise-returning & async functions multiple times with different inputs concurrently.

Install

```
$ npm install p-map
```

Usage

```
const pMap = require('p-map');
const got = require('got');

const sites = [
    getWebsiteFromUsername('https://sindresorhus'), //=> Promise
    'https://ava.li',
    'https://github.com'
];

(async () => {
    const mapper = async site => {
        const {requestUrl} = await got.head(site);
        return requestUrl;
    };

     const result = await pMap(sites, mapper, {concurrency: 2});

    console.log(result);
    //=> ['https://sindresorhus.com/', 'https://ava.li/', 'https://github.com/']
})();
```

API

pMap(input, mapper, options?)

Returns a `Promise` that is fulfilled when all promises in `input` and ones returned from `mapper` are fulfilled, or rejects if any of the promises reject. The fulfilled value is an `Array` of the fulfilled values returned from `mapper` in `input` order.

input

Type: `Iterable<Promise | unknown>`

Iterated over concurrently in the `mapper` function.

mapper(element, index)

Type: `Function`

Expected to return a `Promise` or value.

options

Type: `object`

concurrency

Type: `number` (Integer)
Default: `Infinity`
Minimum: `1`

Number of concurrently pending promises returned by `mapper`.

stopOnError

Type: `boolean`
Default: `true`

When set to `false`, instead of stopping when a promise rejects, it will wait for all the promises to settle and then reject with an aggregated error containing all the errors from the rejected promises.

p-map for enterprise

Available as part of the Tidelift Subscription.

The maintainers of p-map and thousands of other packages are working with Tidelift to deliver commercial support and maintenance for the open source dependencies you use to build your applications. Save time, reduce risk, and improve code health, while paying the maintainers of the exact dependencies you use. Learn more.

Related

- p-all - Run promise-returning & async functions concurrently with optional limited concurrency
- p-filter - Filter promises concurrently
- p-times - Run promise-returning & async functions a specific number of times concurrently
- p-props - Like `Promise.all()` but for `Map` and `Object`
- p-map-series - Map over promises serially
- p-queue - Promise queue with concurrency control
- More...