base64-async DT

2.1.3 • Public • Published 5 years ago

base64-async

> Non-blocking chunked Base64 encoding

Process large Base64 documents without blocking the event loop.

Configurable chunk size option to optimise for your use case.

> **Note:**
>
> Base64 in Node.js is already crazy fast. Breaking the work up into chunks and adding async logic adds **overhead**. If you aren't dealing with large files it will probably be more efficient to just block the event loop for the small amount of time it takes Node.js to process Base64 synchronously.

Install

```
npm install --save base64-async
```

Usage

```
const b64 = require('base64-async');
const fs = require('fs');
const buffer = fs.readFileSync('somehugefile.jpg');

b64.encode(buffer).then(b64String => console.log(b64String));
// aGkgbXVt...

b64.decode(b64String).then(buffer => console.log(buffer));
// <Buffer 68 69 20 6d 75 6d ... >

// or, for the cool kids
const b64String = await b64.encode(buffer);
const buffer = await b64.decode(b64String);

// which is equivalent to this
const b64String = await b64(buffer);
const buffer = await b64(b64String);
// If no method is specified, buffers are encoded, strings are deco
```

Example

```
$ npm run example

Registering 4 asynchronous jobs...
Encoding 100 MB with default Node.js Buffer API...
Base64 encode complete
Hi, I'm an asynchronous job, and I'm late by 231ms
Hi, I'm an asynchronous job, and I'm late by 238ms
```

```
Hi, I'm an asynchronous job, and I'm late by 239ms
Hi, I'm an asynchronous job, and I'm late by 245ms

Registering 4 asynchronous jobs...
Encoding 100 MB with base64-async in chunks of 250 kB...
Hi, I'm an asynchronous job, and I'm on time
Hi, I'm an asynchronous job, and I'm on time
Hi, I'm an asynchronous job, and I'm on time
Hi, I'm an asynchronous job, and I'm on time
Base64 encode complete
```

(**example source code**)

Notice how none of the async jobs can start until the Buffer API has finished encoding and stops blocking the event loop? With `base64-async` the async jobs can execute in-between each chunk of data.

Performance

```
$ npm run bench

Benchmark completed with a chunk size of 250 kB
```

Bytes	Encode Sync	Decode Sync	Encode Async	Decode Async
10 kB	0.097225ms	0.383031ms	1.276201ms	0.537687ms
100 kB	0.198161ms	0.271577ms	0.99799ms	0.356765ms
1 MB	1.924415ms	2.038406ms	2.679117ms	2.544993ms
10 MB	15.749204ms	16.280246ms	33.666111ms	29.918725ms
100 MB	165.189455ms	195.298199ms	246.359068ms	280.792751ms

As you can see, the total processing time is longer with `base64-async` (as we spend some time paused waiting for the event loop). However, if you have an idea of the size of the data you'll be working with, you can play around with the chunk size to get better performance.

The included benchmarking tool accepts arguments to help you test this:

```
$ npm run bench -- --chunkSize=1000000 --bytesToBenchmark=50000000,10(

Benchmark completed with a chunk size of 1 MB
┌─────────┬─────────────┬─────────────┬──────────────┬──────────────┐
│ Bytes   │ Encode Sync │ Decode Sync │ Encode Async │ Decode Async │
├─────────┼─────────────┼─────────────┼──────────────┼──────────────┤
│ 50 MB   │ 79.675533ms │ 87.251079ms │ 92.400367ms  │ 137.468082ms │
├─────────┼─────────────┼─────────────┼──────────────┼──────────────┤
│ 100 MB  │ 203.423705ms│ 173.567974ms│ 186.181857ms │ 264.123311ms │
└─────────┴─────────────┴─────────────┴──────────────┴──────────────┘
```

API

b64(input, [options])

Returns a Promise that resolves to the Base64 encoded/decoded input.

input

Type: `string`, `buffer`

A Base64 string to decode, or a Buffer to encode.

options

Type: `object`

options.chunkSize

Type: `number`
Default: `250000`

Size of the chunk of data to work on before deferring execution to the next iteration of the event loop.

If encoding, the number is interpreted as number of bytes. If decoding, the number is interpreted as number of characters.

b64.encode(input, [options])

Returns a Promise that resolves to a Base64 encoded string.

input

Type: `buffer`

A Buffer to encode.

b64.decode(input, [options])

Returns a Promise that resolves to a decoded Buffer.

input

Type: `string`

A Base64 string to decode.

License

MIT © Luke Childs

Keywords

async asynchronous non-blocking base64 encode decode

Install

```
npm i base64-async
```

Repository

github.com/lukechilds/base64-async

Homepage

github.com/lukechilds/base64-async

Weekly Downloads

1,755

Version

2.1.3

License

MIT

Issues

1

Pull Requests

4

Last publish

5 years ago

Collaborators

Try on RunKit

Report malware

Support

Help

Advisories

Status

Contact npm

Company

About

Blog

Press

Terms & Policies

Policies

Terms of Use

Code of Conduct

Privacy