Unraveling JavaScript's Heart
Mastering the Event Loop for Peak Performance

Antoine PAIRET

ROSA

FOSDEM
04-02-2024

Don’t **block** the **event loop**!

“Prefer ***asynchronous*** code over ***synchronous***”

Don’t **block** the **event loop**!

“Prefer ***asynchronous*** code over ***synchronous***”

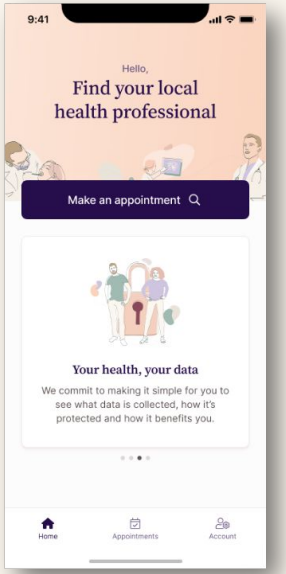
What does that mean?
Is this accurate?

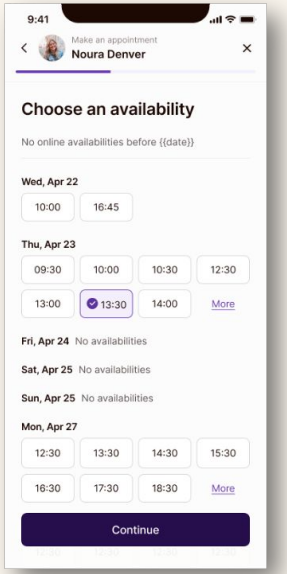
Antoine Pairet

Co-founder & CTO

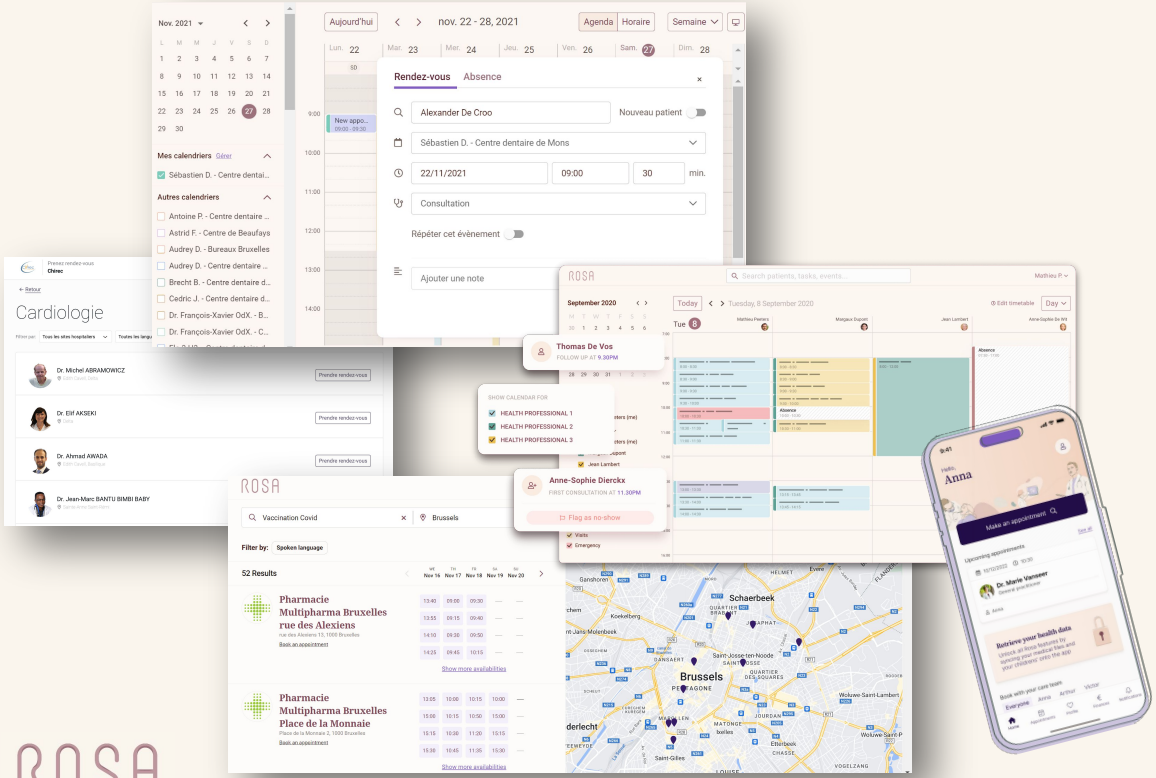
@antoinepairet

ROSA





Parts of Rosa are **CPU “intensive”** or **CPU bound**



rrule

To compute recurrence

bcrypt

For hashing

ical

For import and export

Date diffing

To compute free slots

ROSA

We will look at performance from a ***Mission Critical*** angle

Does it scale?

What if traffic does 3x, 10x?

Is there a DoS risk?

Why was node.js created?

What is the Event loop?

Analysis of bcrypt.js

Thread pools to the rescue!

What are the metrics of the event loop?

Node.js is born because most web apps are ***I/O bound***

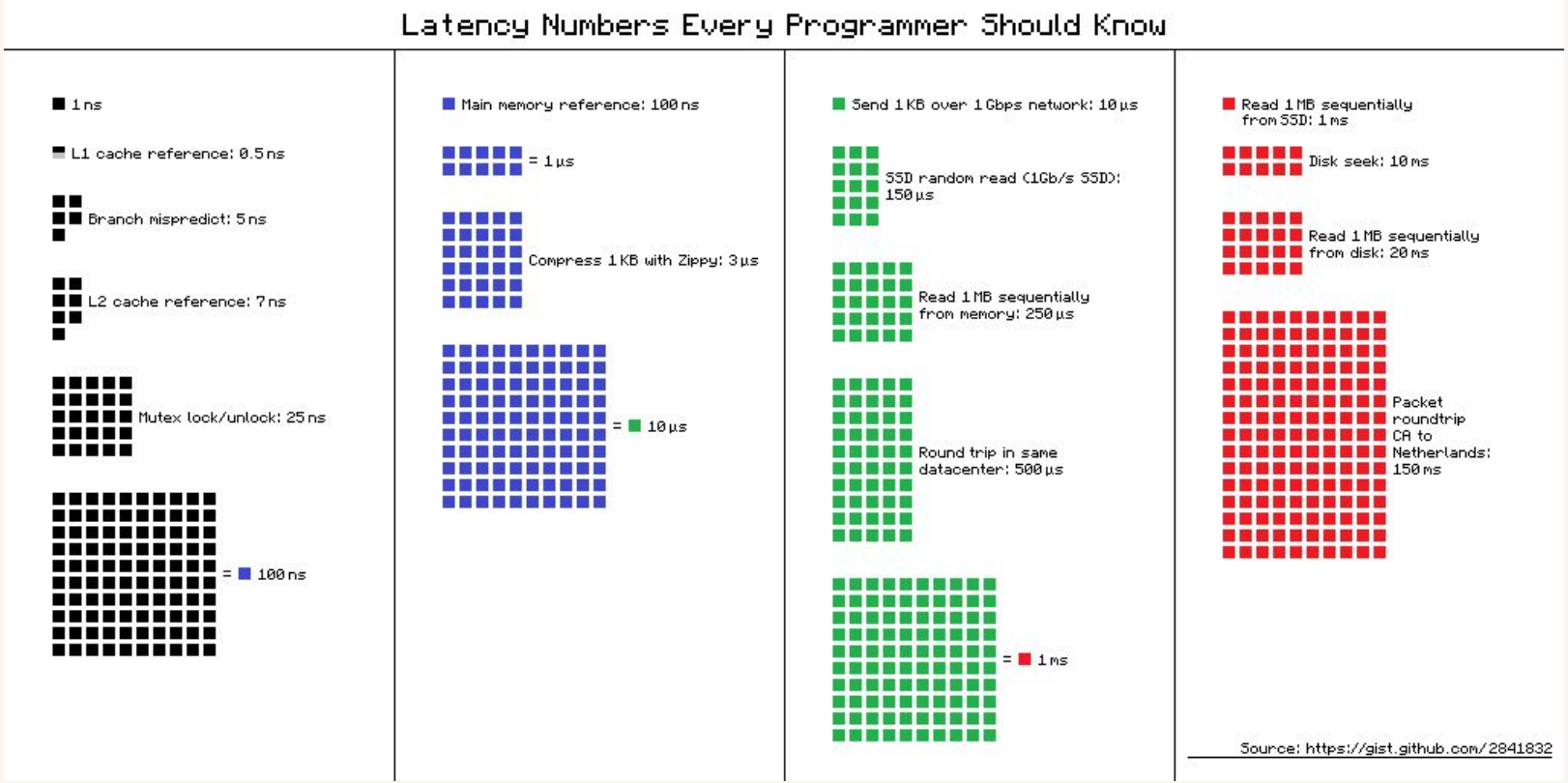
```
// BEFORE: blocking IO
result = db.query('select * from T')
doSomething(result)

// ================

// Node.js: non-blocking IO
db.query('select * from T', function (err,
result) => {
    doSomething(result)
})
```

1 thread per connection
memory overhead
does not scale

single threaded event loop
requires non-blocking IO
scales well if not CPU intensive

https://www.youtube.com/watch?v=M-sc73Y-zQA

Node.js is born because most web apps are ***I/O bound***



https://gist.github.com/hellerbarde/2843375

CPU is ***orders of magnitude faster*** than **I/O**

25 s

50 min

5.8 days

4.8 years

CPU

Mutex lock/unlock

Compress 1K bytes with Zippy

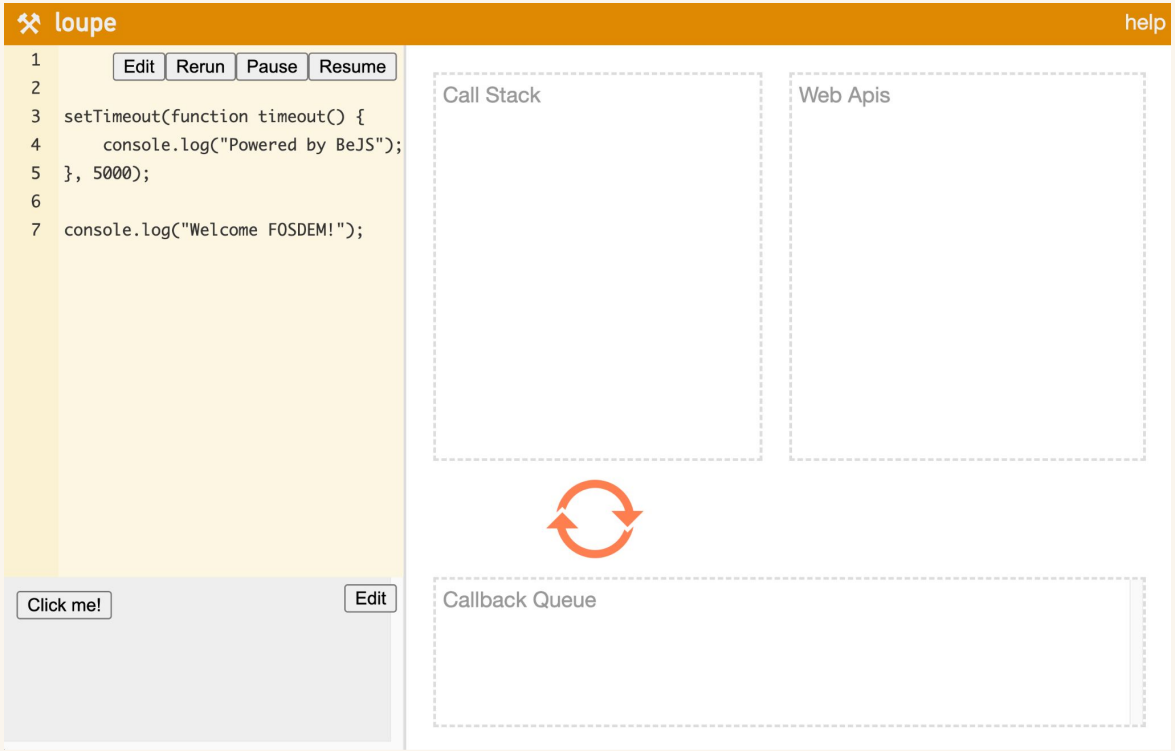
IO

Round trip within same datacenter

Send packet CA->Netherlands->CA

https://gist.github.com/hellerbarde/2843375

What the heck is the event loop?



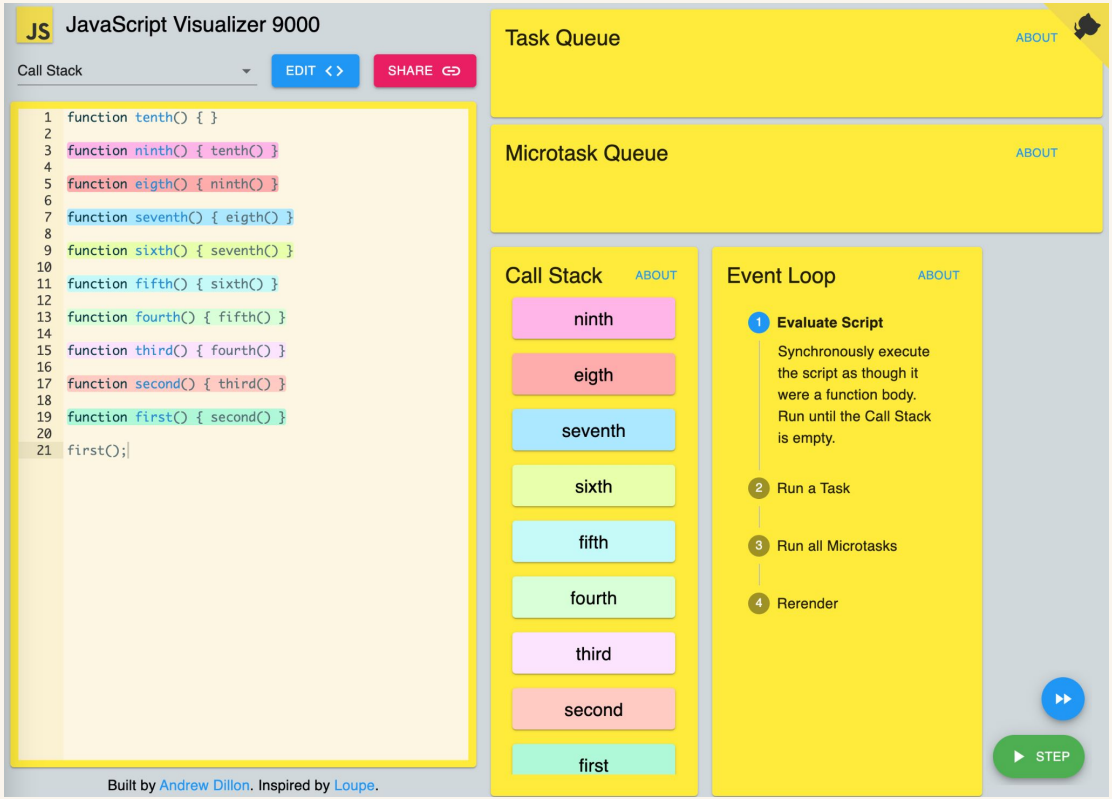
By Philip Roberts

10 years ago!

https://latentflip.com/loupe

https://www.youtube.com/watch?v=8aGhZQkoFbQ&ab_channel=JSConf

What the heck is the event loop?



By Andrew Dillon

https://www.jsv9000.app/

Highlights task vs microtasks

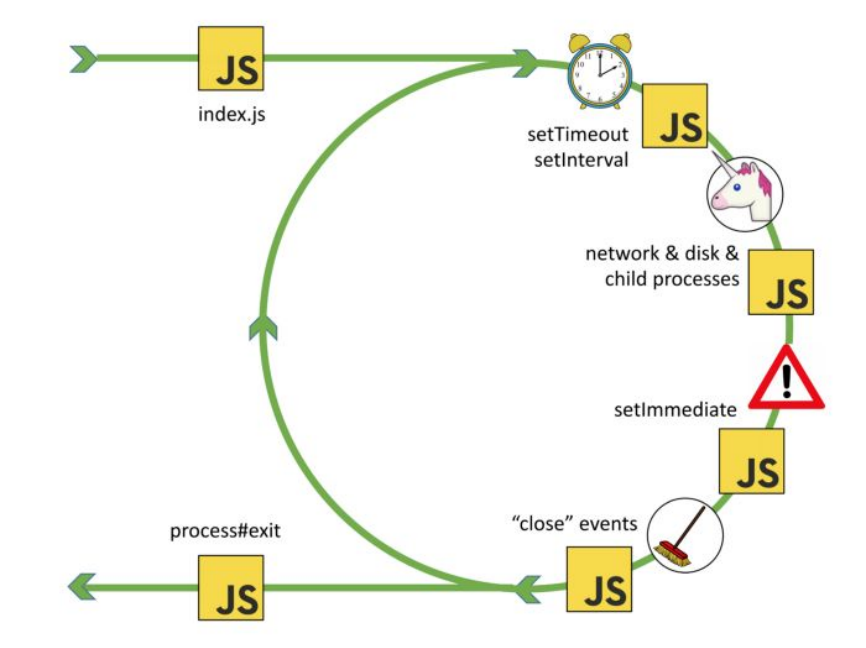
Supports promises

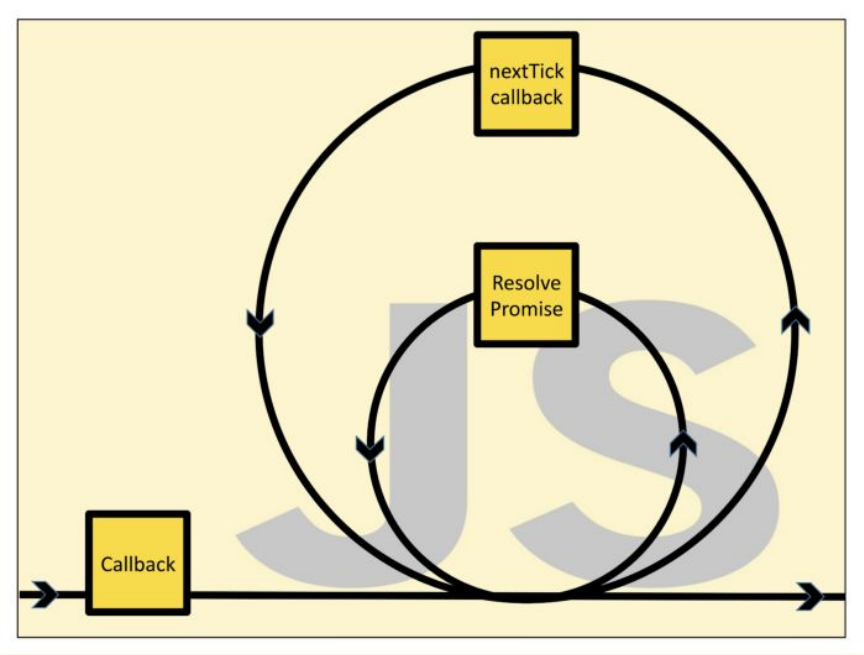
Supports newer APIs

No Web APIs visualization

https://github.com/Hopding/js-visualizer-9000-client
https://github.com/Hopding/js-visualizer-9000-server

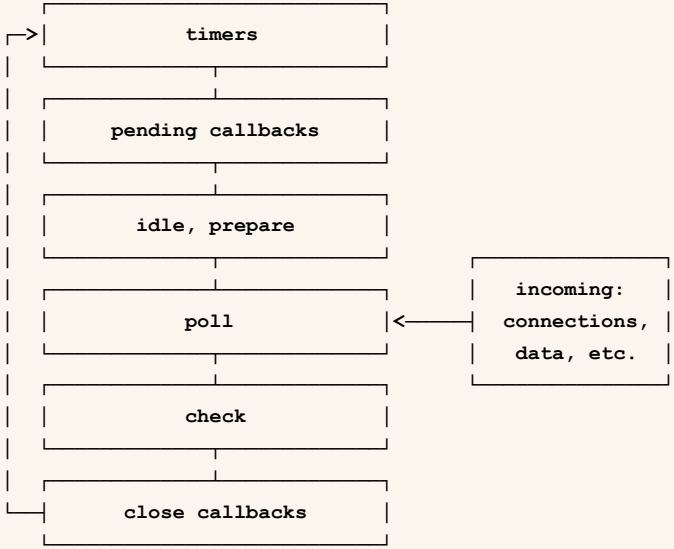
There are multiple queues

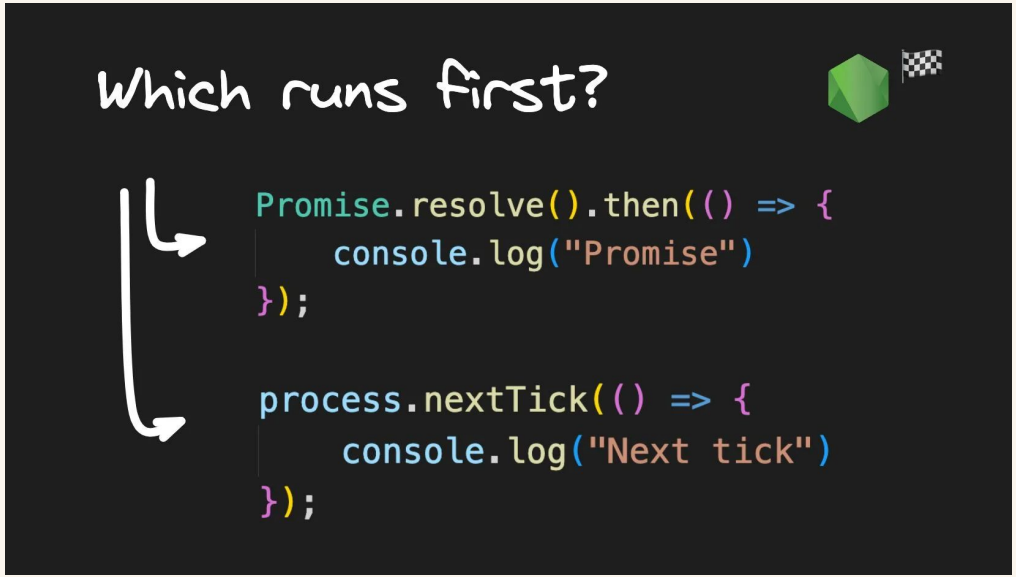




https://www.youtube.com/watch?v=PNa9OMajw9w

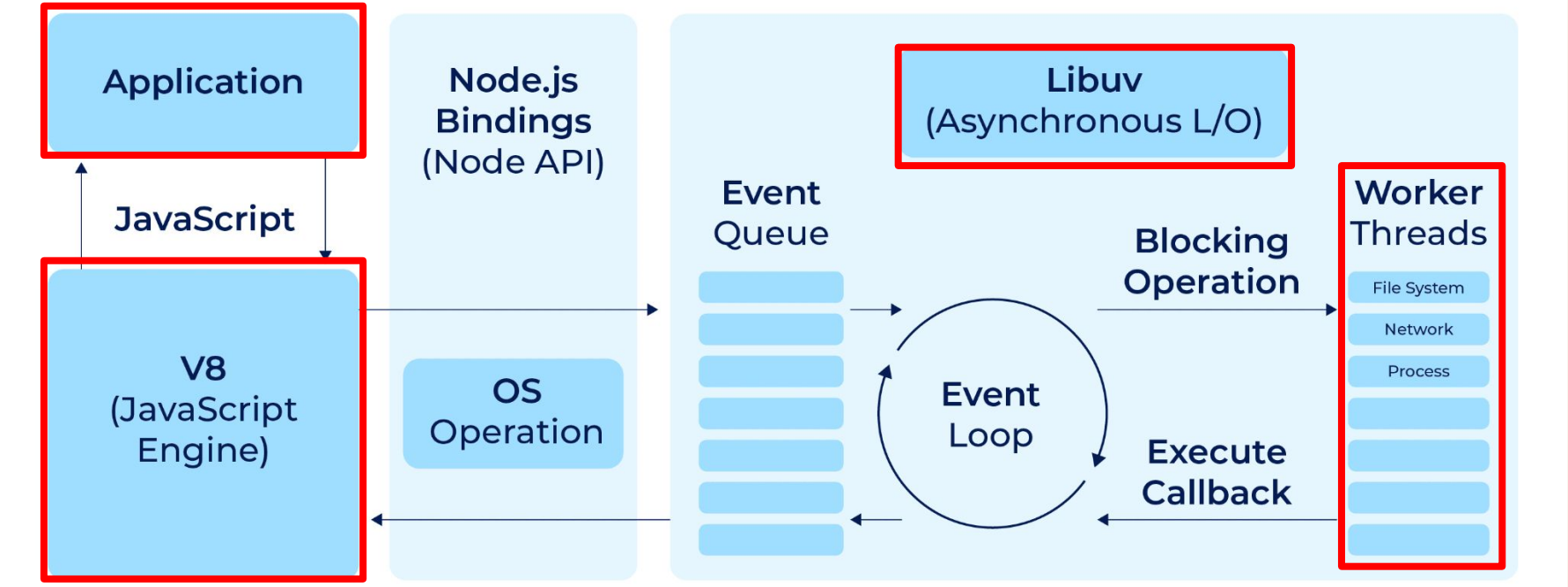
The event loop has multiple phases





https://nodejs.org/en/guides/event-loop-timers-and-nexttick
https://www.builder.io/blog/visual-guide-to-nodejs-event-loop

Node.js architecture is inherently multi-threaded



https://litslink.com/blog/node-js-architecture-from-a-to-z

Node.js architecture is inherently multi-threaded

5+

Main thread

JavaScript code and the ***event loop***.

Libuv's thread pool

asynchronous tasks

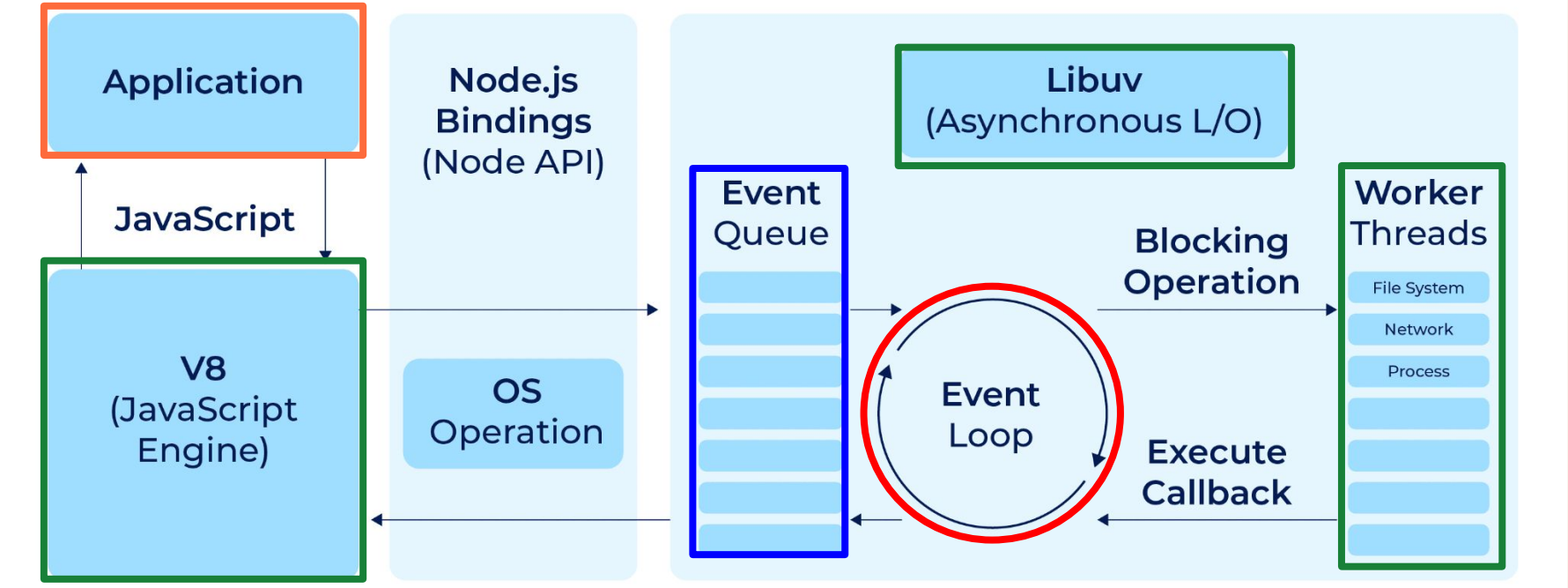
blocking system tasks

Internal threads

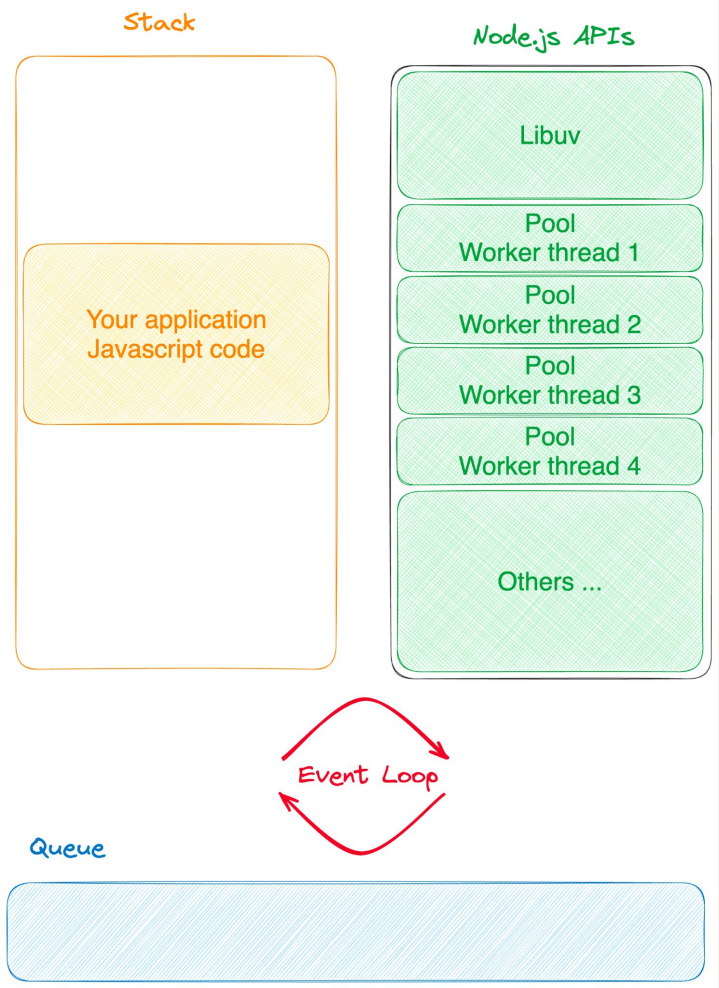
for garbage collection

other runtime management tasks

Node.js architecture is inherently multi-threaded



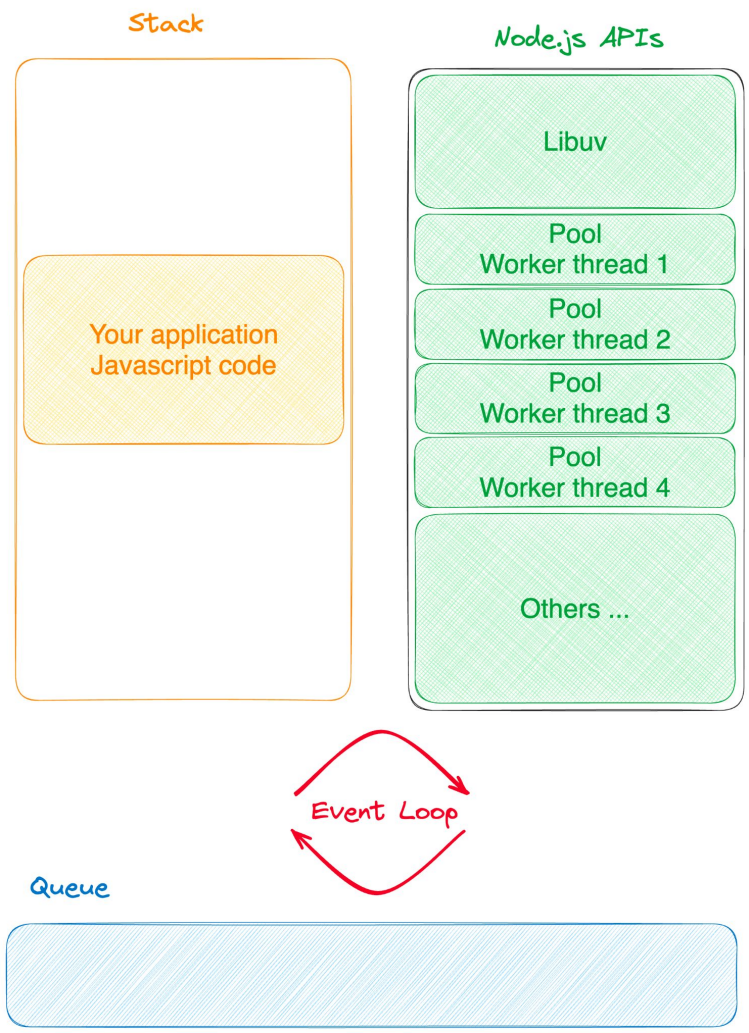
https://litslink.com/blog/node-js-architecture-from-a-to-z


Stack
Your application
Javascript code
Node.js APIs
Libuv
Pool
Worker thread 1
Pool
Worker thread 2
Pool
Worker thread 3
Pool
Worker thread 4
Others ...
Event Loop
Queue

“Prefer ***asynchronous*** code over ***synchronous***”

Core Modules

Pure Javascript libraries




```javascript
const fs = require('node:fs');

const f = 'fileName.txt';
const enc = 'utf8'


// Libuv thread pool
// ==> Non-blocking
fs.readFile(f, enc, (e, d) => {
  console.log(d);
});


// Executed on the main thread
// ==> Blocking
const data = fs.readFileSync(f, enc);
```

“Prefer ***asynchronous*** code over ***synchronous***”

Core Modules

Pure Javascript libraries

How To Safely Store A Password

In which I recommend bcrypt.

31 Jan 2010

Use bcrypt

Use bcrypt. Use bcrypt. Use bcrypt. Use bcrypt. Use bcrypt. Use bcrypt. Use bcrypt. Use bcrypt. Use bcrypt.

https://codahale.com/how-to-safely-store-a-password/

Which bcrypt library should you choose?

Repository

github.com/dcodeIO/bcrypt.js

Homepage

github.com/dcodeIO/bcrypt.js#readme

Weekly Downloads

1,896,411

Pure javascript
Known as ***bcryptjs*** on npm

Repository

github.com/kelektiv/node.bcrypt.js

Homepage

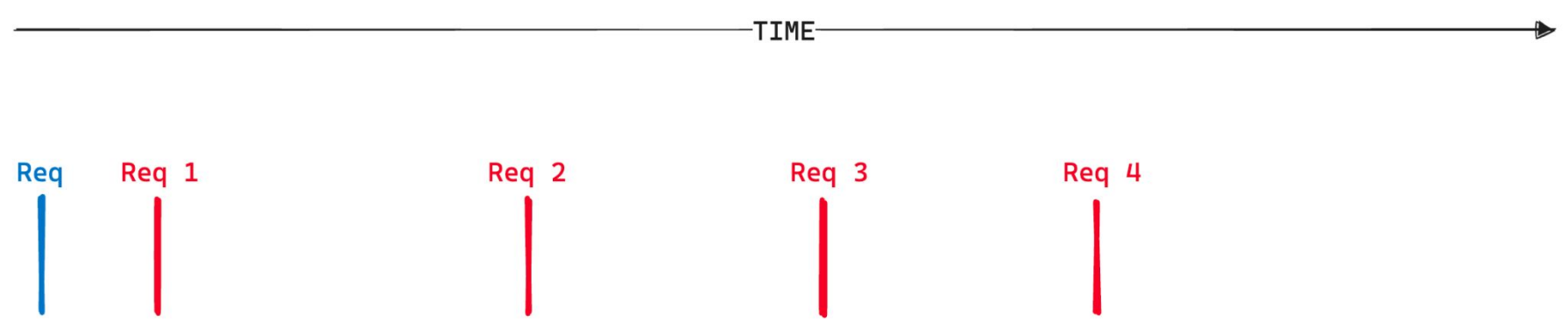
github.com/kelektiv/node.bcrypt.js#rea...

Weekly Downloads

1,475,183

C++ implementation
Known as ***bcrypt*** on npm

```
const bcrypt = require('bcryptjs');

const rounds = 15;
const secret = 'I love BeJS';

// SYNCHRONOUS
const hashed = bcrypt.hashSync(secret, hashRounds);
console.log(hashed);

// ASYNCHRONOUS
bcrypt.hash(secret, rounds, (err, hashed) => {
  console.log(hashed);
});
```

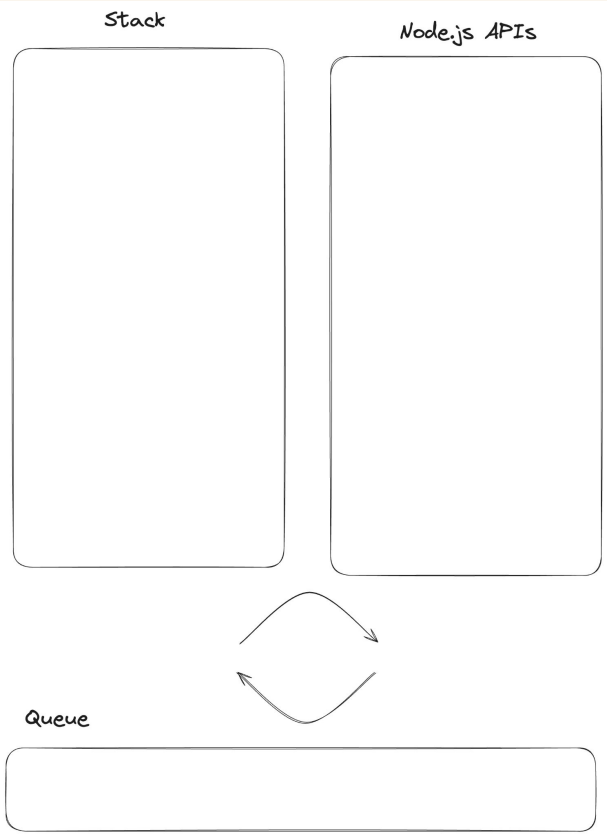

TIME
Req
Req 1
Req 2
Req 3
Req 4

bcryptjs ***asynchronous*** API

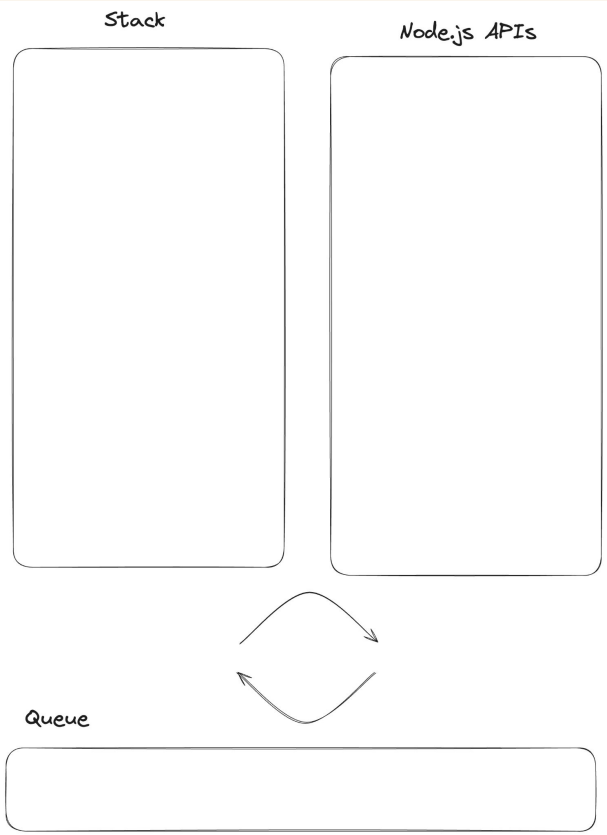
splits the work

into ***smaller synchronous*** chunks

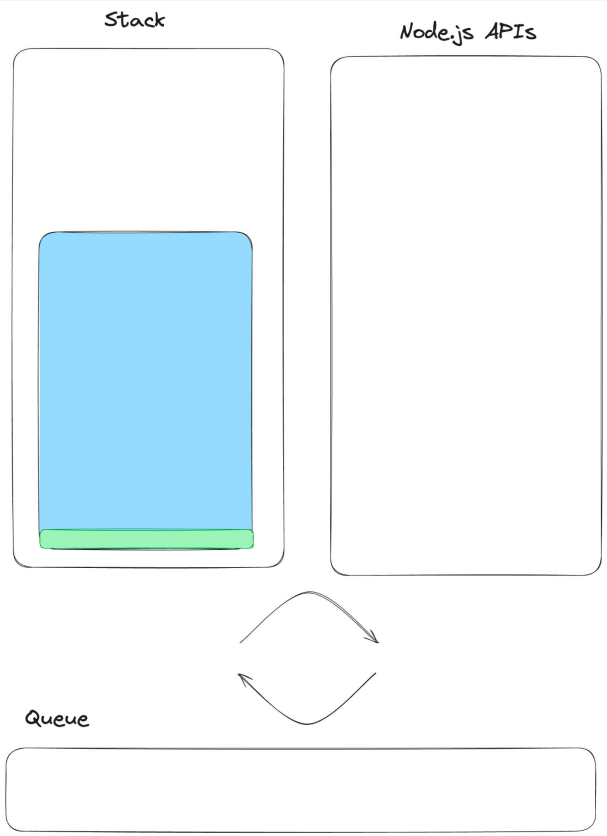
bcryptjs with the ***synchronous*** API



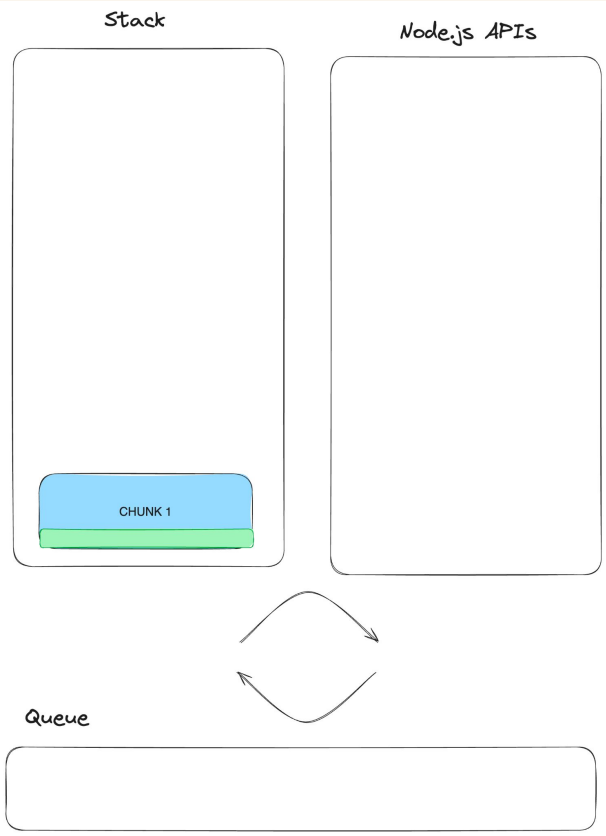
bcryptjs with the ***asynchronous*** API



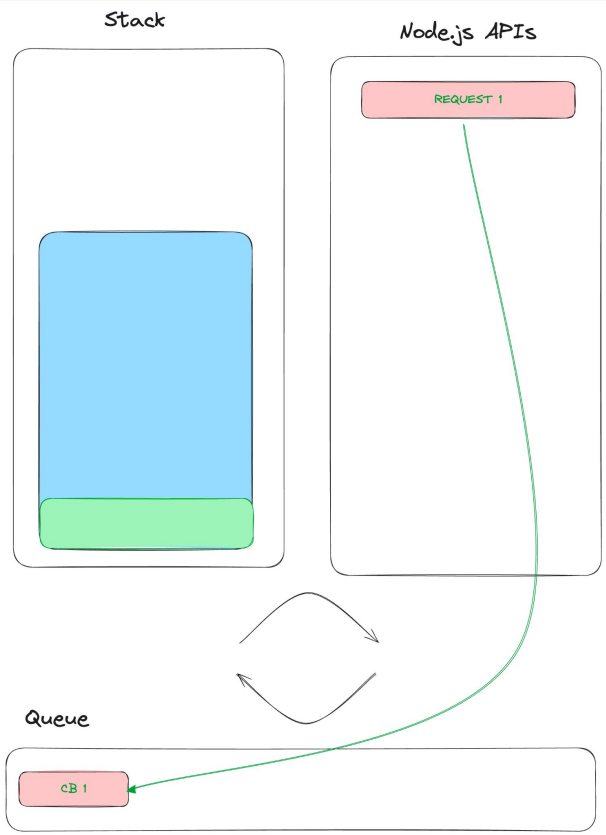
bcryptjs with the synchronous API



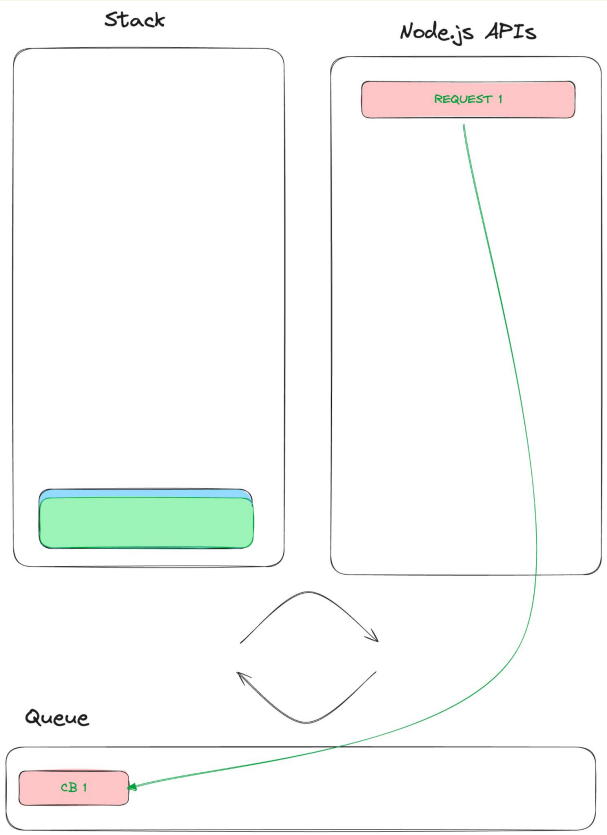
bcryptjs with the asynchronous API



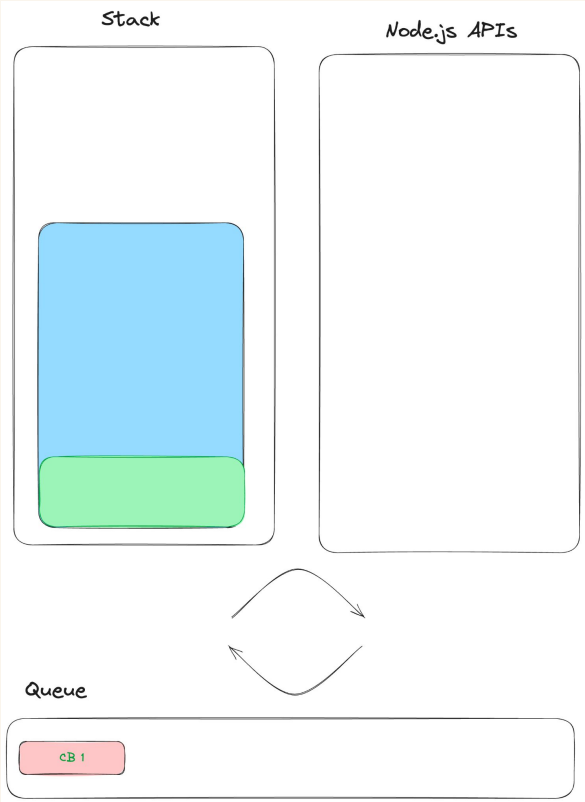
bcryptjs with the synchronous API



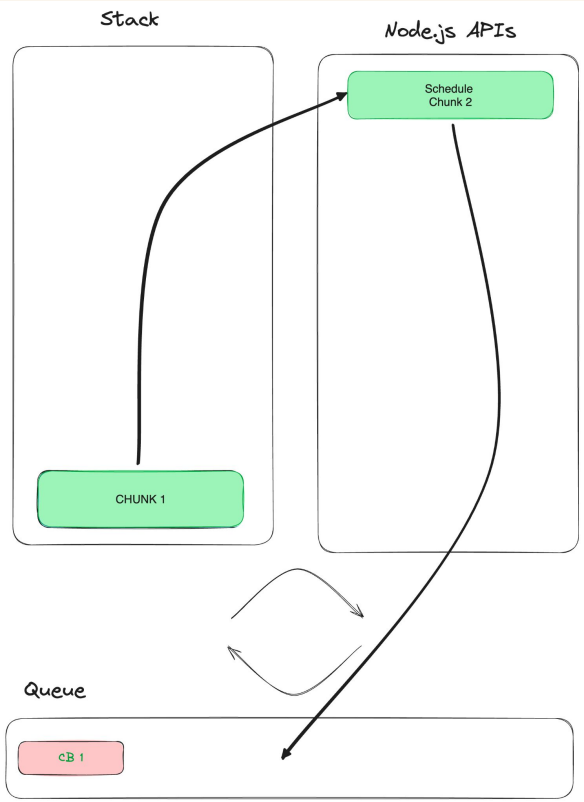
bcryptjs with the asynchronous API



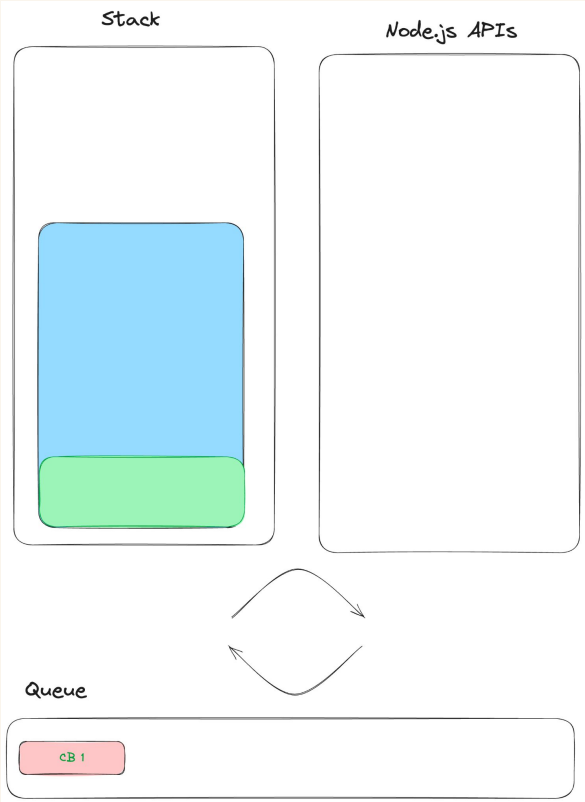
bcryptjs with the synchronous API



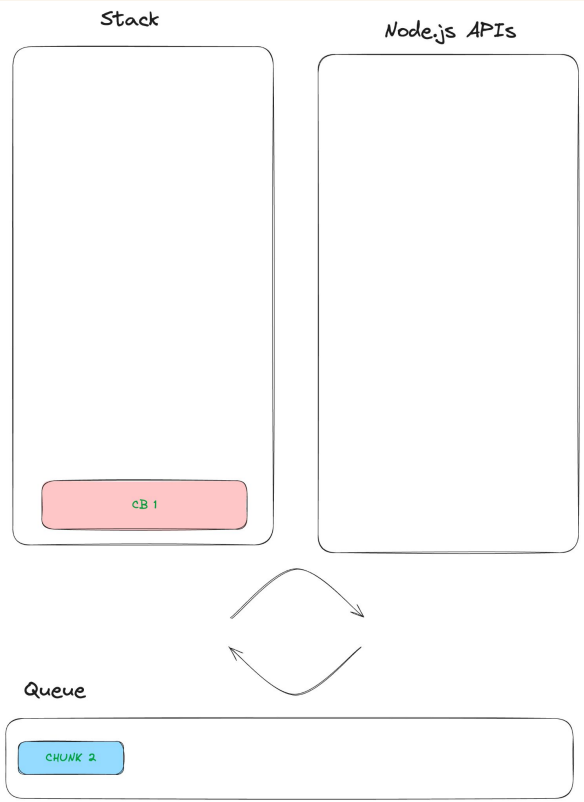
bcryptjs with the asynchronous API



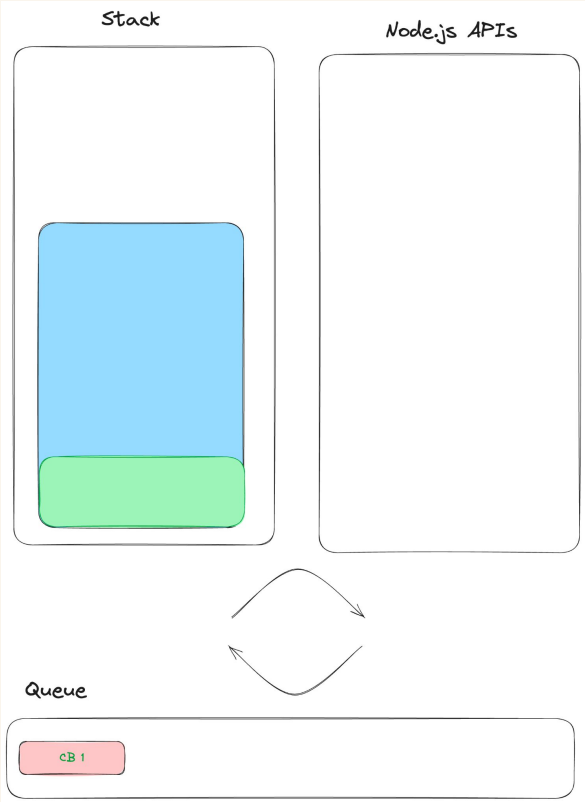
bcryptjs with the synchronous API



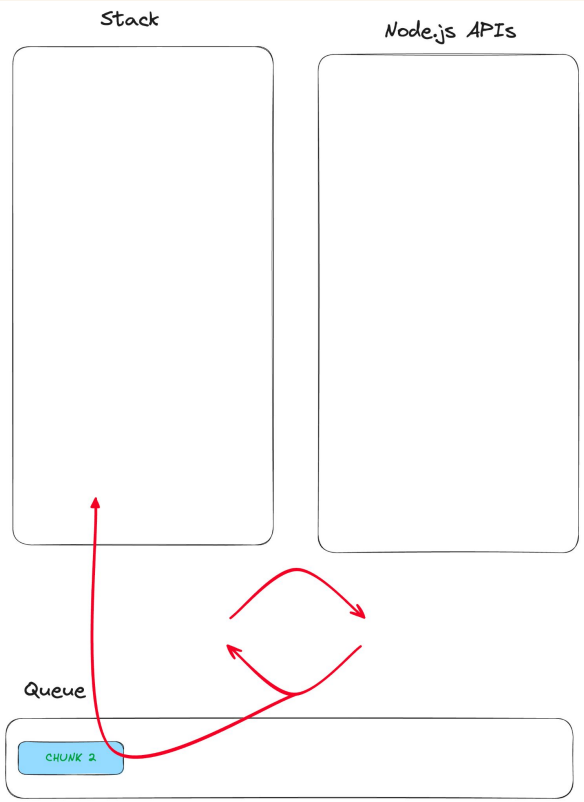
bcryptjs with the asynchronous API



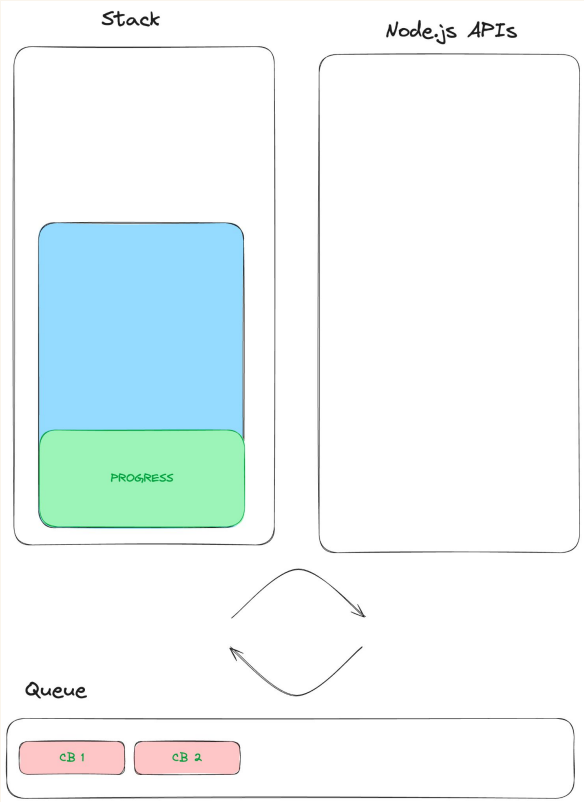
bcryptjs with the synchronous API



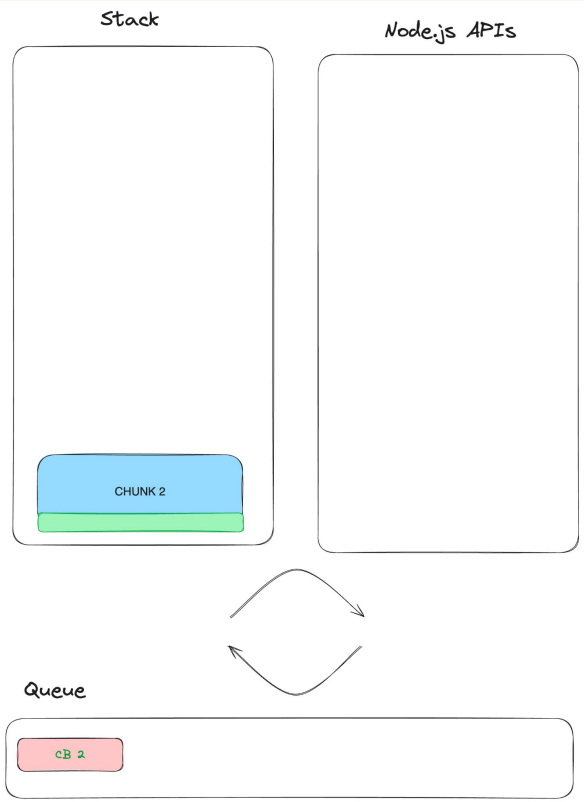
bcryptjs with the asynchronous API



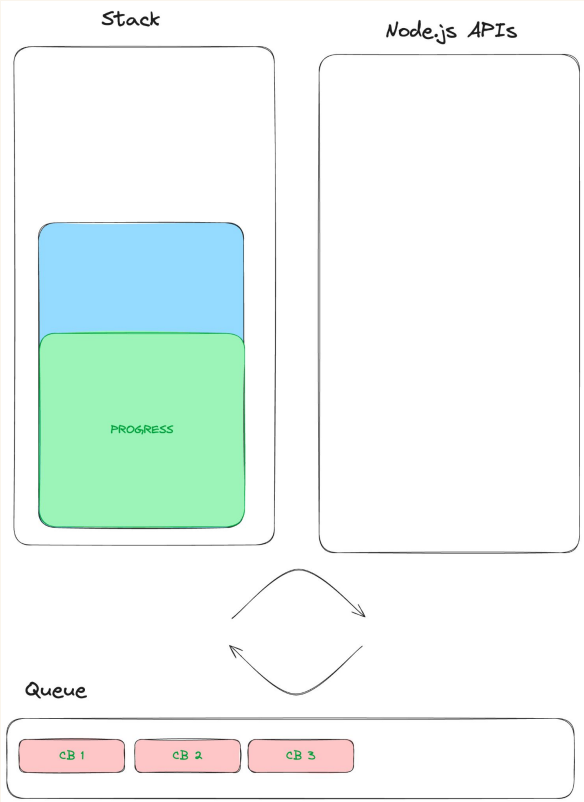
bcryptjs with the synchronous API



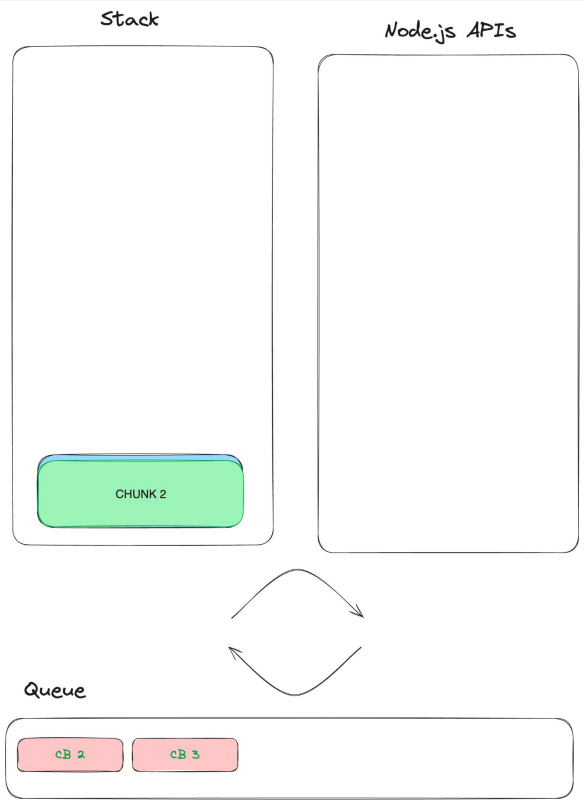
bcryptjs with the asynchronous API



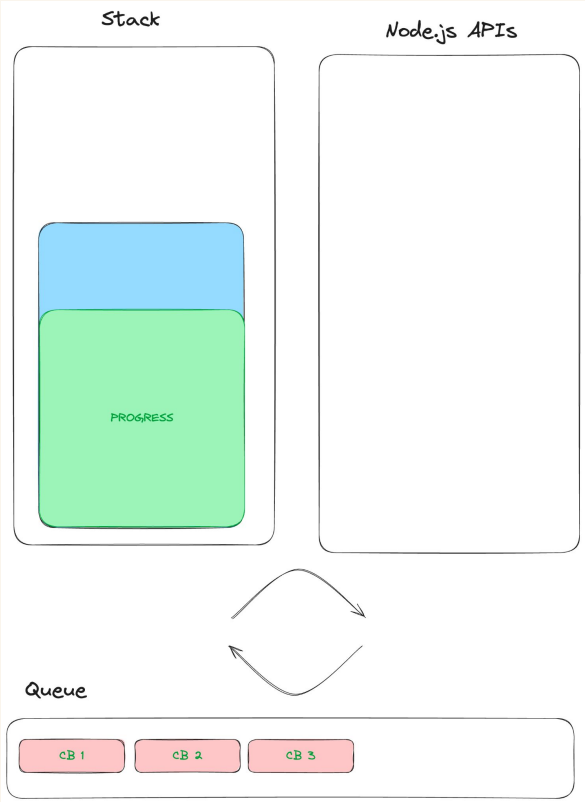
bcryptjs with the synchronous API



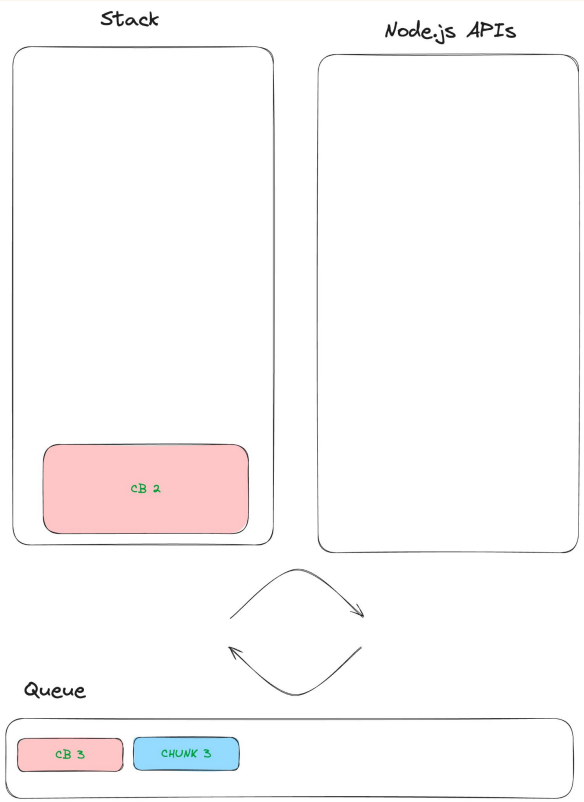
bcryptjs with the asynchronous API



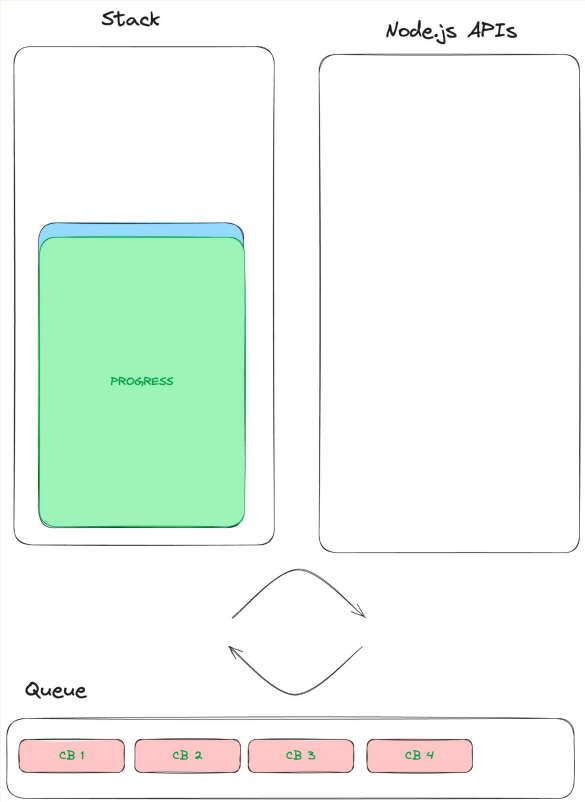
bcryptjs with the synchronous API



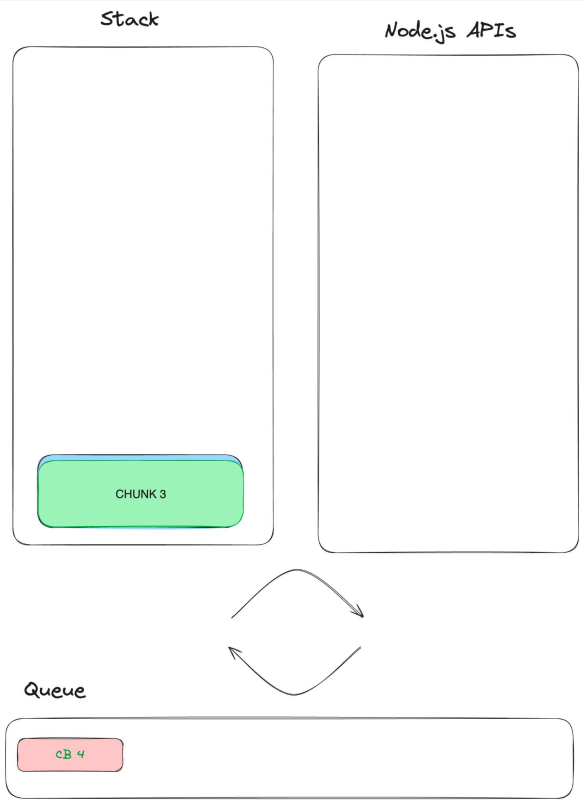
bcryptjs with the asynchronous API



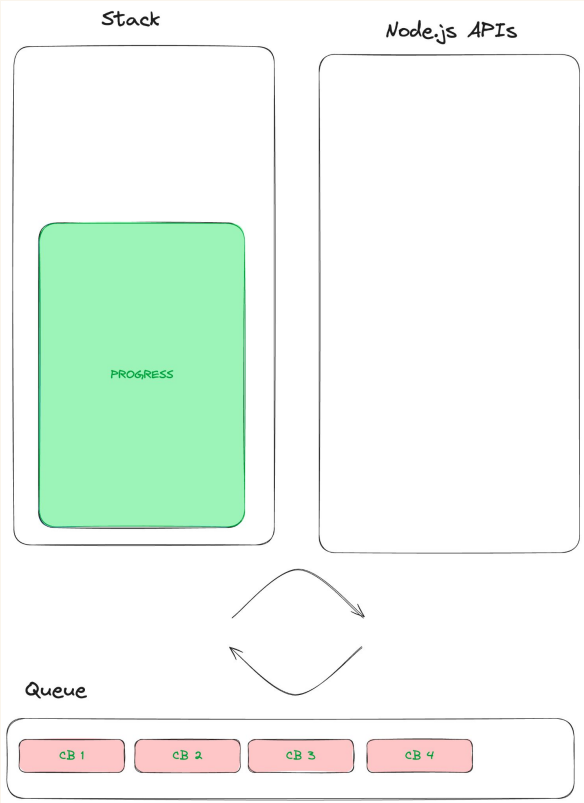
bcryptjs with the synchronous API



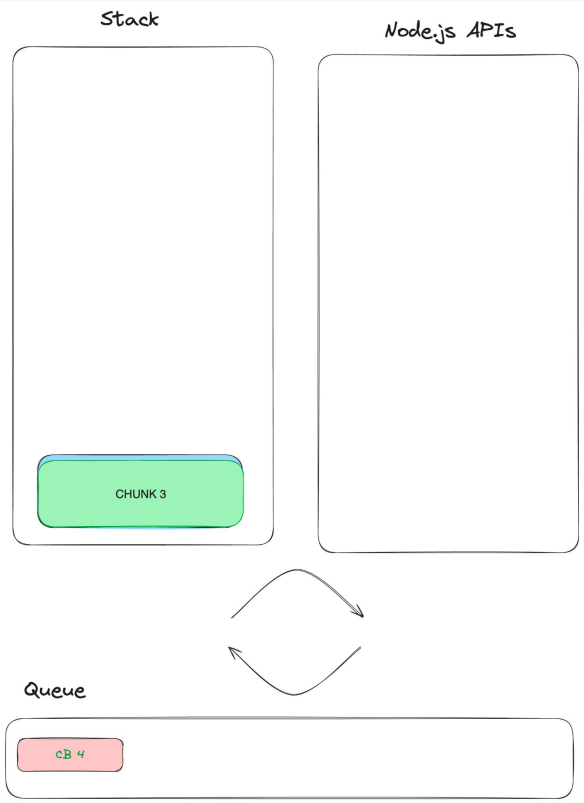
bcryptjs with the asynchronous API



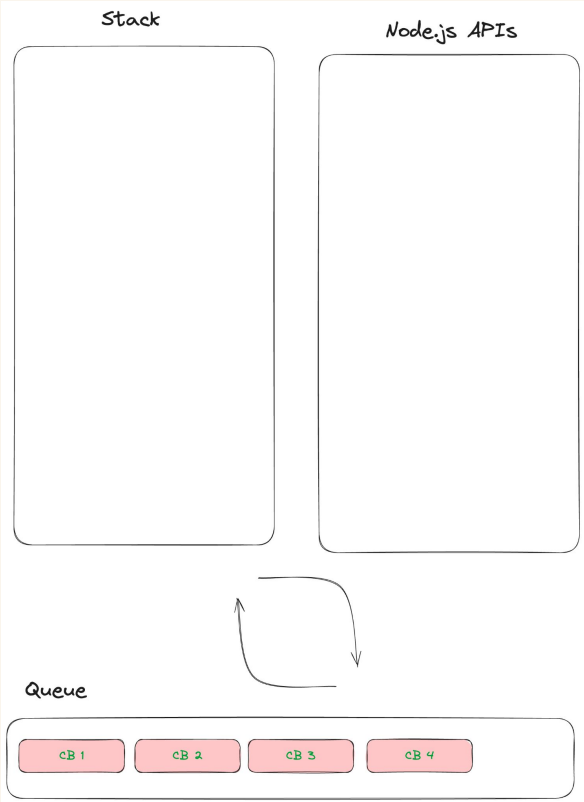
bcryptjs with the synchronous API



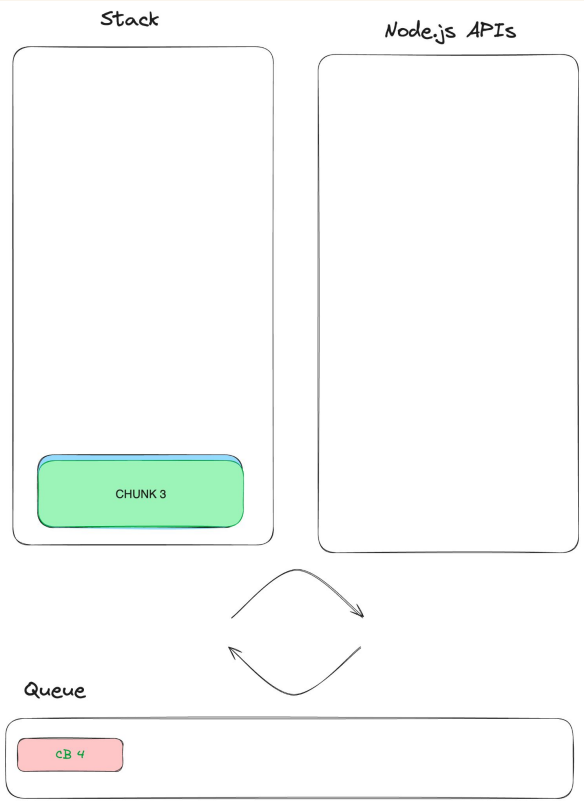
bcryptjs with the asynchronous API



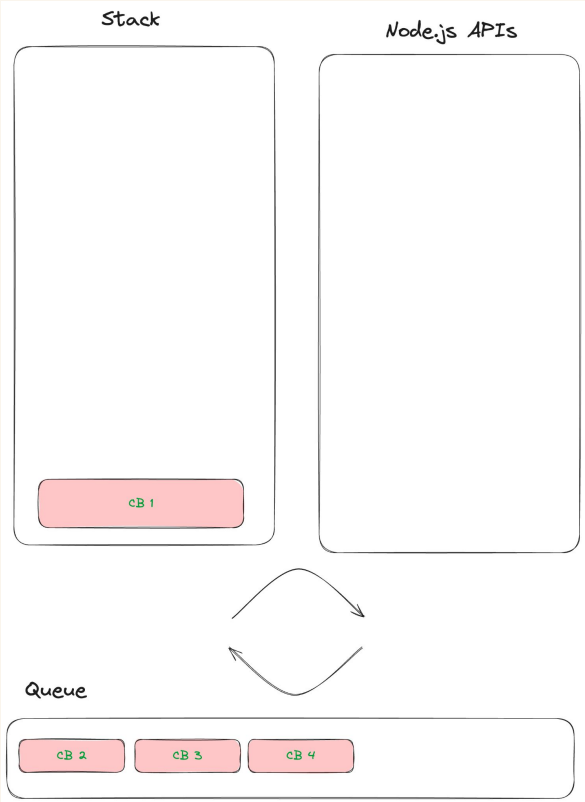
bcryptjs with the synchronous API



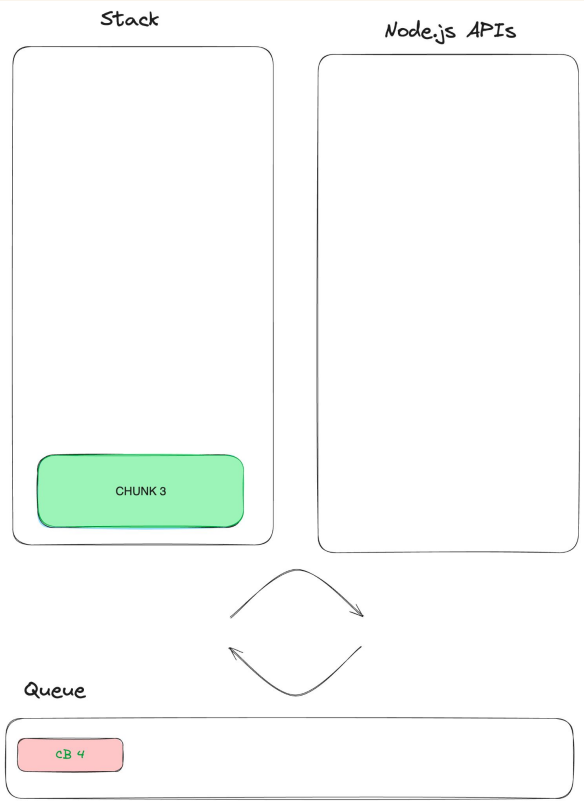
bcryptjs with the asynchronous API



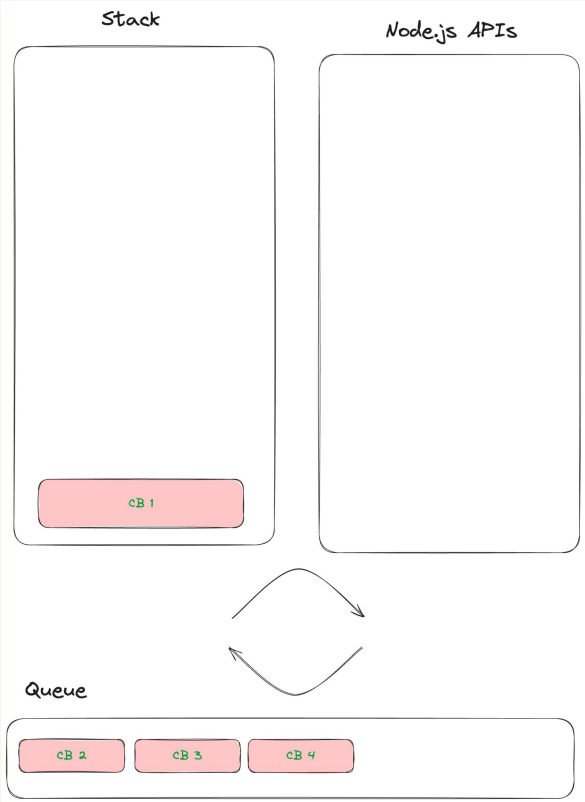
bcryptjs with the synchronous API



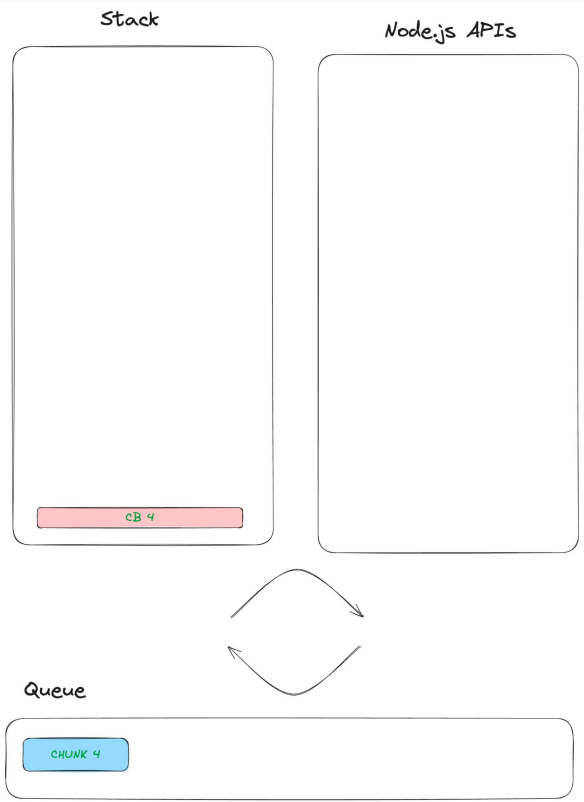
bcryptjs with the asynchronous API



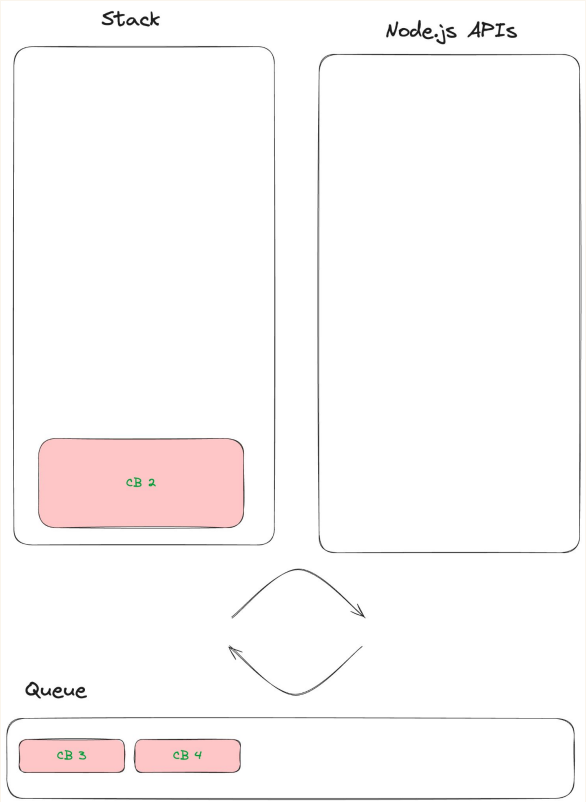
bcryptjs with the synchronous API



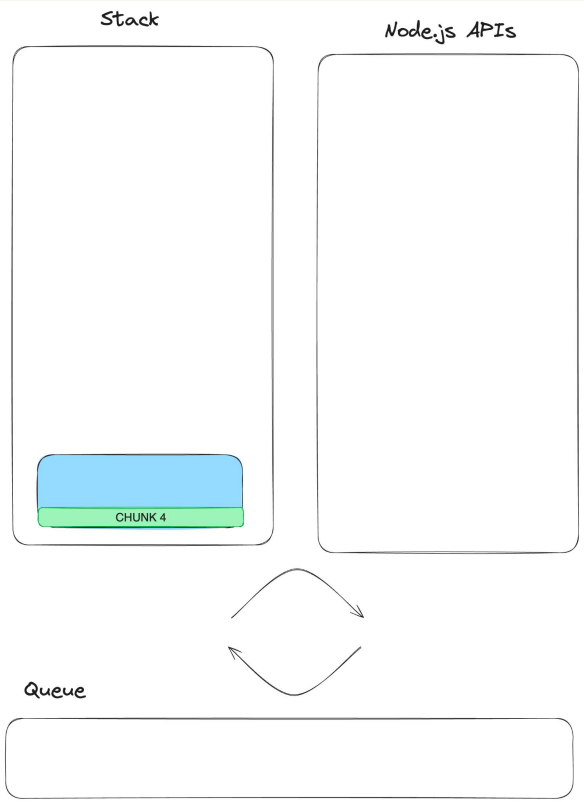
bcryptjs with the asynchronous API



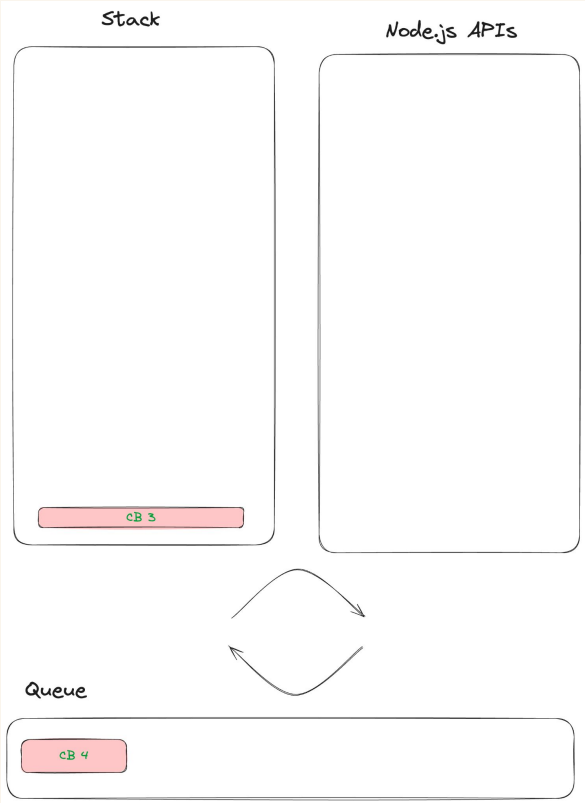
bcryptjs with the synchronous API



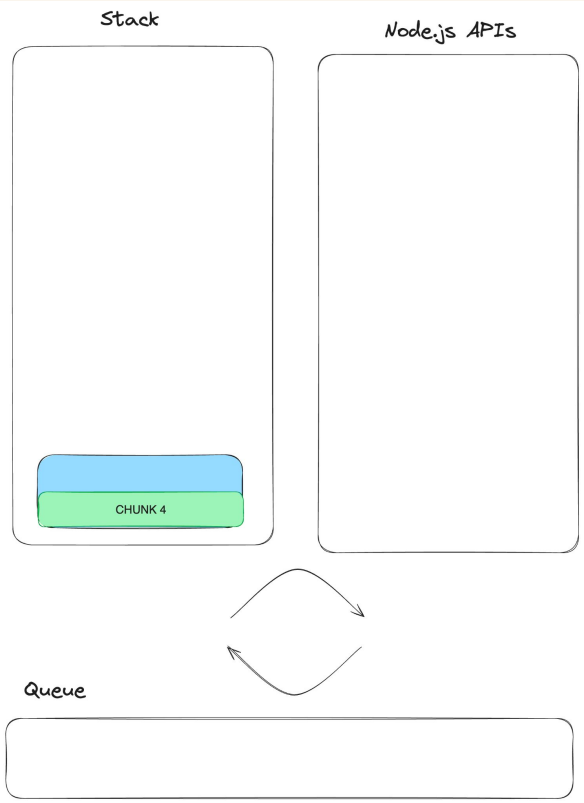
bcryptjs with the asynchronous API



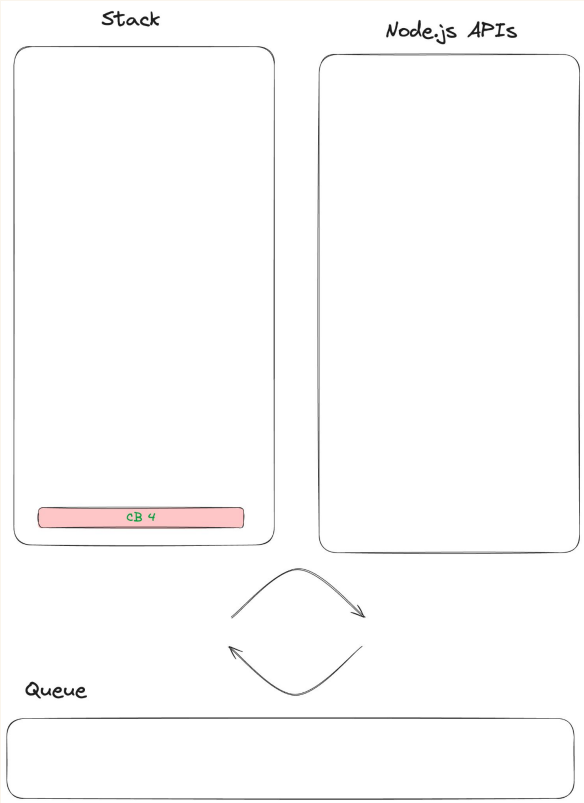
bcryptjs with the synchronous API



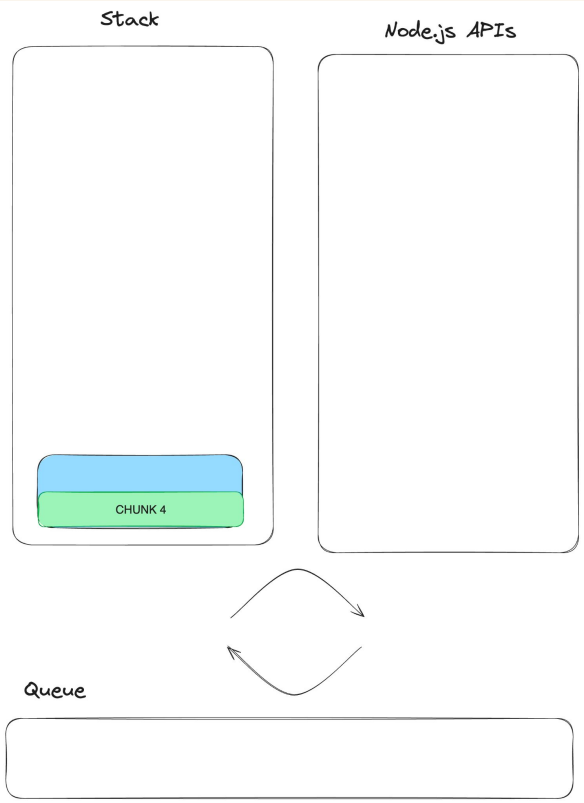
bcryptjs with the asynchronous API



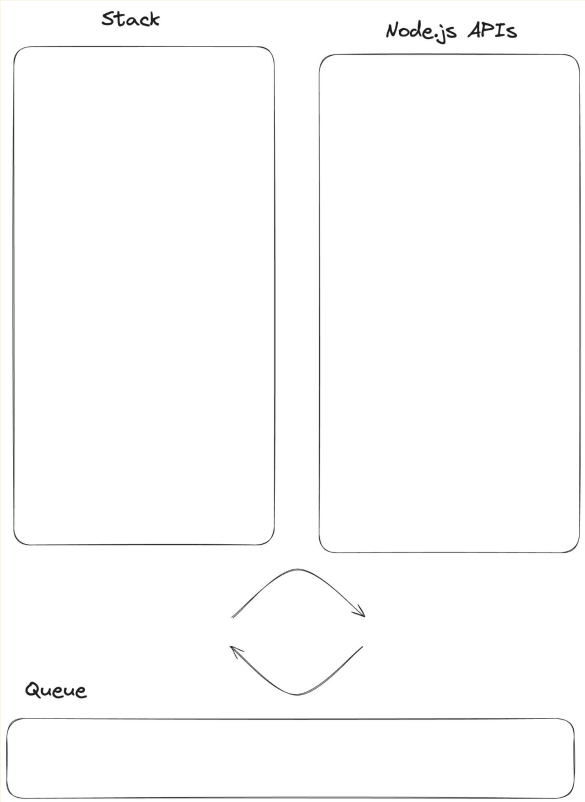
bcryptjs with the synchronous API



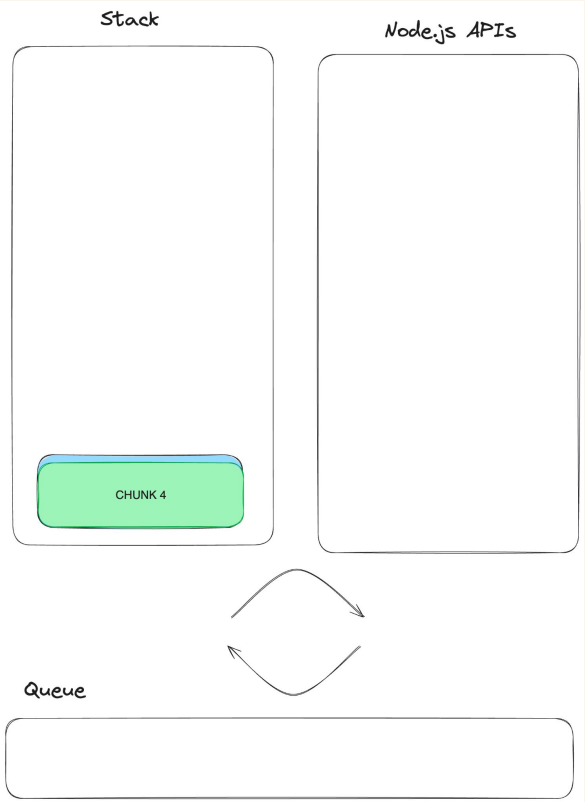
bcryptjs with the asynchronous API



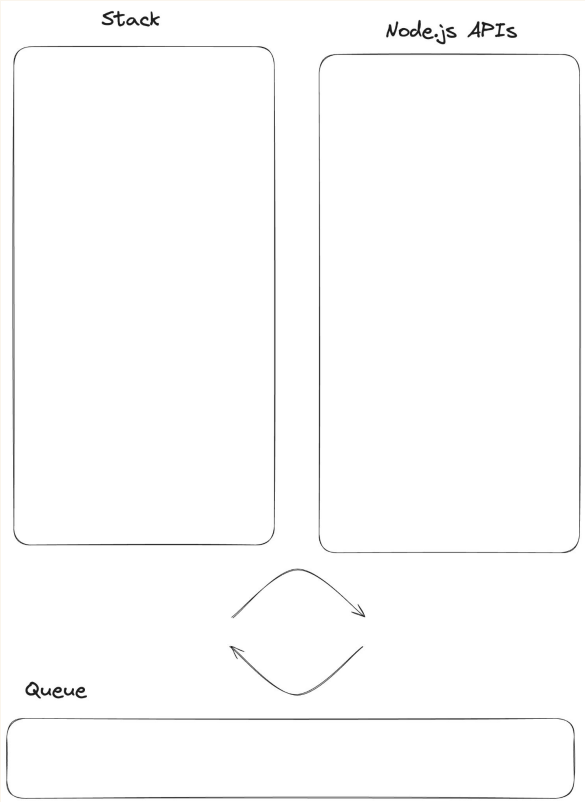
bcryptjs with the synchronous API



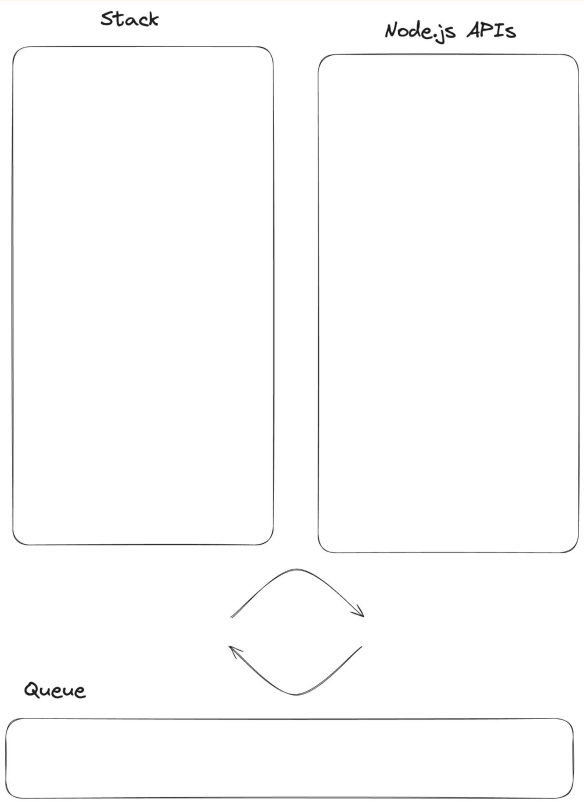
bcryptjs with the asynchronous API

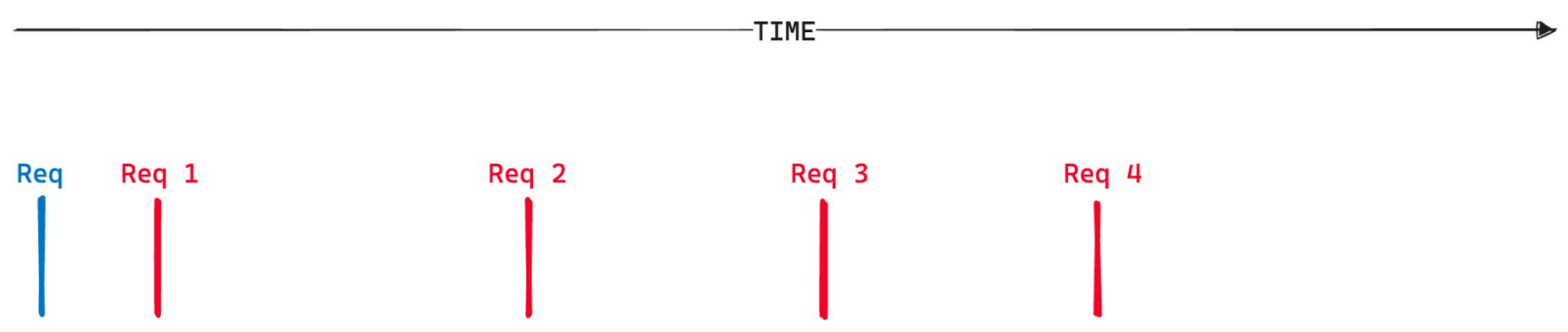


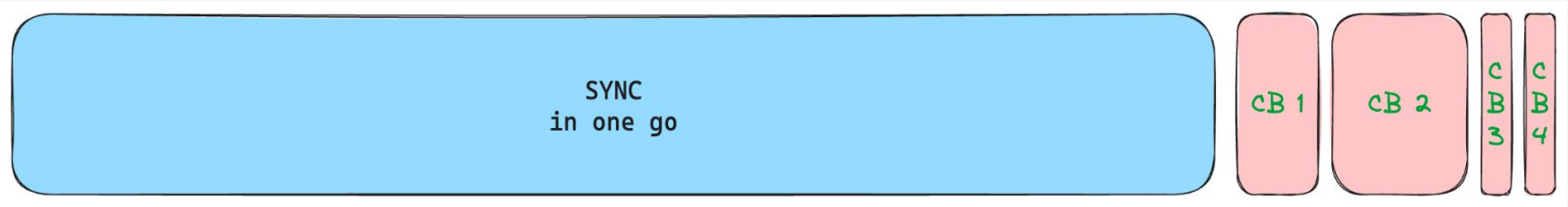
bcryptjs with the synchronous API

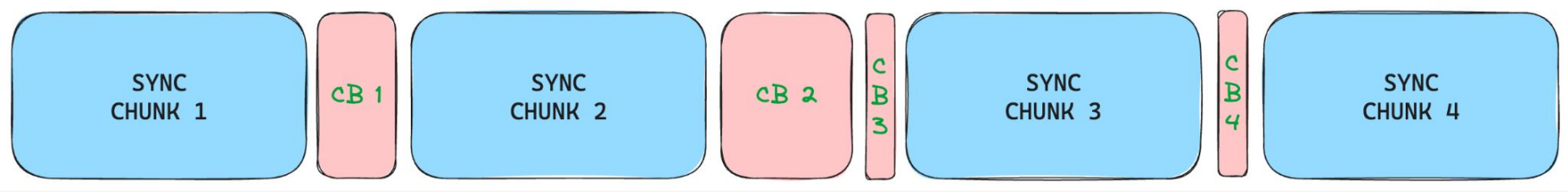


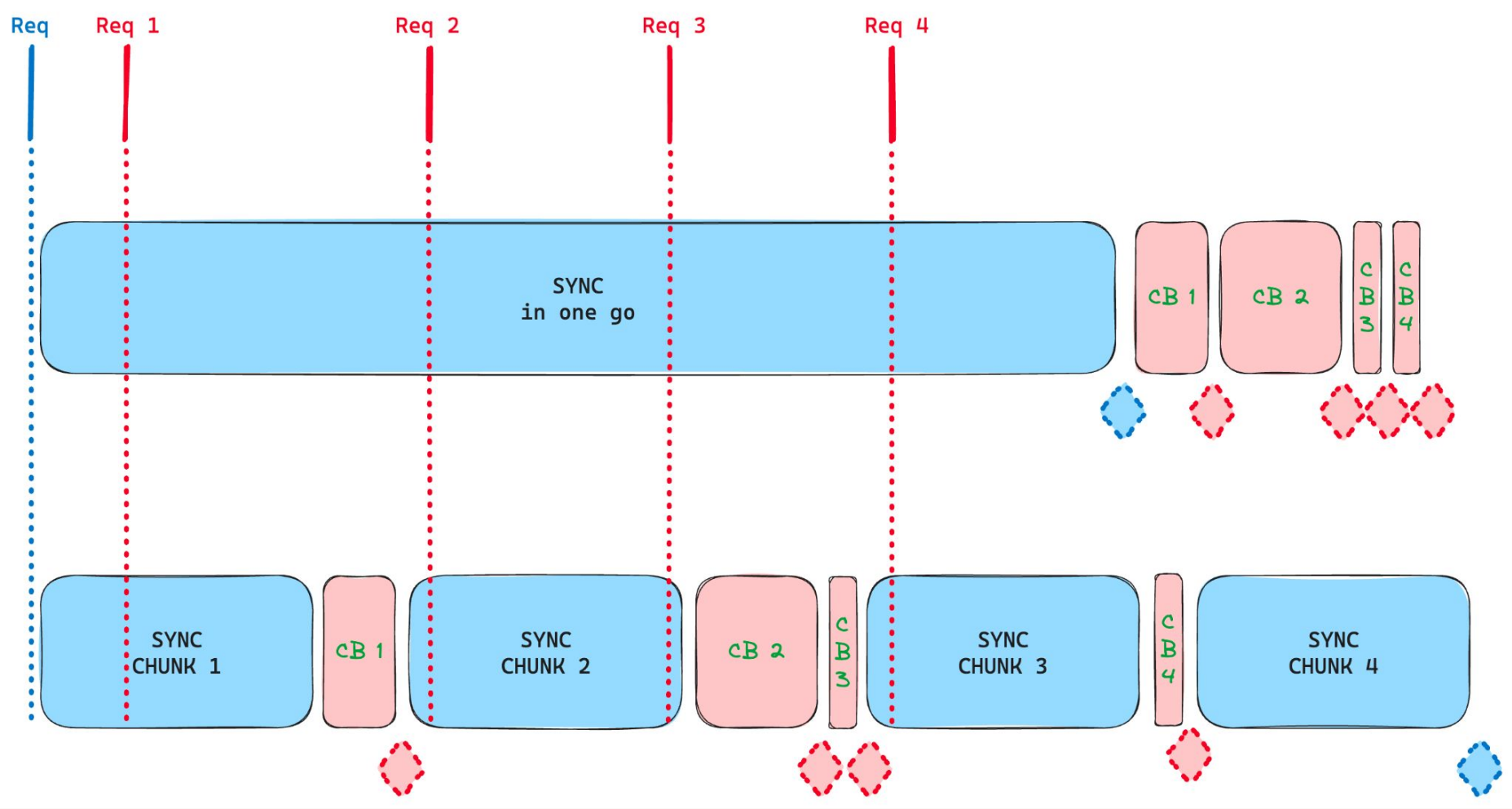
bcryptjs with the asynchronous API

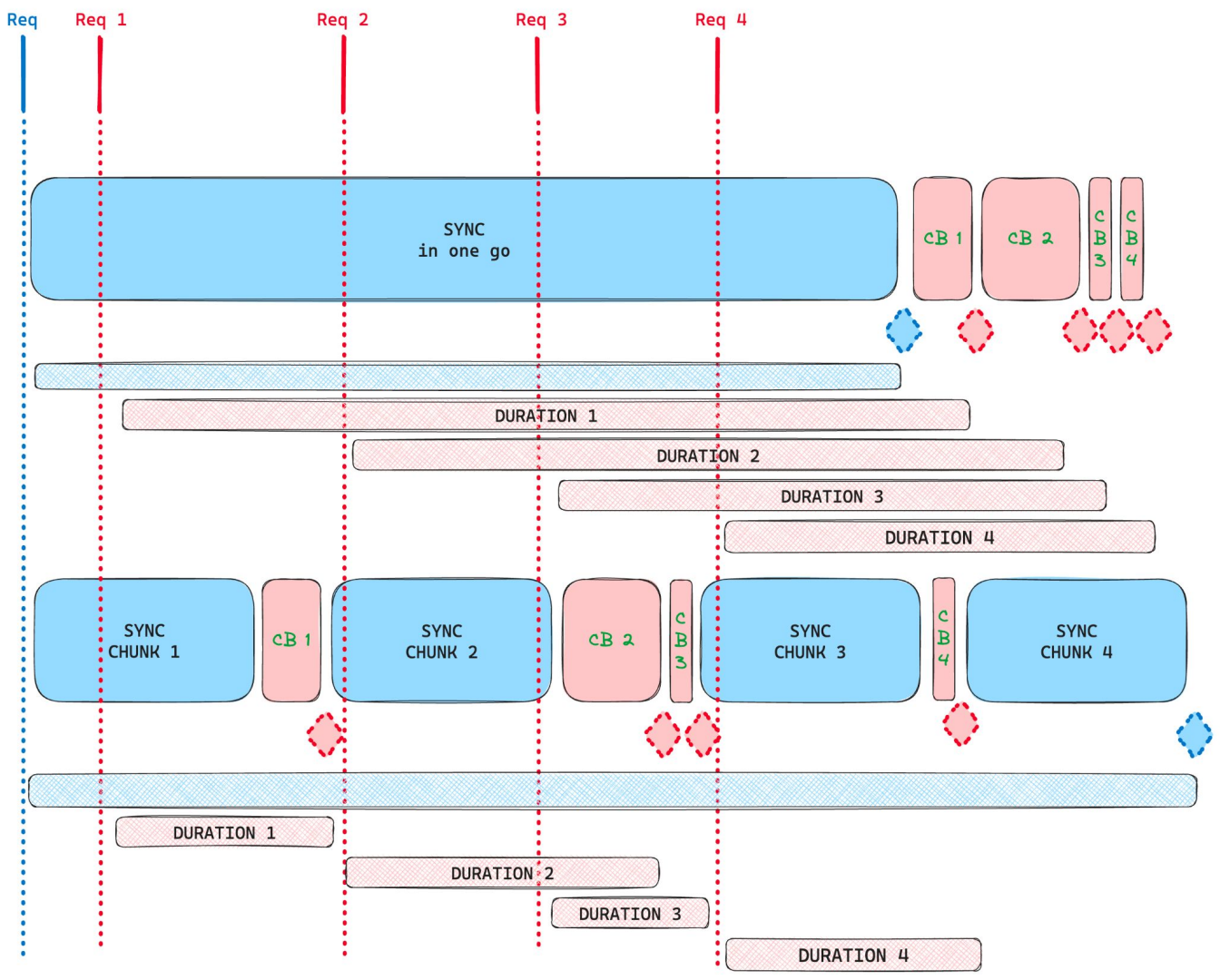



TIME
Req
Req 1
Req 2
Req 3
Req 4

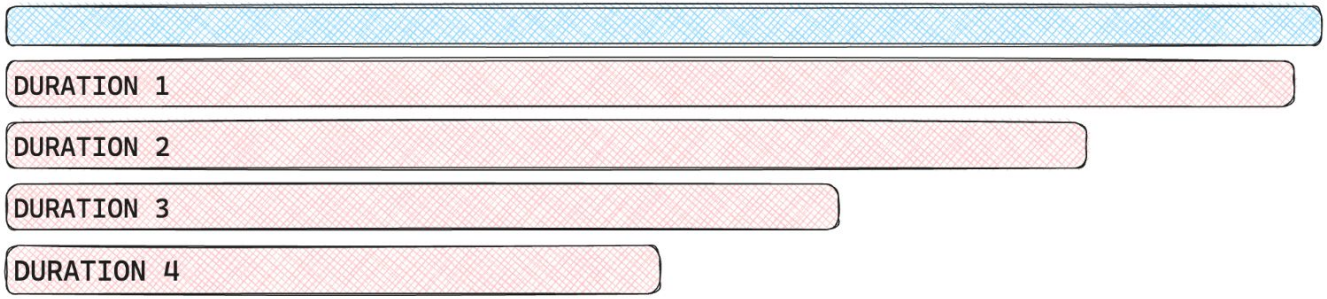

SYNC
in one go
CB 1
CB 2
CB 3
CB 4


SYNC
CHUNK 1
CB 1
SYNC
CHUNK 2
CB 2
CB 3
SYNC
CHUNK 3
CB 4
SYNC
CHUNK 4


Req
Req 1
Req 2
Req 3
Req 4
SYNC
in one go
CB 1
CB 2
CB 3
CB 4
SYNC
CHUNK 1
CB 1
SYNC
CHUNK 2
CB 2
CB 3
SYNC
CHUNK 3
CB 4
SYNC
CHUNK 4


Req
Req 1
Req 2
Req 3
Req 4
SYNC
in one go
CB 1
CB 2
CB 3
CB 4
DURATION 1
DURATION 2
DURATION 3
DURATION 4
SYNC
CHUNK 1
CB 1
SYNC
CHUNK 2
CB 2
CB 3
SYNC
CHUNK 3
CB 4
SYNC
CHUNK 4
DURATION 1
DURATION 2
DURATION 3
DURATION 4

While bcryptjs ***async*** is an improvement, it still ***blocks***!



bcryptjs SYNC

bcryptjs ASYNC

Which bcrypt library should you choose?

Repository

github.com/dcodeIO/bcrypt.js

Homepage

github.com/dcodeIO/bcrypt.js#readme

Weekly Downloads

1,896,411

Pure javascript
Known as ***bcryptjs*** on npm

Repository

github.com/kelektiv/node.bcrypt.js

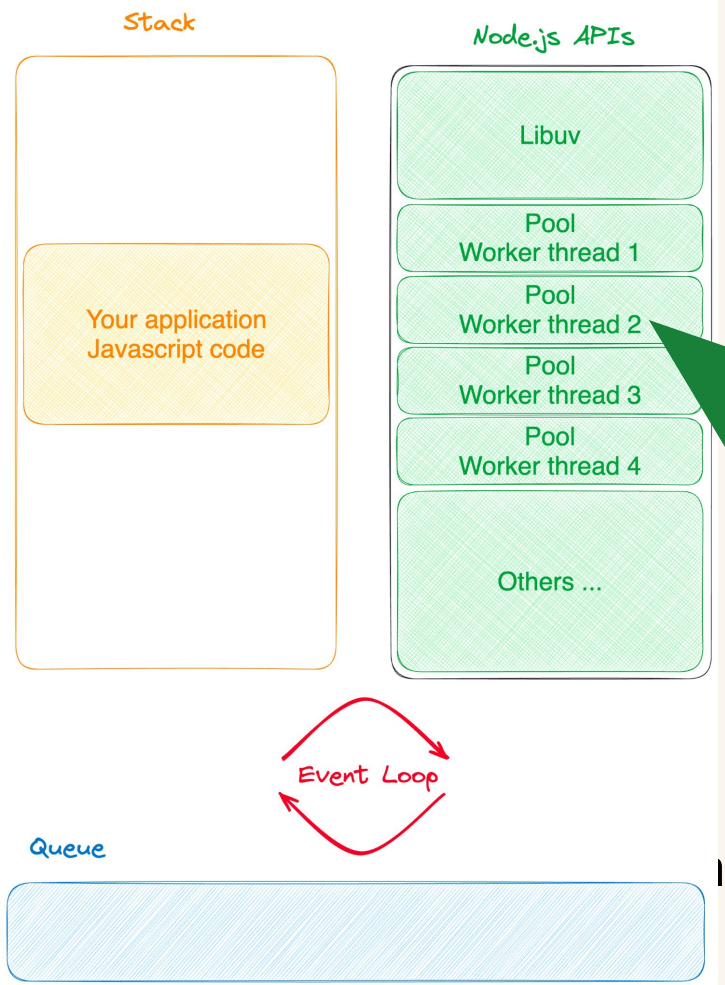
Homepage

github.com/kelektiv/node.bcrypt.js#rea...

Weekly Downloads

1,475,183

C++ implementation
Known as ***bcrypt*** on npm



Repository

github.com/kelektiv/node.bcrypt.js

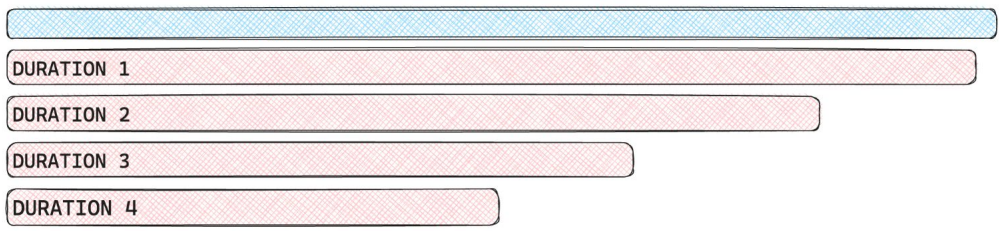
Homepage

github.com/kelektiv/node.bcrypt.js#rea...

Weekly Downloads

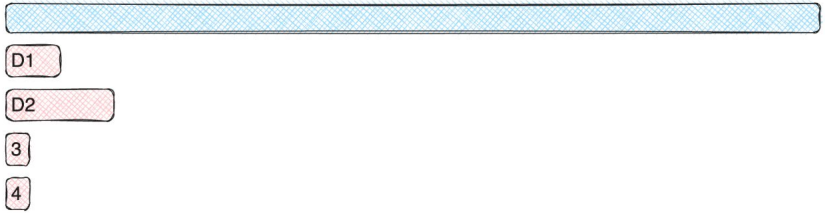
1,475,183

C++ implementation
Known as ***bcrypt*** on npm



bcryptjs SYNC

bcryptjs ASYNC



bcrypt ASYNC C++

While bcrypt.js is compatible to the C++ bcrypt binding, it is written in pure JavaScript and thus slower (about 30%), effectively reducing the number of iterations that can be processed in an equal time span.

Computing recurrent events can be slow

```
import { rrulestr } from 'rrule';

function rruleGenerateDates() {
  const recurence = `DTSTART;TZID=Europe/Brussels:20210324T110000
    \nRRULE:INTERVAL=1;FREQ=WEEKLY;BYDAY=WE,TH`;

  const rule = rrulestr(recurence);

  const start = new Date('2022-04-04T10:00');
  const end = new Date('2023-04-04T11:00');
  const dates = rule.between(start, end);
  return dates;
}
```

Between 4 … 40 ms

https://github.com/jkbrzt/rrule/issues/580

If you have no other choice …

https://www.freepik.com/free-photo/beautiful-tropical-beach-sea-ocean-with-coconut-palm-tree-sunrise-time_3531876.htm

… bring your swimsuit!

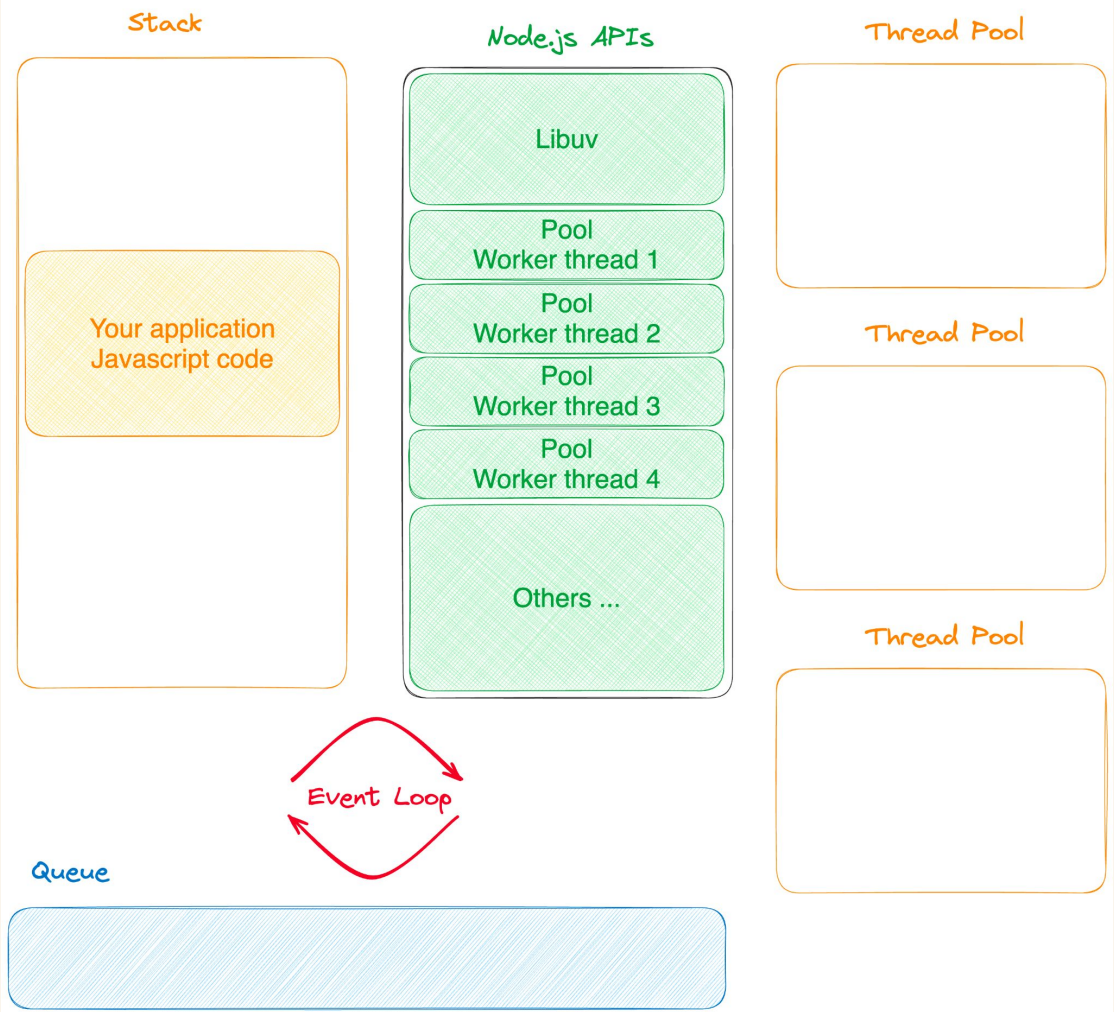
piscinajs/piscina

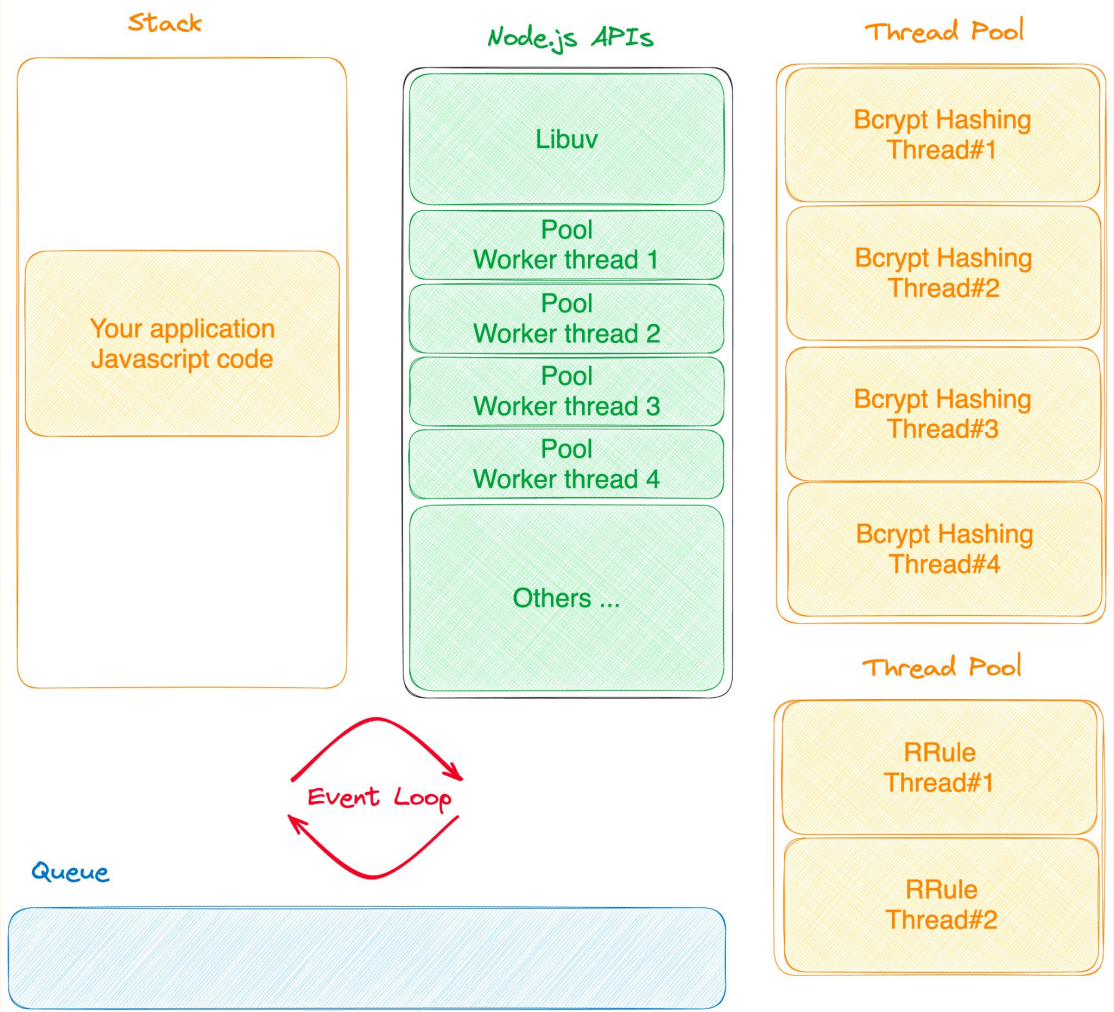
- ✔ Fast communication between threads
- ✔ Covers both fixed-task and variable-task scenarios
- ✔ Supports flexible pool sizes
- ✔ Proper async tracking integration
- ✔ Tracking statistics for run and wait times
- ✔ Cancellation Support
- ✔ Supports enforcing memory resource limits
- ✔ Supports CommonJS, ESM, and TypeScript
- ✔ Custom task queues
- ✔ Optional CPU scheduling priorities on Linux

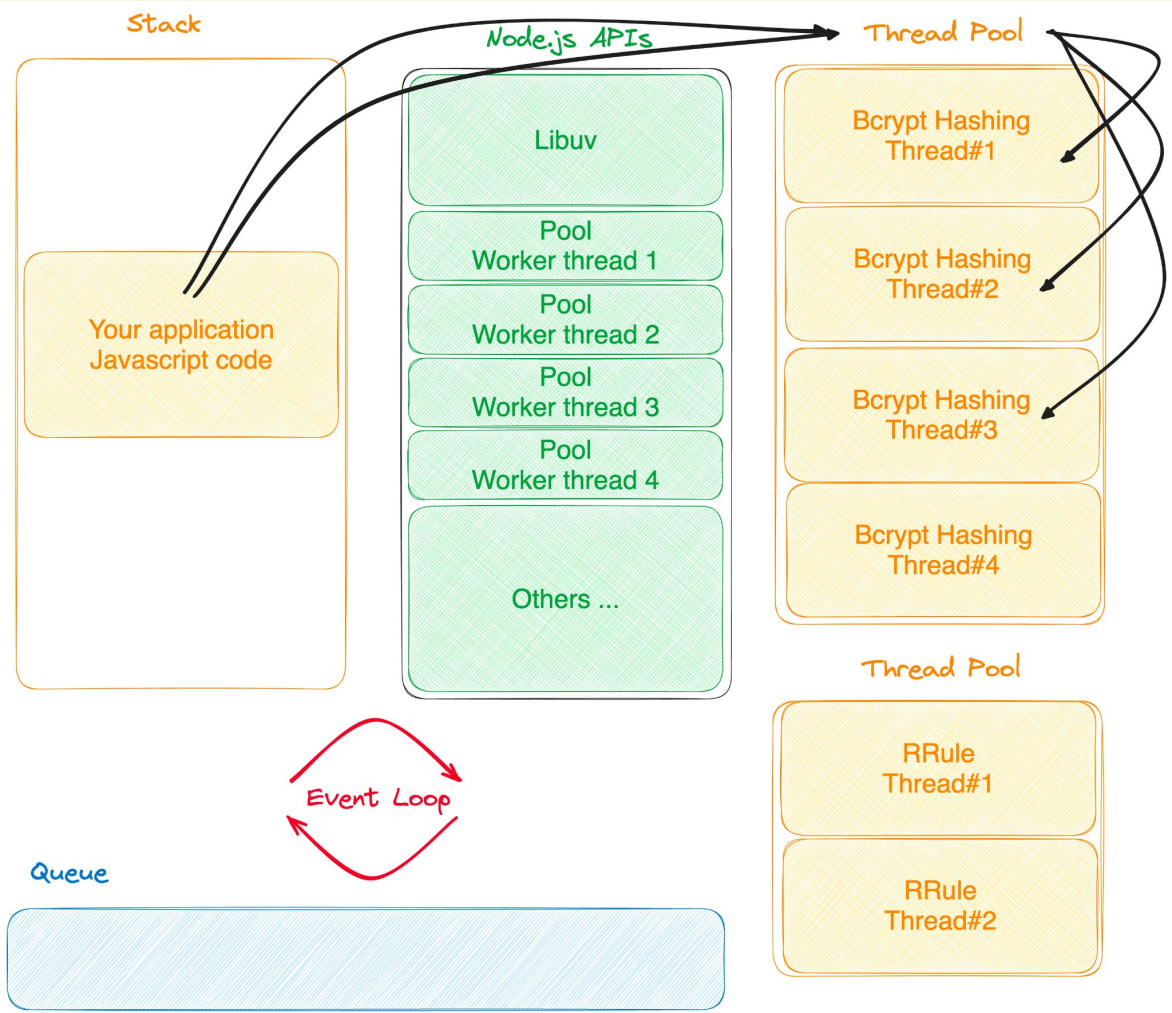
https://github.com/piscinajs/piscina

https://github.com/piscinajs/piscina

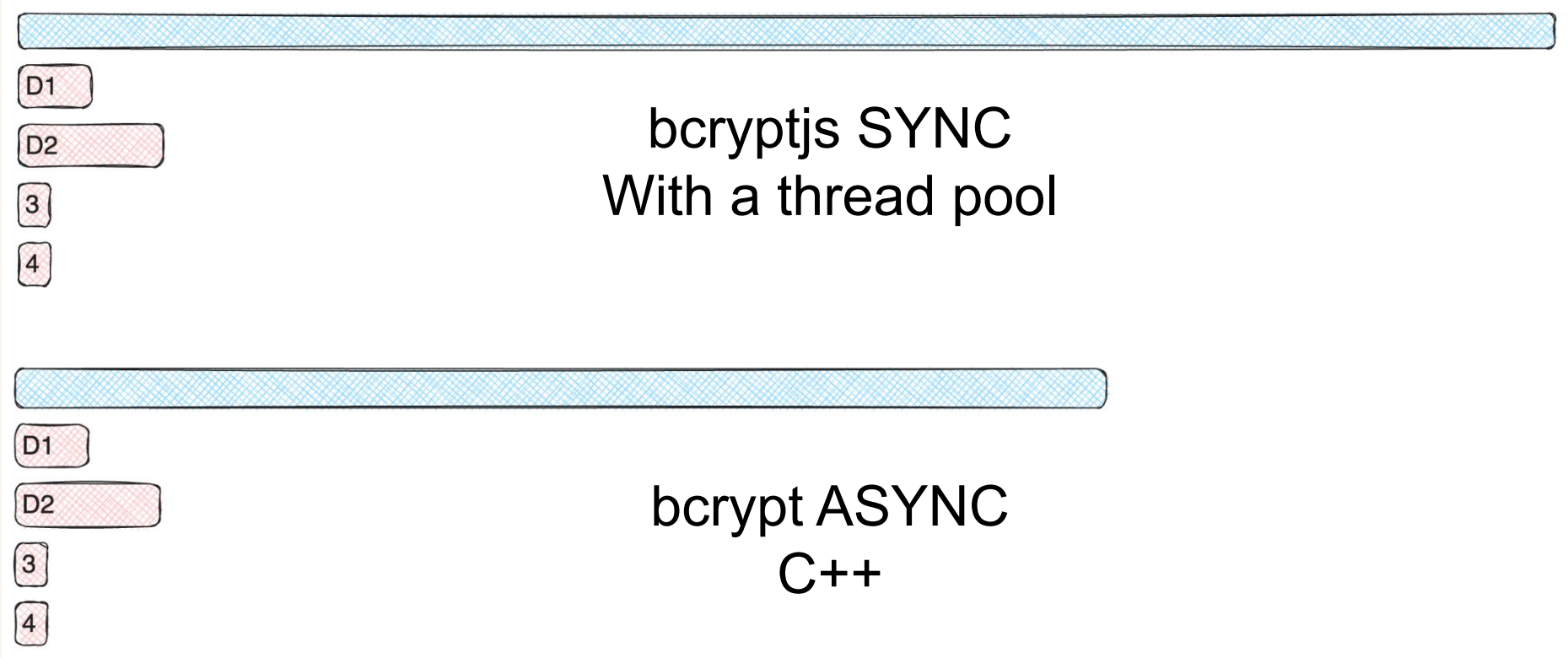
```
const path = require('path');
const Piscina = require('piscina');

const piscina = new Piscina({
  filename: path.resolve(__dirname, 'worker.js')
});

(async function() {
  const result = await piscina.run(input);
  console.log(result);
})();
```


Stack
Your application
Javascript code
Node.js APIs
Libuv
Pool
Worker thread 1
Pool
Worker thread 2
Pool
Worker thread 3
Pool
Worker thread 4
Others ...
Thread Pool
Thread Pool
Thread Pool
Event Loop
Queue


Stack
Your application
Javascript code
Node.js APIs
Libuv
Pool
Worker thread 1
Pool
Worker thread 2
Pool
Worker thread 3
Pool
Worker thread 4
Others ...
Thread Pool
Bcrypt Hashing
Thread#1
Bcrypt Hashing
Thread#2
Bcrypt Hashing
Thread#3
Bcrypt Hashing
Thread#4
Thread Pool
RRule
Thread#1
RRule
Thread#2
Event Loop
Queue


Stack
Node.js APIs
Thread Pool
Your application
Javascript code
Libuv
Pool
Worker thread 1
Pool
Worker thread 2
Pool
Worker thread 3
Pool
Worker thread 4
Others ...
Bcrypt Hashing
Thread#1
Bcrypt Hashing
Thread#2
Bcrypt Hashing
Thread#3
Bcrypt Hashing
Thread#4
Thread Pool
RRule
Thread#1
RRule
Thread#2
Event Loop
Queue

A thread pool prevents blocking the event loop



Thread pools are not a silver bullet

Adjust Thread Pool Size
- UV_THREADPOOL_SIZE
- Piscina pools

Distribute Workload Evenly

Match CPU Cores
- # threads
- # vcpus / # cores

Check memory usage

Monitor Event Loop

https://www.pragspective.com/from-the-desk-of/silver-bullet/

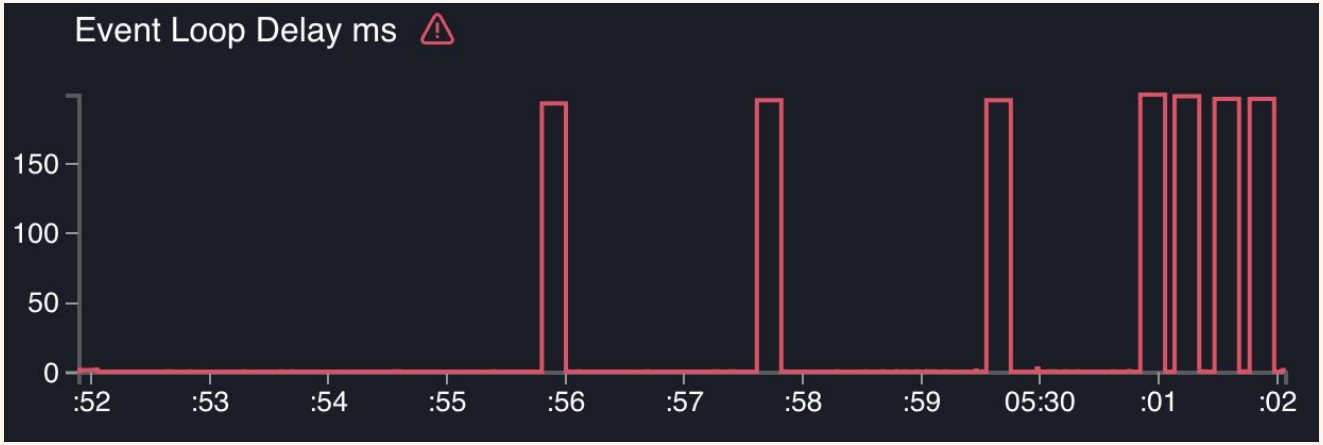
Observability & Telemetry to the rescue

Event Loop Delay

the time difference between
when an event loop tick is scheduled
when it actually begins execution

Max CPU time per tick

maximum time spent in a single tick
per minute



https://github.com/clinicjs/node-clinic

```javascript
const timer = process.hrtime.bigint;
setInterval(() => {
  const start = timer();
  setImmediate(() => {
    const end = timer();
    console.log(`delay of ${end - start} ns`);
  });
}, 1000);
```

Multithreaded Javascript - Concurrency beyond the event loop
Oreilly, Thomas Hunter & Bryan English

Don’t **block** the **event loop**!

Do ***NOT*** perform ***CPU intensive*** tasks on the ***main thread***

Have some Coffee

Enjoy the Show

Take some time off

Thank you FOSDEM

Antoine Pairet

Co-founder & CTO

@antoinepairet

ROSA