JONAS.IO
SCHMEDTMANN

NODE.JS, EXPRESS & MONGODB

THE COMPLETE BOOTCAMP

@JONASSCHMEDTMAN

SECTION

INTRODUCTION TO NODE.JS

LECTURE

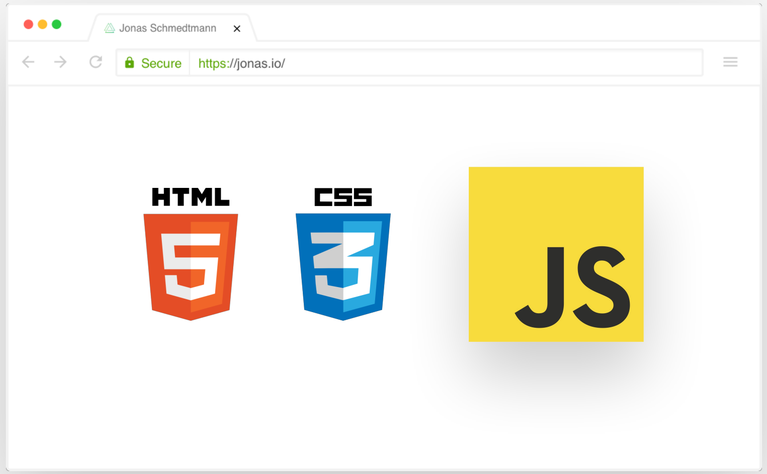
WHAT IS NODE.JS AND WHY USE IT?

WHAT IS NODE.JS?

NODE.JS

NODE.JS IS A JAVASCRIPT RUNTIME BUILT ON GOOGLE'S OPEN-SOURCE V8 JAVASCRIPT ENGINE. 🤔

NODE.JS: JAVASCRIPT OUTSIDE OF THE BROWSER



BROWSER

NODE.JS

JAVASCRIPT ON THE SERVER!

Perfect conditions for using Node.js as a web server

We can use JavaScript on the server-side of web development 😄

Build fast, highly scalable network applications (back-end)

WHY AND WHEN TO USE NODE.JS?

NODE.JS PROS

- Single-threaded, based on event driven, non-blocking I/O model 🤯😅
- Perfect for building **fast** and **scalable** data-intensive apps;
- Companies like NETFLIX UBER PayPal ebay have started using node in production;
- **JavaScript across the entire stack:** faster and more efficient development;
- **NPM:** huge library of open-source packages available for everyone for free;
- **Very active** developer community.

USE NODE.JS

- API with database behind it (preferably NoSQL);
- Data streaming (think YouTube);
- Real-time chat application;
- Server-side web application.

DON'T USE

- Applications with heavy server-side processing (CPU-intensive).

RAILS php

JONAS.IO
SCHMEDTMANN

NODE.JS, EXPRESS & MONGODB

THE COMPLETE BOOTCAMP

@JONASSCHMEDTMAN

SECTION

INTRODUCTION TO NODE.JS

LECTURE

BLOCKING AND NON-BLOCKING:
ASYNCHRONOUS NATURE OF NODE.JS

SYNCHRONOUS VS. ASYNCHRONOUS CODE (BLOCKING VS. NON-BLOCKING)

```
const fs = require('fs');

// Blocking code execution
const input = fs.readFileSync('input.txt', 'utf-8');
console.log(input);
```

SYNCHRONOUS

BLOCKING 👎

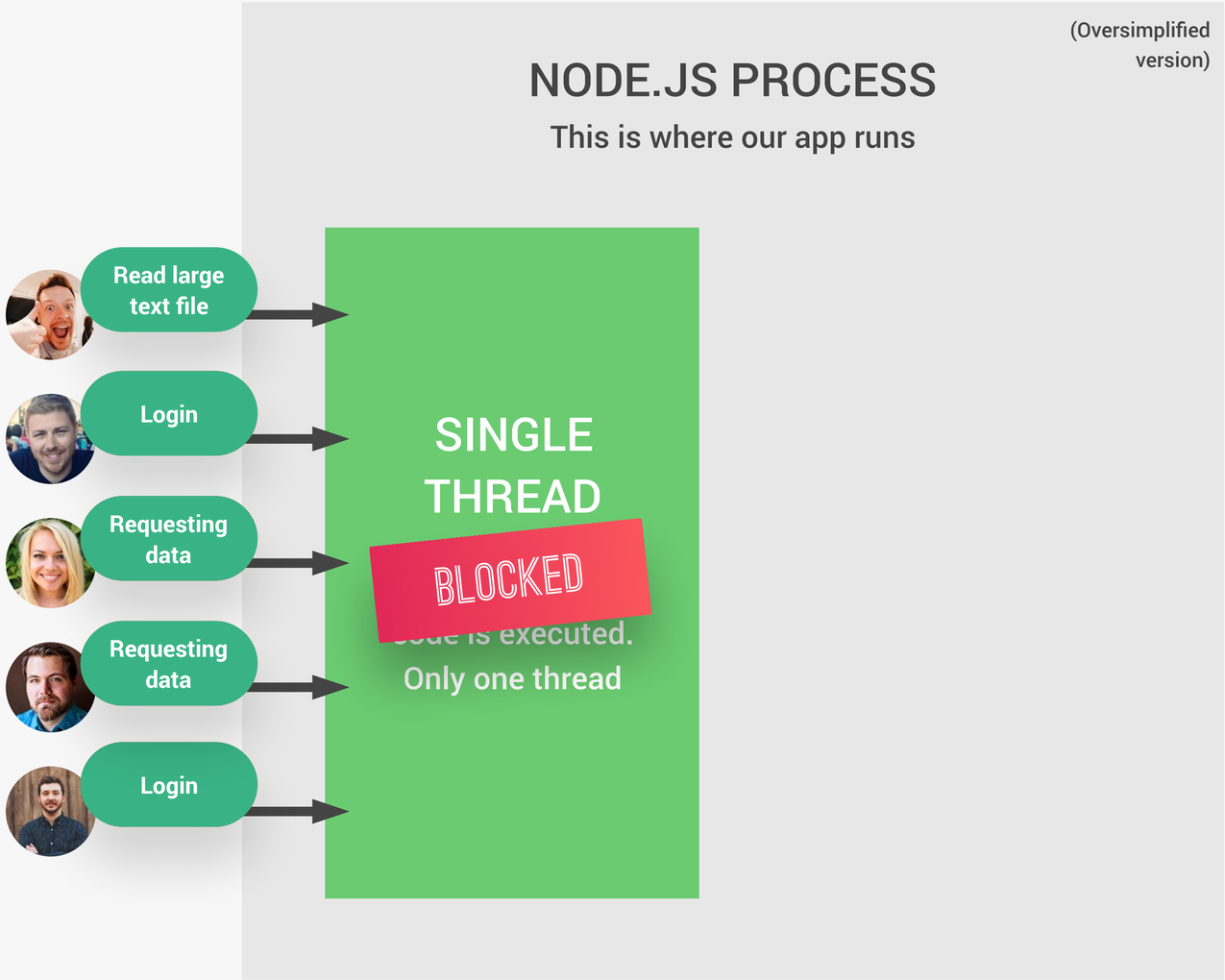
```
const fs = require('fs');

// Non-blocking code execution
fs.readFile('input.txt', 'utf-8', (err, data) => {
    console.log(data);
});
console.log('Reading file...');
```

ASYNCHRONOUS

NON-BLOCKING 👍

THE ASYNCHRONOUS NATURE OF NODE.JS: AN OVERVIEW



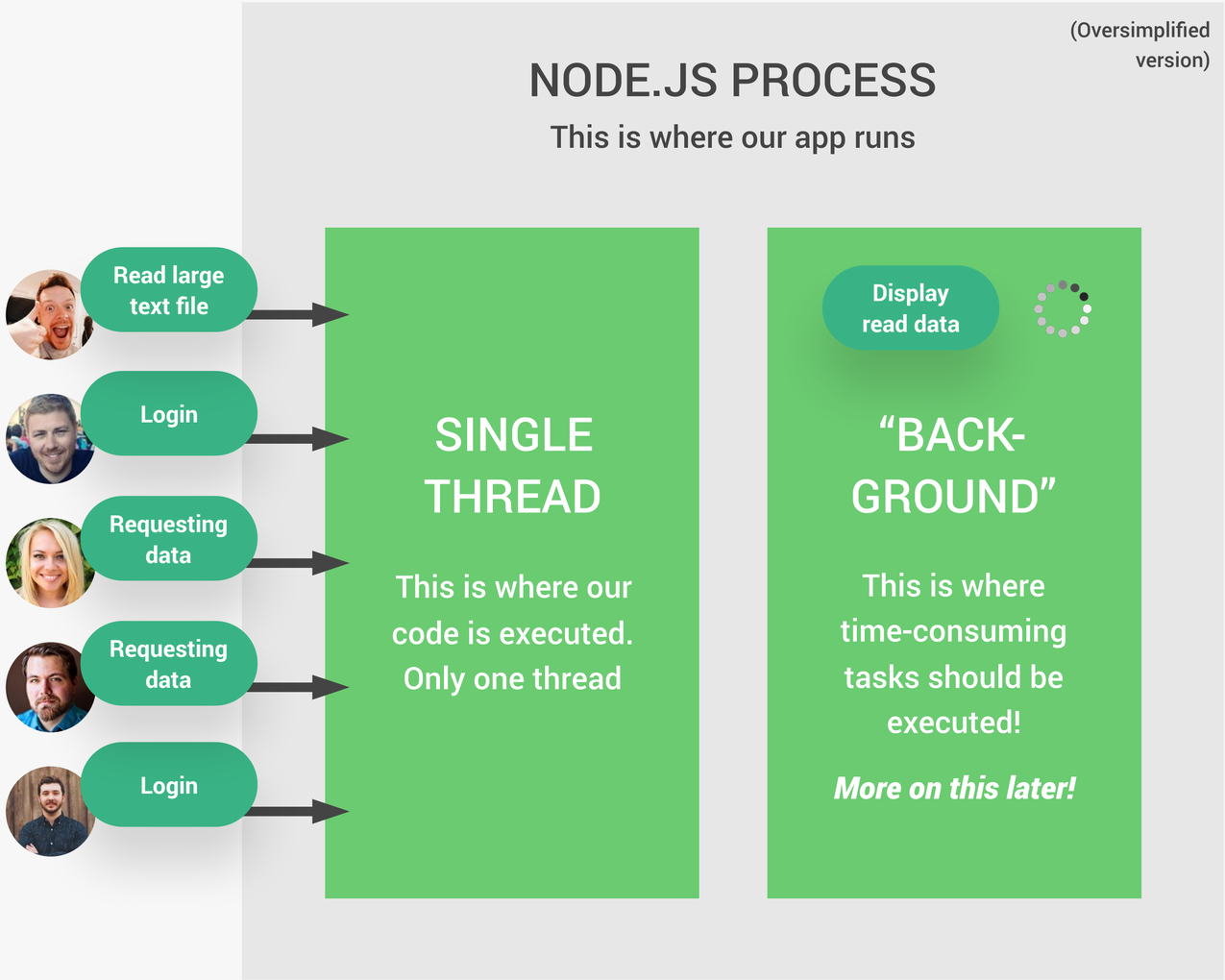
SYNCHRONOUS WAY

```
const fs = require('fs');

// Blocking code execution
const input = fs.readFileSync('input.txt', 'utf-8');
console.log(input);
```

👉 It's **YOUR** job as a developer to avoid this kind of situation!

THE ASYNCHRONOUS NATURE OF NODE.JS: AN OVERVIEW



ASYNCHRONOUS WAY

```
const fs = require('fs');

// Non-blocking code execution
fs.readFile('input.txt', 'utf-8', (err, data) => {
    console.log(data);
});
console.log('Reading file...');
```

👉 Non-blocking I/O model

👉 This is why we use so many callback functions in Node.js

👉 Callbacks ≠ Asynchronous

THE PROBLEM: CALLBACK HELL...

CALLBACK HELL

```
const fs = require('fs');

fs.readFile('start.txt', 'utf-8', (err, data1) => {
    fs.readFile(`${data1}.txt`, 'utf-8', (err, data2) => {
        fs.readFile('append.txt', 'utf-8', (err, data3) => {
            fs.writeFile('final.txt', `${data2} ${data3}`, 'utf-8', (err) => {
                if (err) throw err;
                console.log('Your file has been saved :D');
            });
        });
    });
});
```

👉 **SOLUTION:** Using Promises or Async/Await [Optional Section]