RACE.

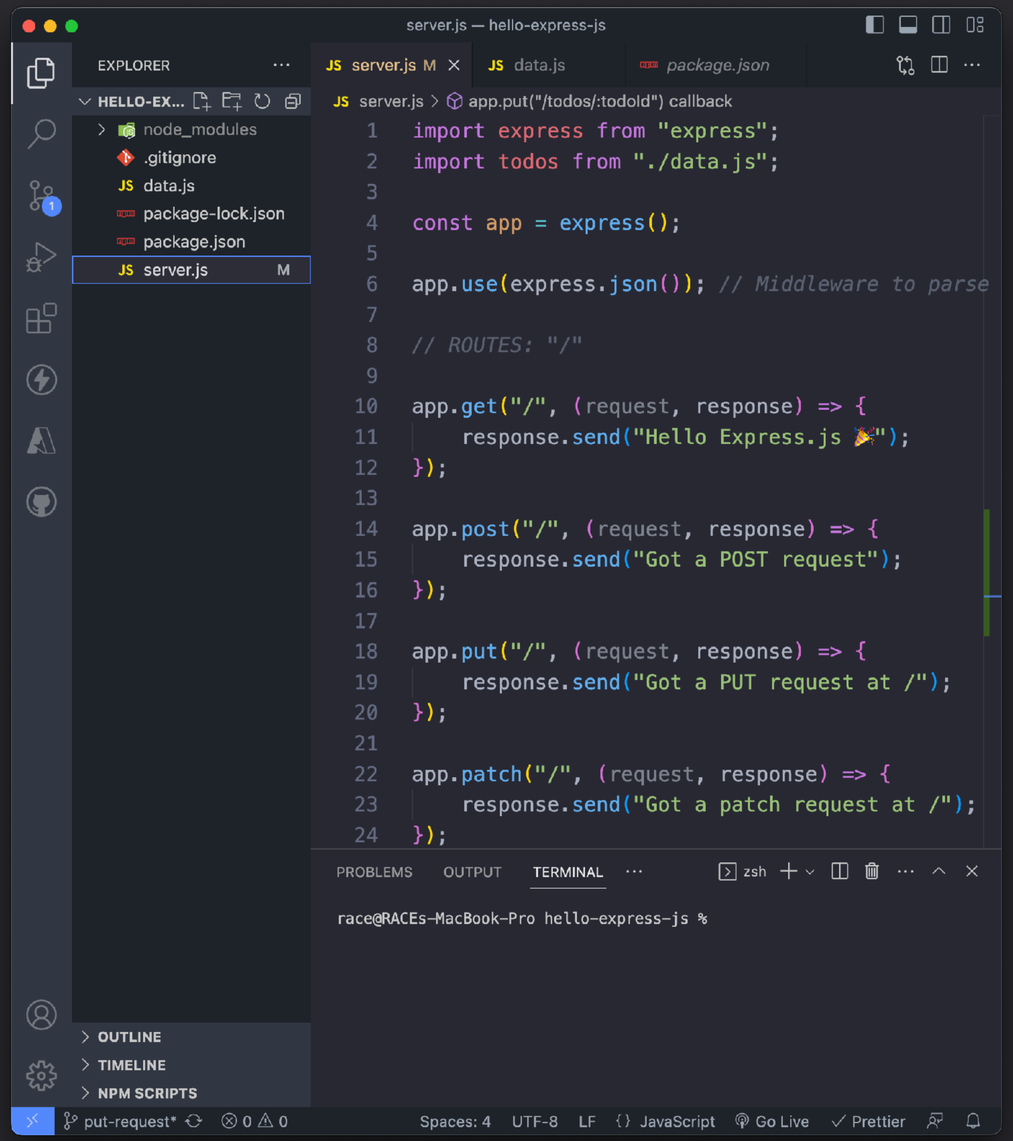
ERHVERVSAKADEMI AARHUS
BUSINESS ACADEMY AARHUS

Agenda

1. Hej og velkommen!
2. Client-server & request/response cyklussen
3. Intro til Node.js
4. Intro til Express
5. GET og POST requests
6. Hands-on: Opgaver

Opgaver

- Hello Node.js
- Hello HTTP Module (uden Express.js)
- Getting Started with Express.js
- Fortsæt på Scrimba-kursus Learn Node.js
- Ekstraopgaver
  - Postman - Getting Started
  - Node.js File System
  - Introduction to Backend Development with Node.js & Express.js



I’m Rasmus Cederdorff (RACE)

Senior Lecturer
Freelance Web App Developer

- Programming with an eye for UI and UX.
- Web Development, JavaScript, React, BaaS & Node.js
- “I speak JavaScript”.
- Websites, Webshops, Web Apps, Mobile Apps, Server App and BaaS.

Computer Science
EAAA
2011

Web Development (PBA)
EAAA
2013

Master of Science in IT
SDU
2015

2013
App Developer
Visiolink ApS

2014
Freelance Web App Developer
Cederdorff

2018
Lecturer
EAAA

2021
Senior Lecturer
EAAA

2022
Senior Lecturer
KEA

2023
Senior Lecturer
EAAA

Backend ⟷ Frontend

I’m Rasmus Cederdorff

Holstebro

Alicia & Ida

From Holstebro

I’m into sports

Love (🍎) gadgets, to take pictures & interior design projects

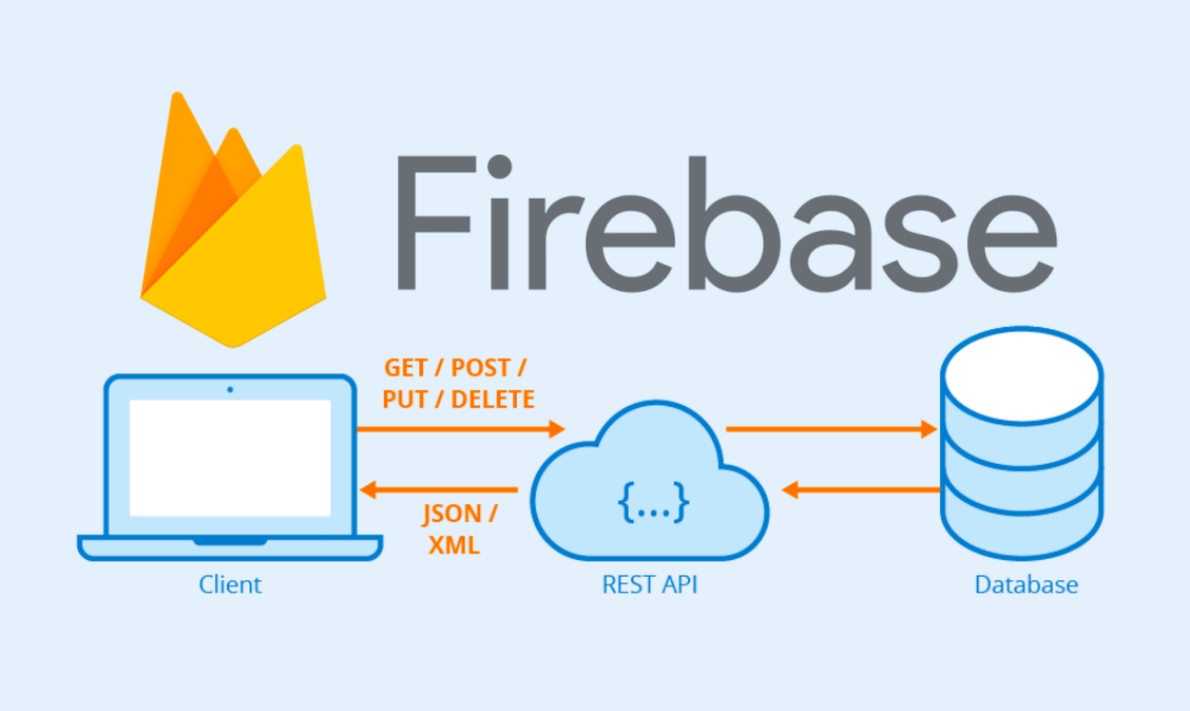
What’s with my arm?

“The cool thing about JavaScript is that you can create stuff that will put a smile on your face, as a developer.

You can create stuff and it will visually look like something, and it'll do stuff, and it makes you feel good, **it makes you fall in love with programming.**”

Lex Fridman
Computer Scientist
https://www.youtube.com/watch?v=GLhyjVZp0cw&t=252s

When someone asks you how to get data from a database in a js code


Firebase
GET / POST /
PUT / DELETE
JSON /
XML
{...}
Client
REST API
Database

Rasmus
Firebase
made with mematic

Code Every Day

Men hvorfor gør I det her ved os?

Technology | 2024 Stack Over

survey.stackoverflow.co/2024/technology/

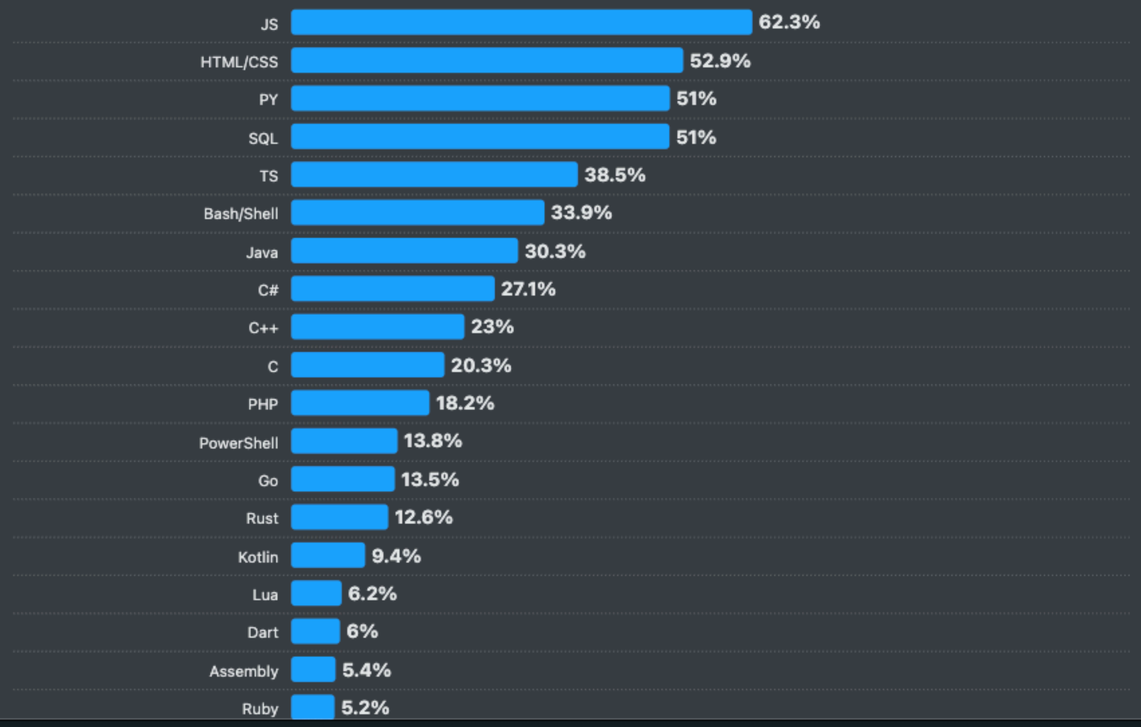
Most popular technologies 2.1

Programming, scripting, and markup languages

JavaScript has been a mainstay in the developer survey and on Stack Overflow since our first survey. The most popular programming language has been JavaScript every year we have done the survey except for 2013 and 2014, when SQL was the most popular language.

Which **programming, scripting, and markup languages** have you done extensive development work in over the past year, and which do you want to work in over the next year? (If you both worked with the language and want to continue to do so, please check both boxes in that row.)

All Respondents | Professional Developers | Learning to Code | Other Coders



https://survey.stackoverflow.co/2024/technology/#1-programming-scripting-and-markup-languages

What is the world's most popular language?
A: French
B: English
C: Spanish
D: JavaScript

Technology | 2024 Stack Ove

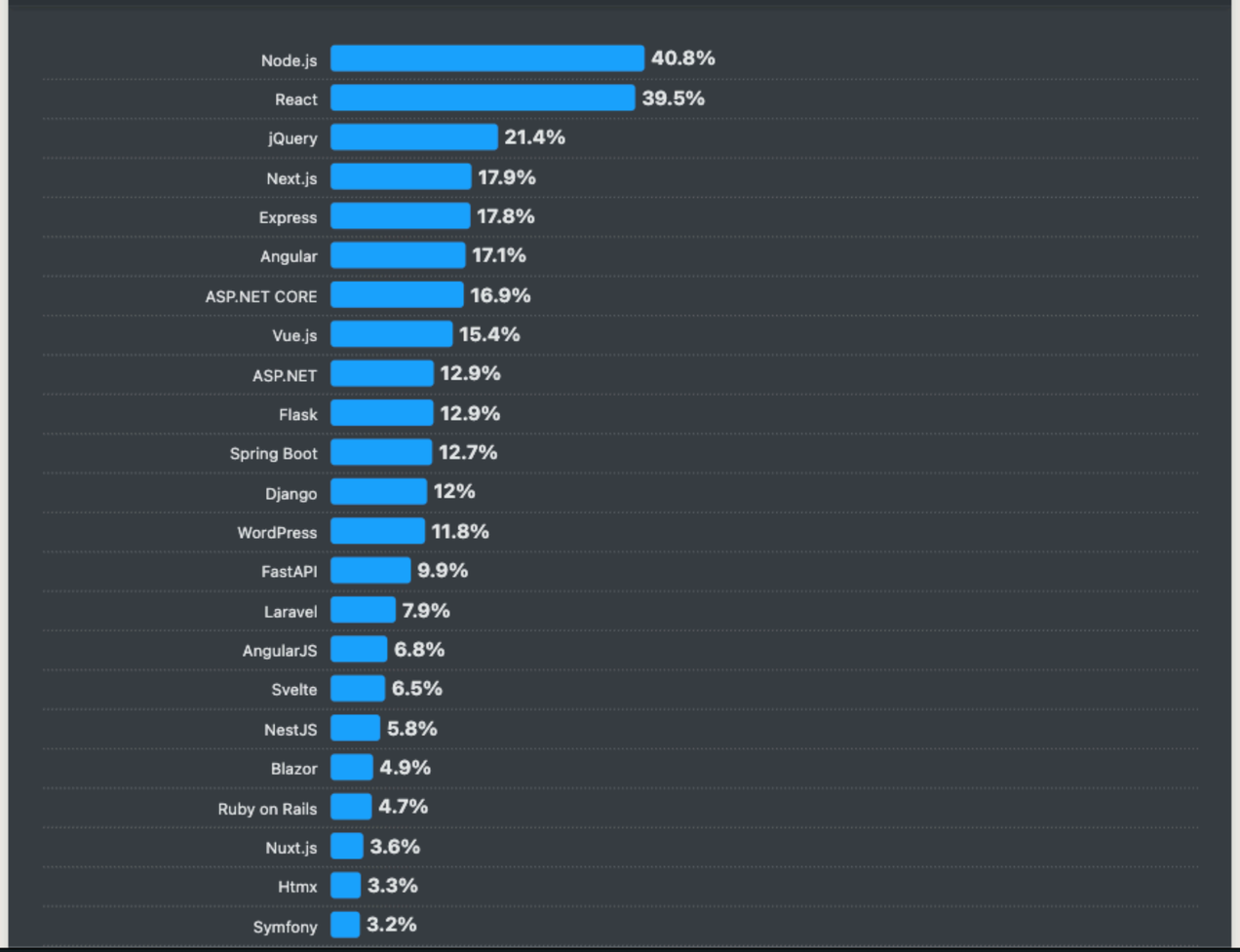
survey.stackoverflow.co/2024/technology/#1-web-frameworks-and-technologies

Web frameworks and technologies

Node.js peaked in 2020 with its highest recorded usage score of 51%. While not as popular, it's still the most used web technology in the survey this year and has increased popularity among those learning to code from last year.

Which **web frameworks and web technologies** have you done extensive development work in over the past year, and which do you want to work in over the next year? (If you both worked with the framework and want to continue to do so, please check both boxes in that row.)

All Respondents | Professional Developers | Learning to Code | Other Coders



https://survey.stackoverflow.co/2024/technology/#1-web-frameworks-and-technologies

Technology | 2024 Stack Ove

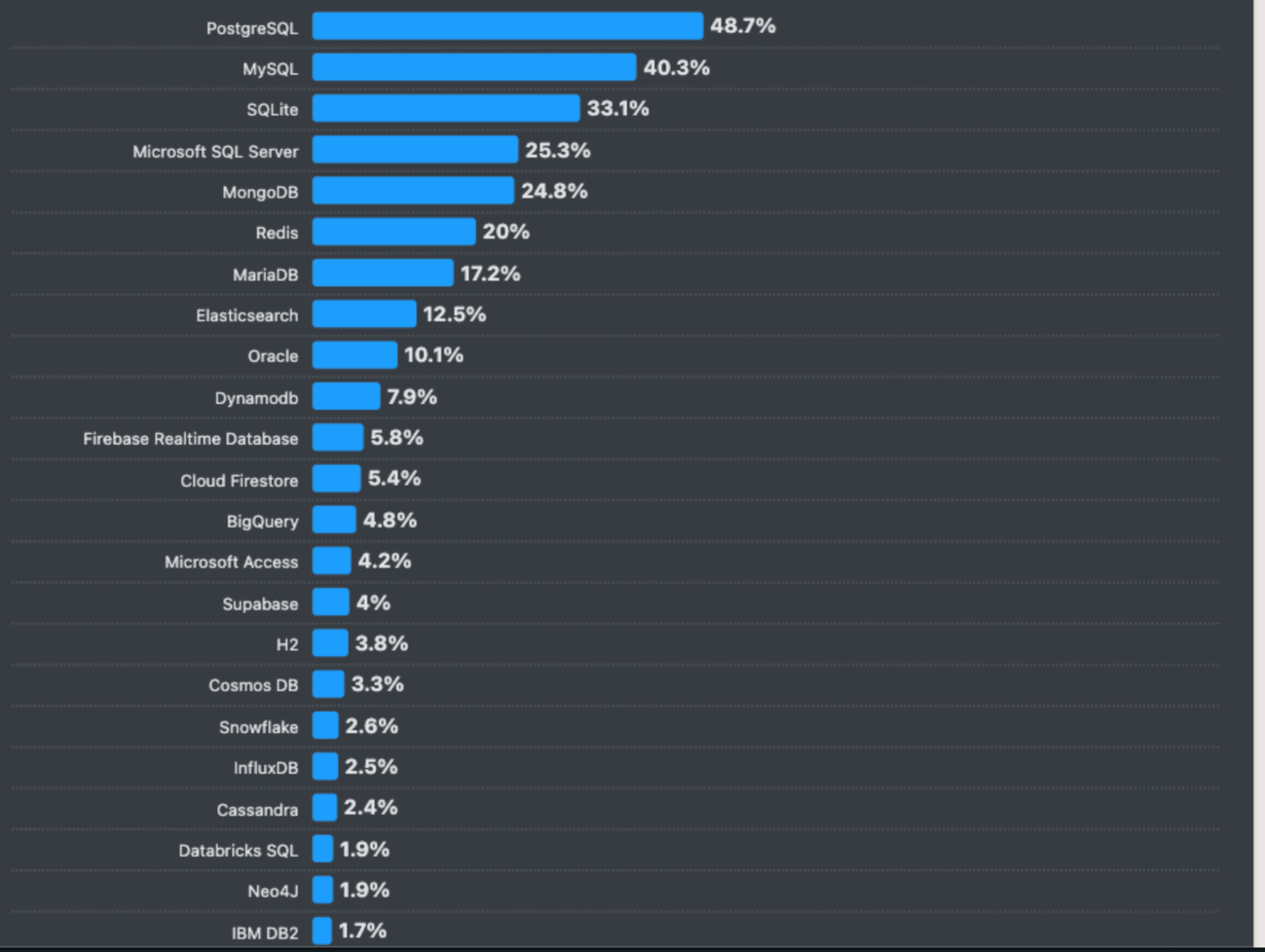
survey.stackoverflow.co/2024/technology/

Databases

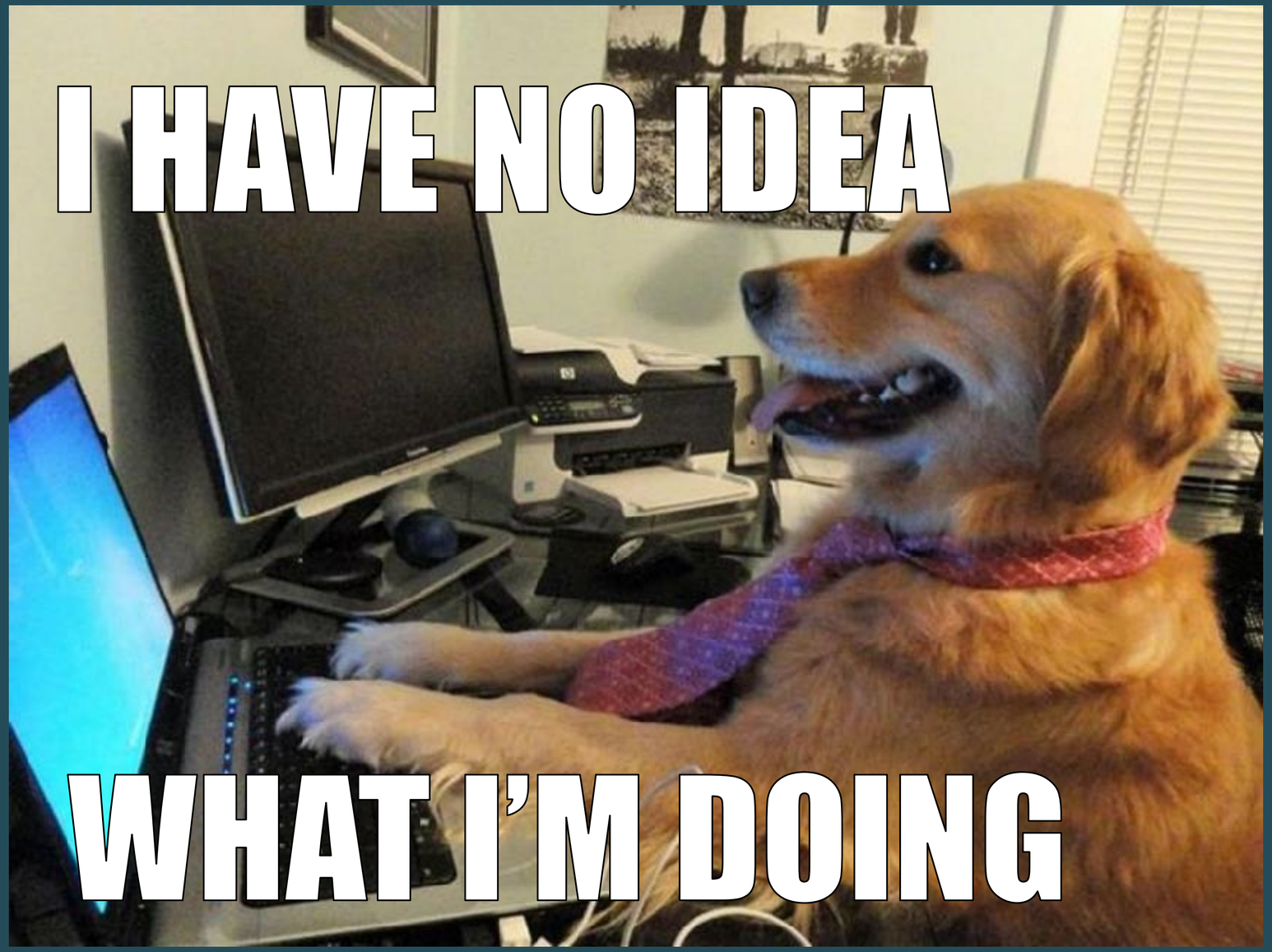
PostgreSQL debuted in the developer survey in 2018 when 33% of developers reported using it, compared with the most popular option that year: MySQL, in use by 59% of developers. Six years later, PostgreSQL is used by 49% of developers and is the most popular database for the second year in a row.

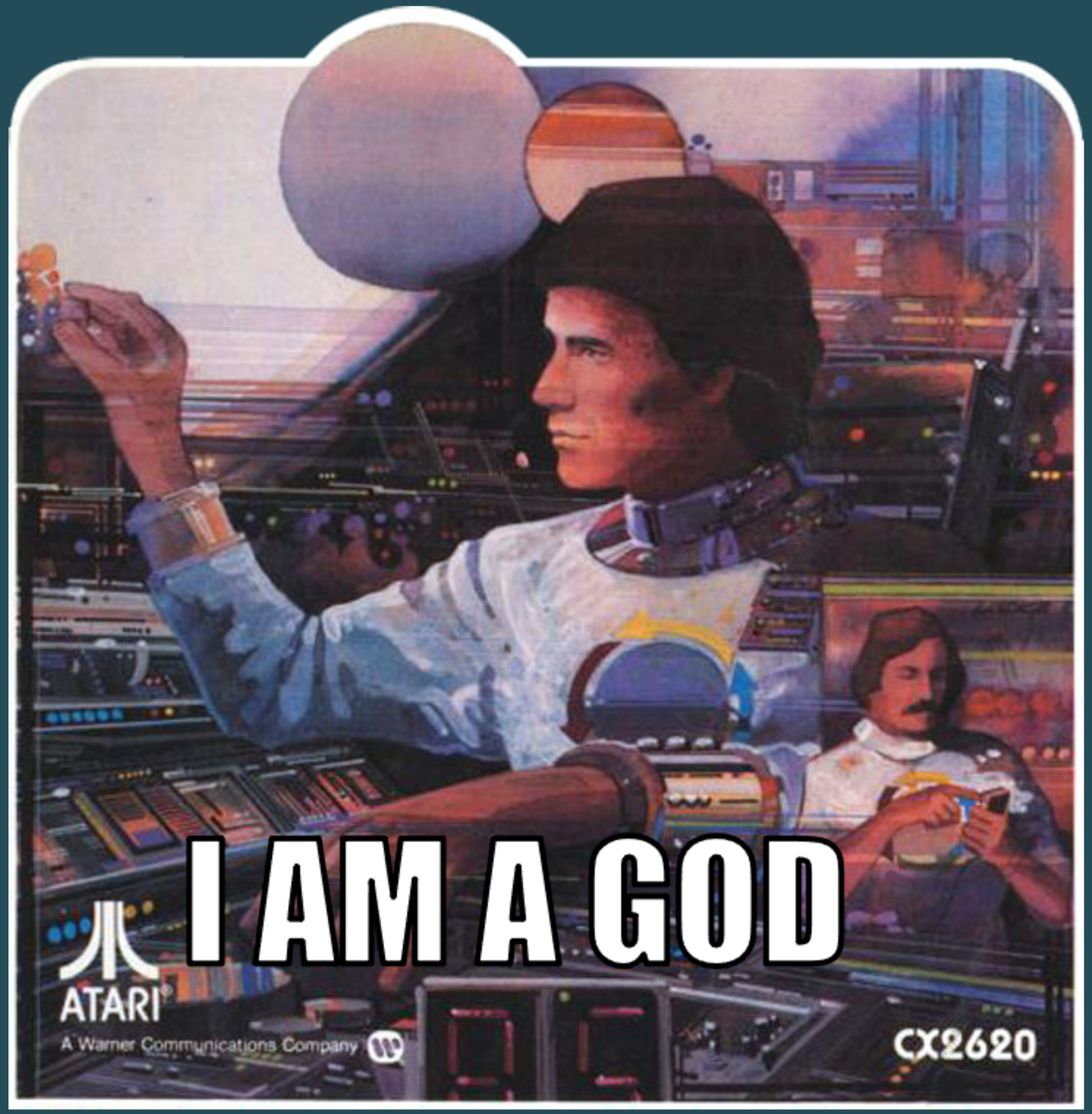
Which **database environments** have you done extensive development work in over the past year, and which do you want to work in over the next year? (If you both worked with the database and want to continue to do so, please check both boxes in that row.)

All Respondents | Professional Developers | Learning to Code | Other Coders



https://survey.stackoverflow.co/2024/technology/#1-databases


I HAVE NO IDEA
WHAT I'M DOING

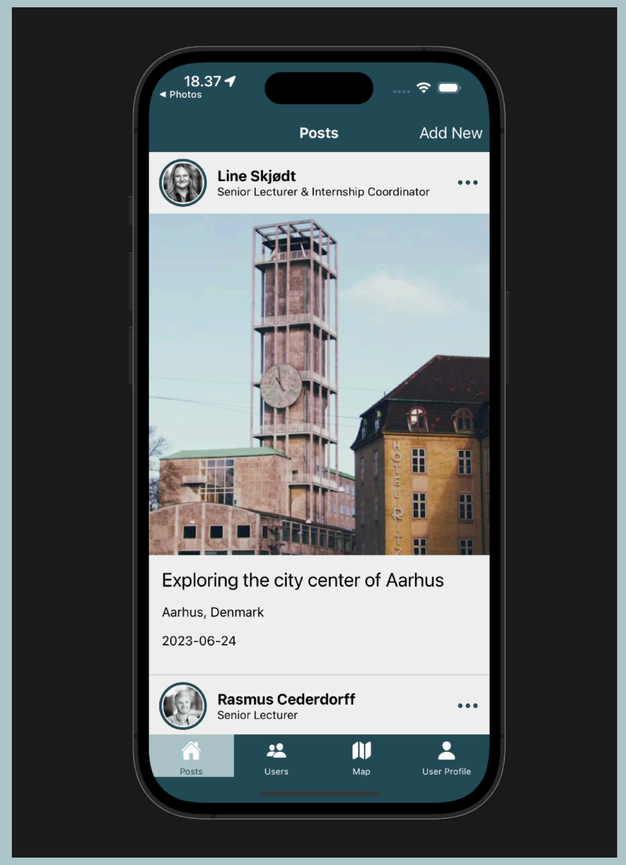

I AM A GOD
ATARI
A Warner Communications Company
CX2620

Client Server Architecture

APP

REST API

DATA SOURCE



FETCH(GET, POST, PUT, DELETE)

REQUEST

JSON DATA

RESPONSE

posts

users

CLIENT

SERVER

Client-Server Model

Communication between web **clients** and web **servers**.

CLIENT

SERVER

Client-Server Model

Communication between web **clients** and web **servers**.

Browsers
Or any type of
program or device

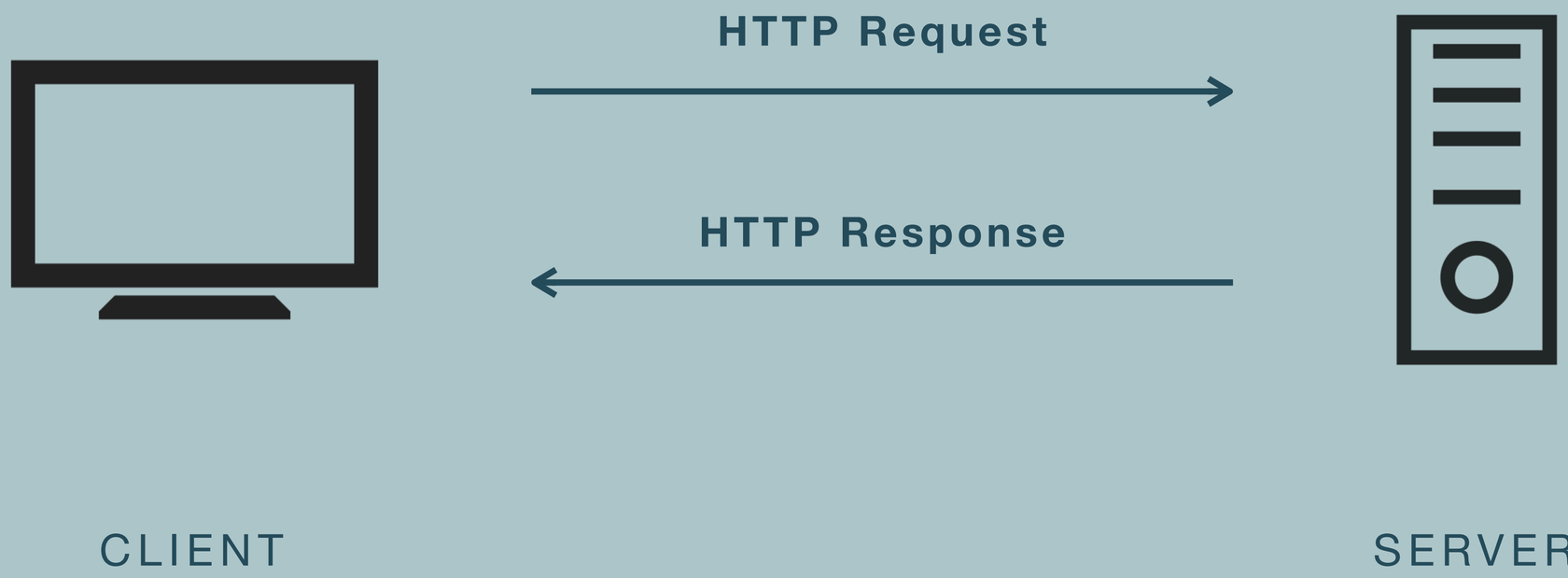
Cloud Computers
Often computers
in the cloud

CLIENT

SERVER

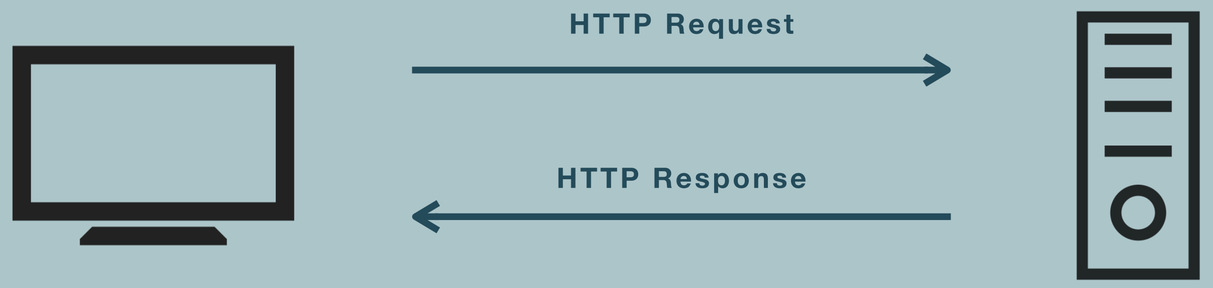
Client-Server Model

Communication between web **clients** and web **servers**.



Hyper Text Transfer Protocol

- A protocol and standard for fetching data, HTML and other resources (text, images, videos, scripts, JSON).
- The foundation of the web.



What is HTTP

Not Secure | w3schools.com/whatis/whatis_http.asp

HTML CSS JAVASCRIPT SQL PYTHON

HTTP Request / Response

Communication between clients and servers is done by **requests** and **responses**:

1. A client (a browser) sends an **HTTP request** to the web
2. A web server receives the request
3. The server runs an application to process the request
4. The server returns an **HTTP response** (output) to the browser
5. The client (the browser) receives the response

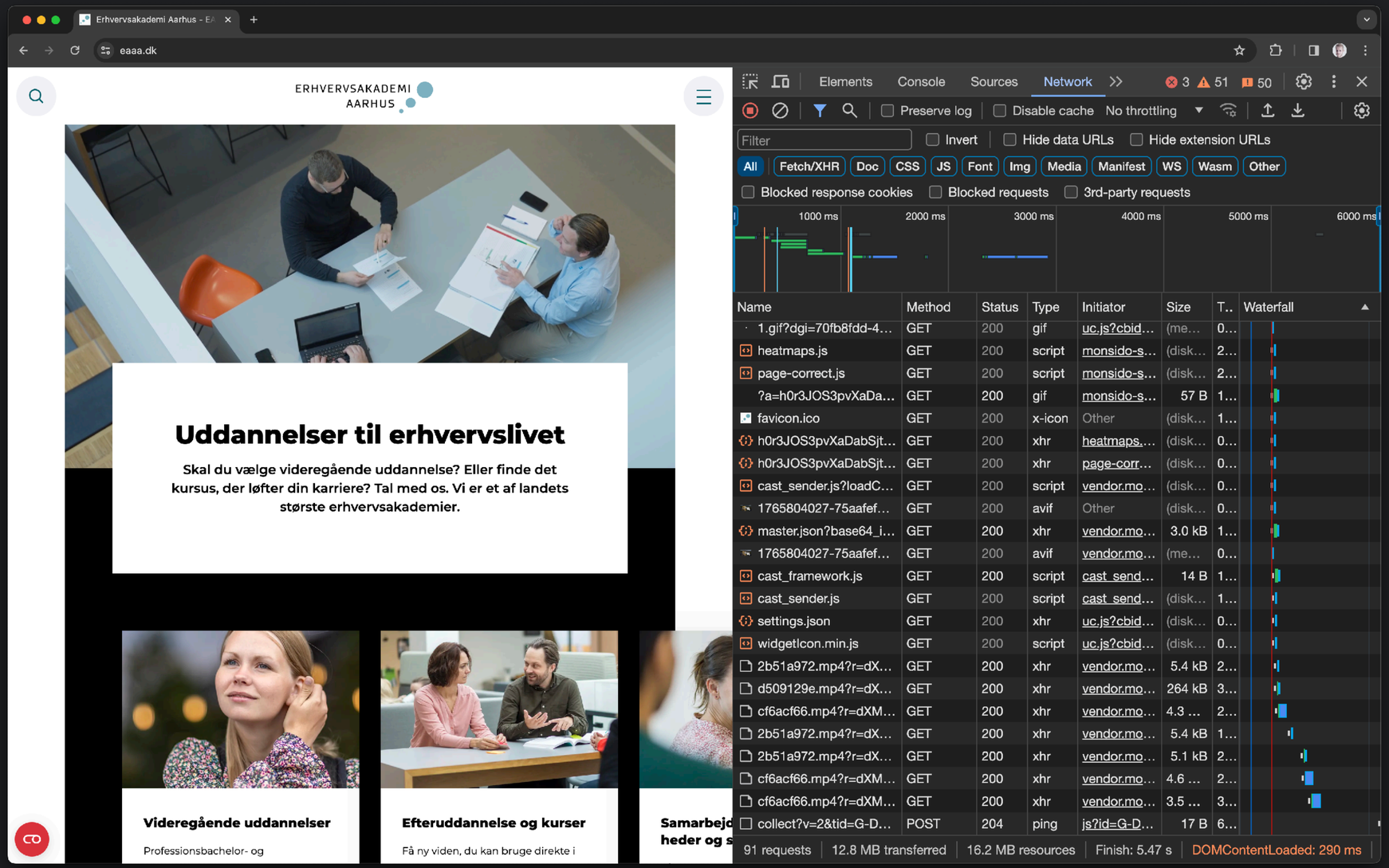
The HTTP Request Circle

A typical HTTP request / response circle:

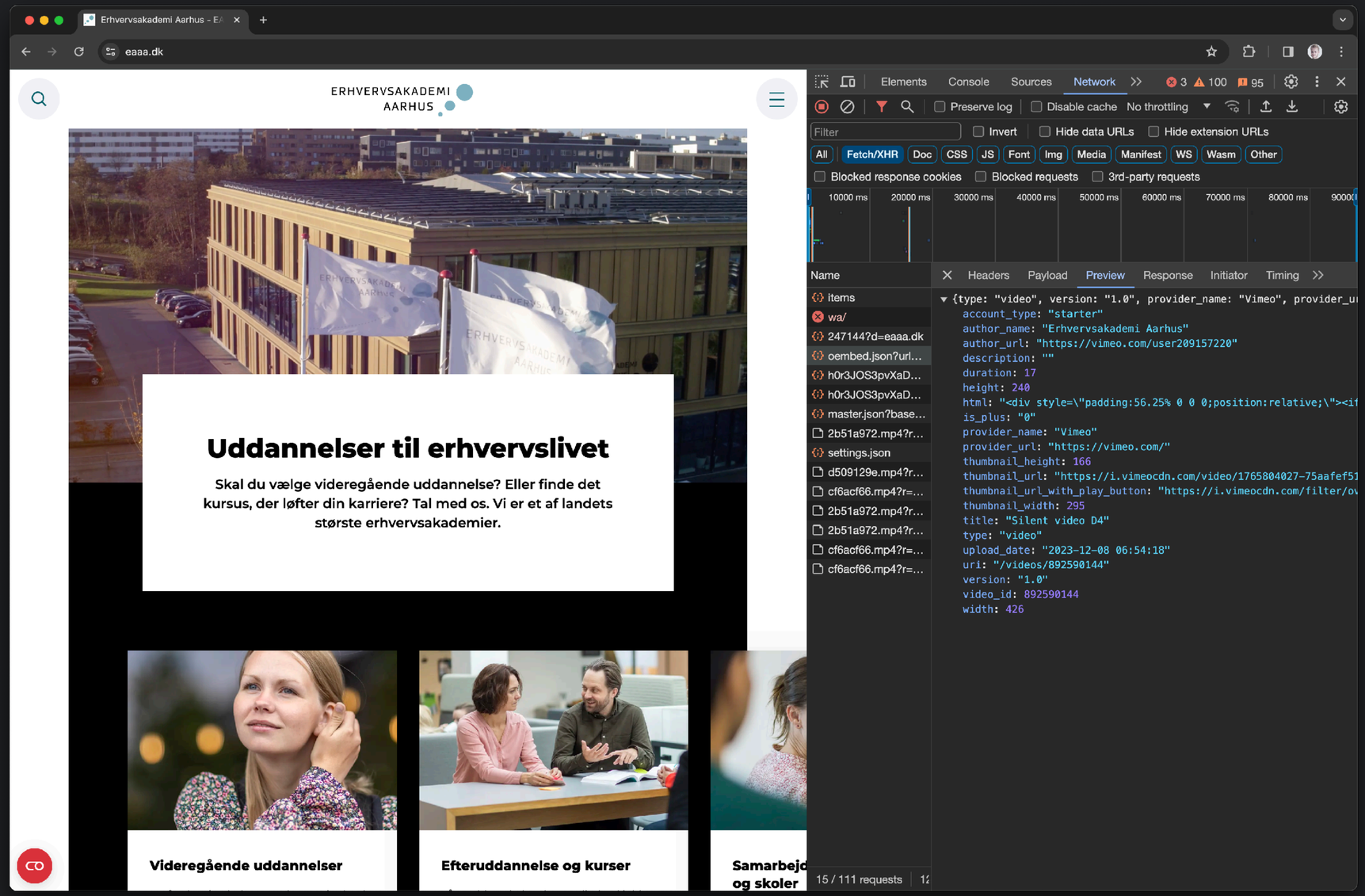
1. The browser requests an HTML page. The server returns an HTML file.
2. The browser requests a style sheet. The server returns a CSS file.
3. The browser requests an JPG image. The server returns a JPG file.
4. The browser requests JavaScript code. The server returns a JS file
5. The browser requests data. The server returns data (in XML or JSON).

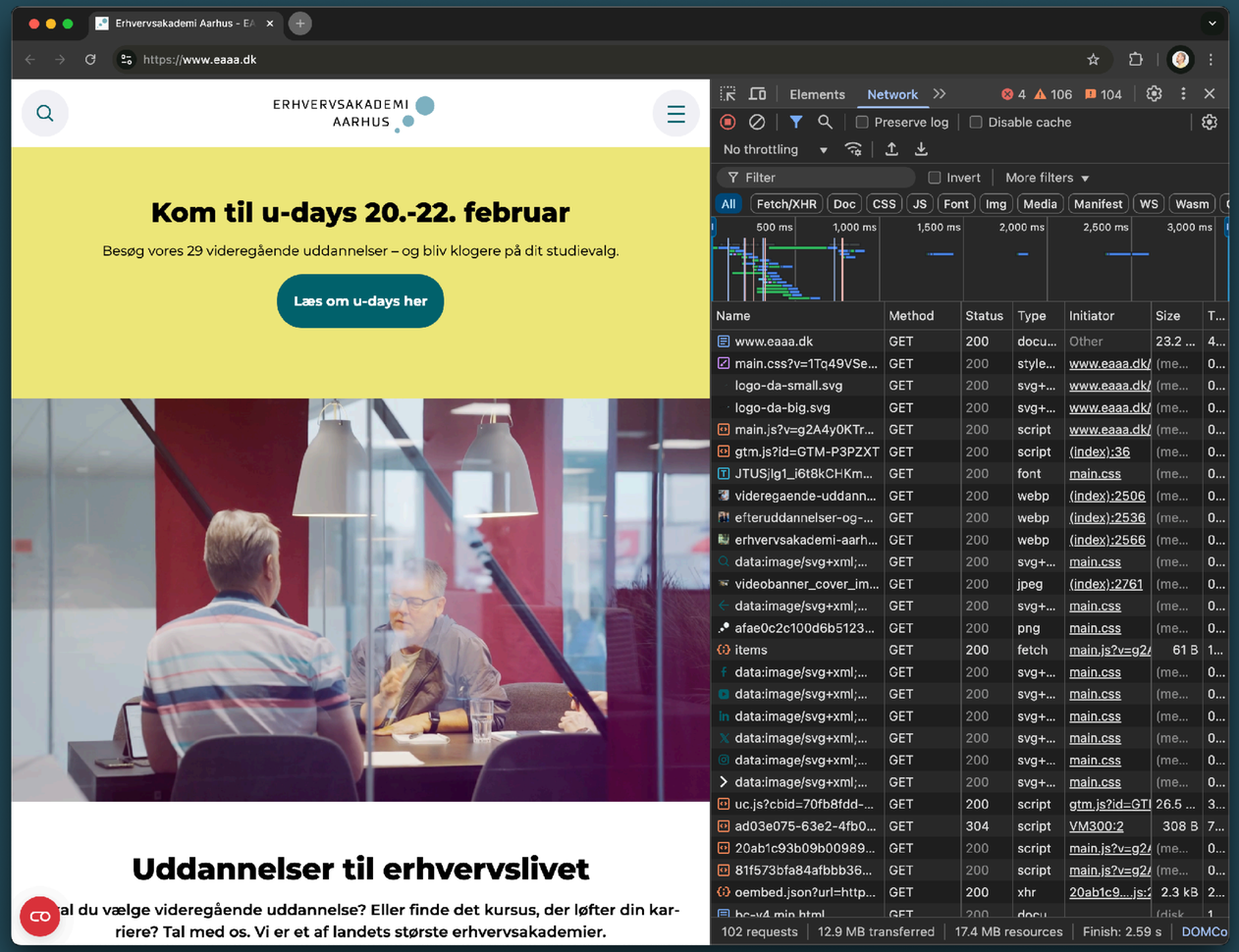
https://www.w3schools.com/whatis/whatis_http.asp

Network Tab


Erhvervsakademi Aarhus - EA
eaaa.dk
ERHVERVSAKADEMI AARHUS
Uddannelser til erhvervslivet
Skal du vælge videregående uddannelse? Eller finde det kursus, der løfter din karriere? Tal med os. Vi er et af landets største erhvervsakademier.
Videregående uddannelser
Professionsbachelor- og
Efteruddannelse og kurser
Få ny viden, du kan bruge direkte i
Elements
Console
Sources
Network
3
51
50
Preserve log
Disable cache
No throttling
Filter
Invert
Hide data URLs
Hide extension URLs
All
Fetch/XHR
Doc
CSS
JS
Font
Img
Media
Manifest
WS
Wasm
Other
Blocked response cookies
Blocked requests
3rd-party requests
1000 ms
2000 ms
3000 ms
4000 ms
5000 ms
6000 ms
Name
Method
Status
Type
Initiator
Size
T..
Waterfall
1.gif?dgi=70fb8fdd-4... GET 200 gif uc.js?cbid... (me... 0...
heatmaps.js GET 200 script monsido-s... (disk... 2...
page-correct.js GET 200 script monsido-s... (disk... 2...
?a=h0r3JOS3pvXaDa... GET 200 gif monsido-s... 57 B 1...
favicon.ico GET 200 x-icon Other (disk... 1...
h0r3JOS3pvXaDabSjt... GET 200 xhr heatmaps.... (disk... 0...
h0r3JOS3pvXaDabSjt... GET 200 xhr page-corr... (disk... 0...
cast_sender.js?loadC... GET 200 script vendor.mo... (disk... 0...
1765804027-75aafef... GET 200 avif Other (disk... 0...
master.json?base64_i... GET 200 xhr vendor.mo... 3.0 kB 1...
1765804027-75aafef... GET 200 avif vendor.mo... (me... 0...
cast_framework.js GET 200 script cast_send... 14 B 1...
cast_sender.js GET 200 script cast_send... (disk... 1...
settings.json GET 200 xhr uc.js?cbid... (disk... 0...
widgetIcon.min.js GET 200 script uc.js?cbid... (disk... 0...
2b51a972.mp4?r=dX... GET 200 xhr vendor.mo... 5.4 kB 2...
d509129e.mp4?r=dX... GET 200 xhr vendor.mo... 264 kB 3...
cf6acf66.mp4?r=dXM... GET 200 xhr vendor.mo... 4.3 ... 2...
2b51a972.mp4?r=dX... GET 200 xhr vendor.mo... 5.4 kB 1...
2b51a972.mp4?r=dX... GET 200 xhr vendor.mo... 5.1 kB 2...
cf6acf66.mp4?r=dXM... GET 200 xhr vendor.mo... 4.6 ... 2...
cf6acf66.mp4?r=dXM... GET 200 xhr vendor.mo... 3.5 ... 3...
collect?v=2&tid=G-D... POST 204 ping js?id=G-D... 17 B 6...
91 requests
12.8 MB transferred
16.2 MB resources
Finish: 5.47 s
DOMContentLoaded: 290 ms

Network Tab


Erhvervsakademi Aarhus - EA
eaaa.dk
ERHVERVSAKADEMI AARHUS
Uddannelser til erhvervslivet
Skal du vælge videregående uddannelse? Eller finde det kursus, der løfter din karriere? Tal med os. Vi er et af landets største erhvervsakademier.
Videregående uddannelser
Efteruddannelse og kurser
Samarbejd og skoler
Elements
Console
Sources
Network
Preserve log
Disable cache
No throttling
Filter
Invert
Hide data URLs
Hide extension URLs
All
Fetch/XHR
Doc
CSS
JS
Font
Img
Media
Manifest
WS
Wasm
Other
Blocked response cookies
Blocked requests
3rd-party requests
Name
Headers
Payload
Preview
Response
Initiator
Timing
items
wa/
247144?d=eaaa.dk
oembed.json?url...
h0r3JOS3pvXaD...
h0r3JOS3pvXaD...
master.json?base...
2b51a972.mp4?r...
settings.json
d509129e.mp4?r...
cf6acf66.mp4?r=...
2b51a972.mp4?r...
2b51a972.mp4?r...
cf6acf66.mp4?r=...
cf6acf66.mp4?r=...
15 / 111 requests
{type: "video", version: "1.0", provider_name: "Vimeo", provider_u
account_type: "starter"
author_name: "Erhvervsakademi Aarhus"
author_url: "https://vimeo.com/user209157220"
description: ""
duration: 17
height: 240
html: "<div style=\"padding:56.25% 0 0 0;position:relative;\"><if
is_plus: "0"
provider_name: "Vimeo"
provider_url: "https://vimeo.com/"
thumbnail_height: 166
thumbnail_url: "https://i.vimeocdn.com/video/1765804027-75aafef51
thumbnail_url_with_play_button: "https://i.vimeocdn.com/filter/ov
thumbnail_width: 295
title: "Silent video D4"
type: "video"
upload_date: "2023-12-08 06:54:18"
uri: "/videos/892590144"
version: "1.0"
video_id: 892590144
width: 426



Network Tab

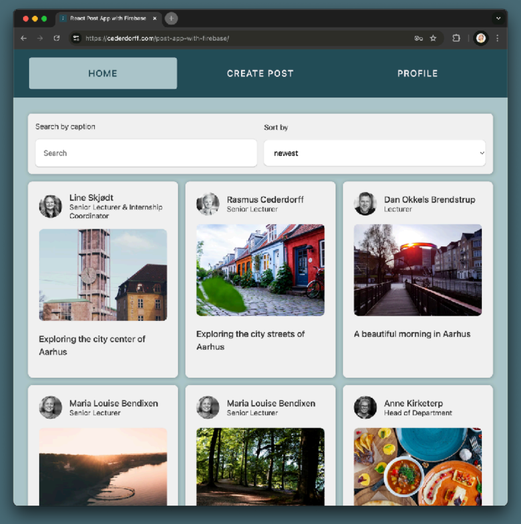
Investigating Network Activity in the Browser

Fetch, HTTP Request & Response

WEB APP

REST API

DATA SOURCE

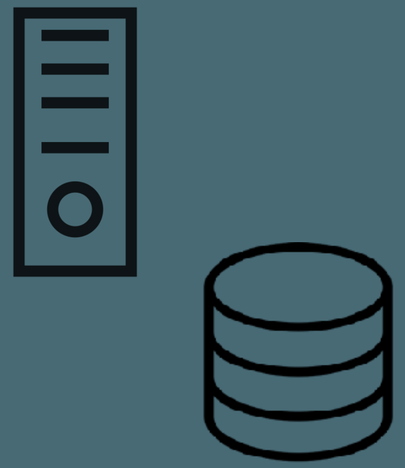


FETCH(GET, POST, PUT, DELETE)

REQUEST

JSON DATA

RESPONSE



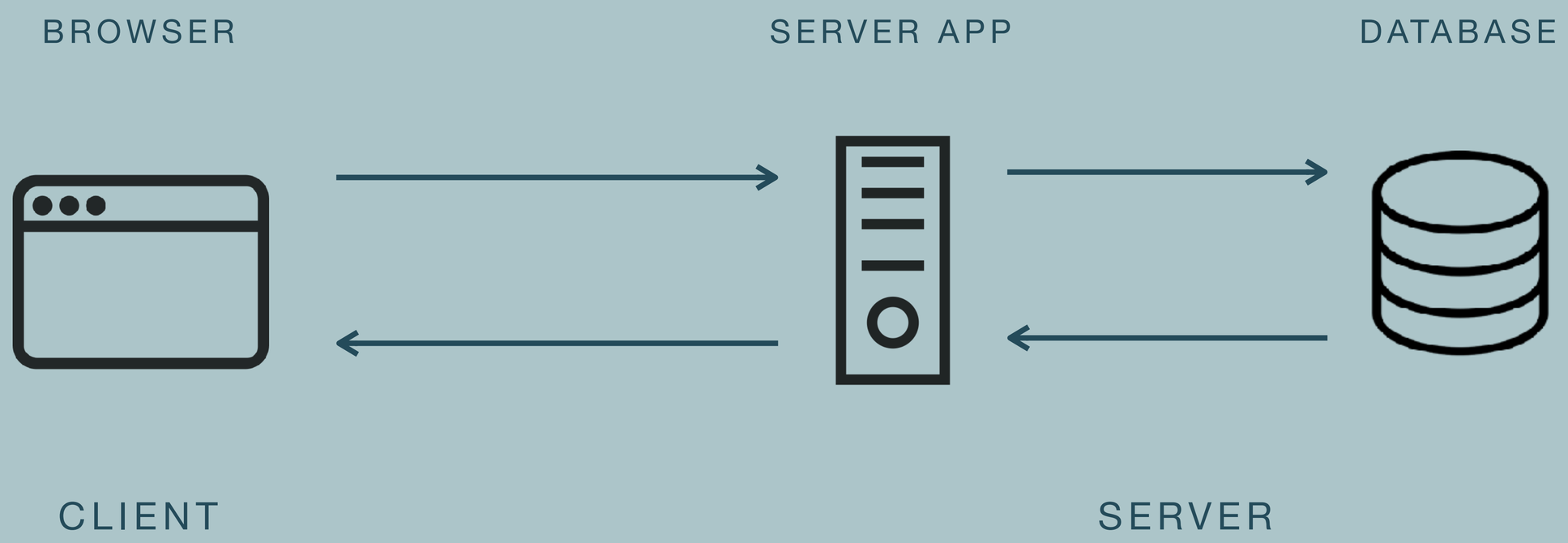
posts

users

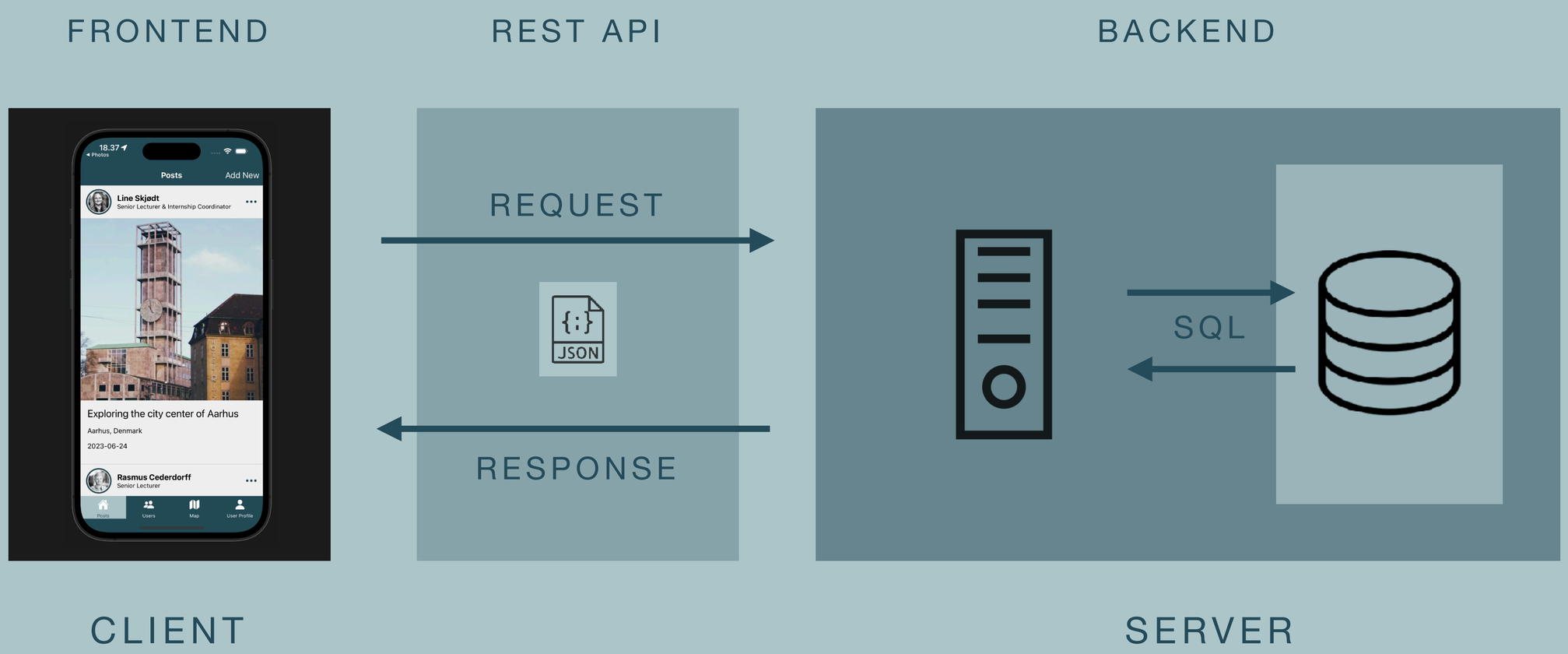
CLIENT

SERVER

Web Dev Architecture



Web Dev Architecture



Introduction to

- Server-side JavaScript runtime environment.
- Executes JavaScript outside of the browser.

- Imagine Node.js as a powerful engine that lets you run JavaScript code outside of a web browser.
- Instead of just using JavaScript for web pages, Node.js lets you use it on your computer or on a server.
- It's like having a super-fast computer that can understand and do all the smart things you do in a web browser with JavaScript, but now it can do it on your own computer or a server.

Run JavaScript, not just in browsers 🎉

app.js — node-js-fun

EXPLORER

NODE-JS-FUN

app.js

app.js > ...

```
1  const yourFavoriteLecturer = {
2      name: "Peter Lind",
3      mail: "petl@kea.dk"
4  };
5
6  console.log(yourFavoriteLecturer);
7
```

PROBLEMS OUTPUT TERMINAL zsh

```
race@RACEs-MacBook-Pro node-js-fun % node app.js
{ name: 'Peter Lind', mail: 'petl@kea.dk' }
race@RACEs-MacBook-Pro node-js-fun %
```

OUTLINE

TIMELINE

0 0 Ln 7, Col 1 Spaces: 4 UTF-8 LF {} JavaScript Go Live Prettier

Key Points

- **JavaScript Beyond Browser**
  Instead of being limited to doing things only on the internet, Node.js lets you use JavaScript to build programs and applications on your computer or server. You get to use the same language you know from web pages.

- **Fast and Efficient**
  Node.js is built to be really fast and efficient. It can handle many things at once without getting slow. This is really useful when you're working with apps that need to do lots of things at the same time, like chat apps or real-time updates.

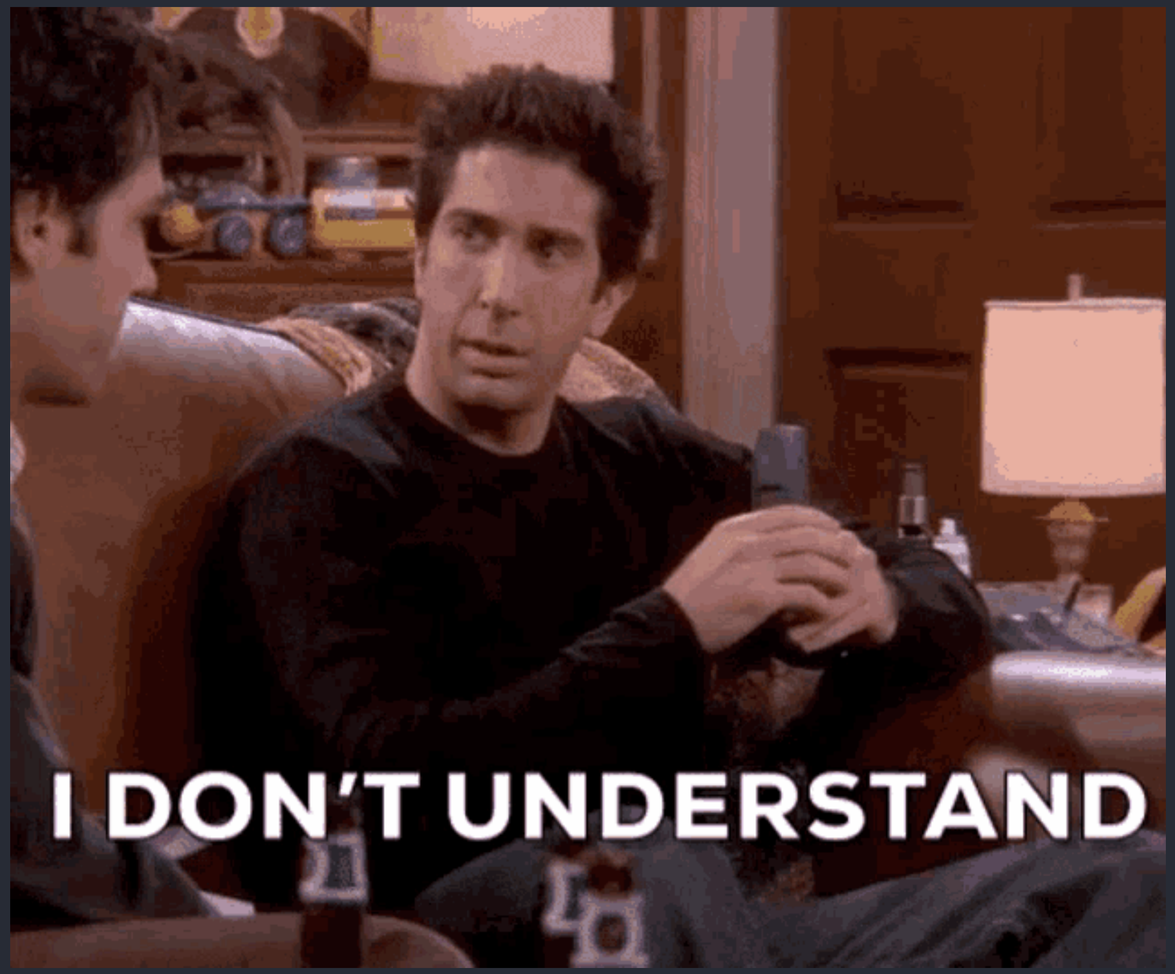
- **Don't Wait for Things**
  Usually, when a computer does something, it has to wait until that task is finished before moving on to the next one. Node.js does things differently. It can start a task and then keep working on other things while it waits for the first task to finish.

- **Customize with Modules**
  Think of modules as little building blocks of code. Node.js gives you lots of ready-made code that you can use in your programs. You don't have to invent everything from scratch. It makes it easy to add new features to your projects.

- **Used for Many Things**
  People use Node.js to build all sorts of things, from simple web servers to complex applications. It's popular in modern web development because it allows fast and efficient communication between clients and servers.

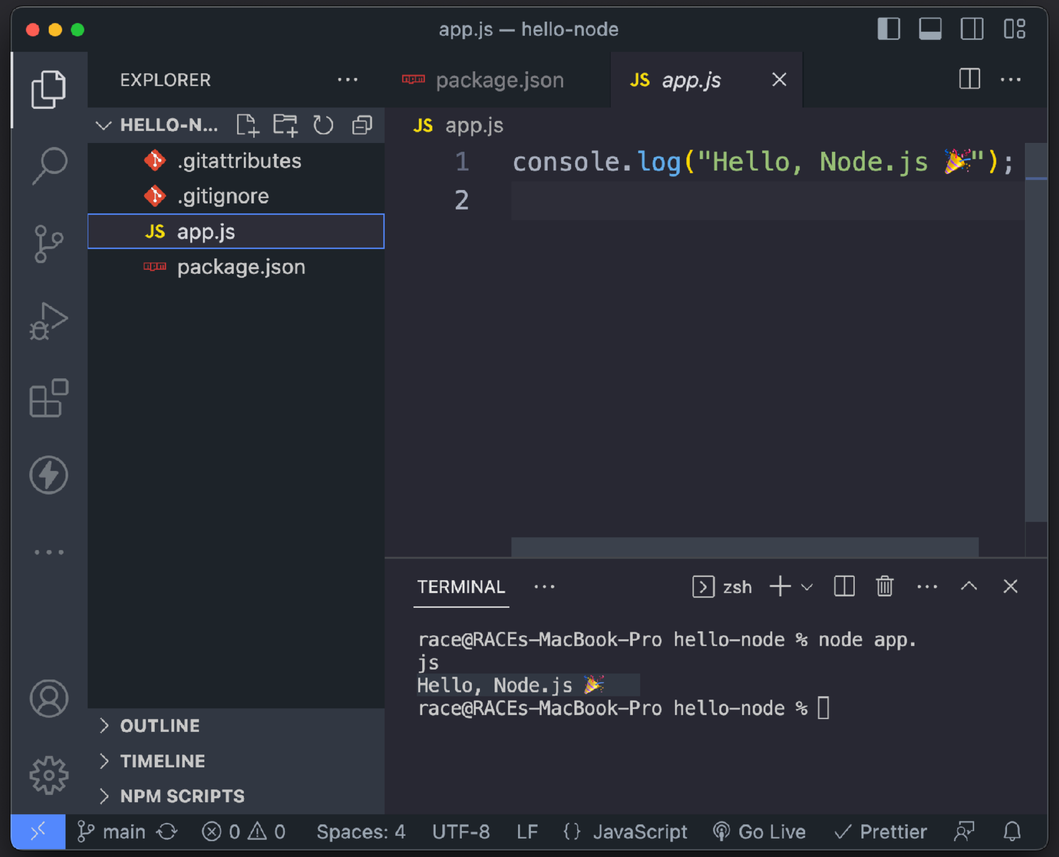
Please,

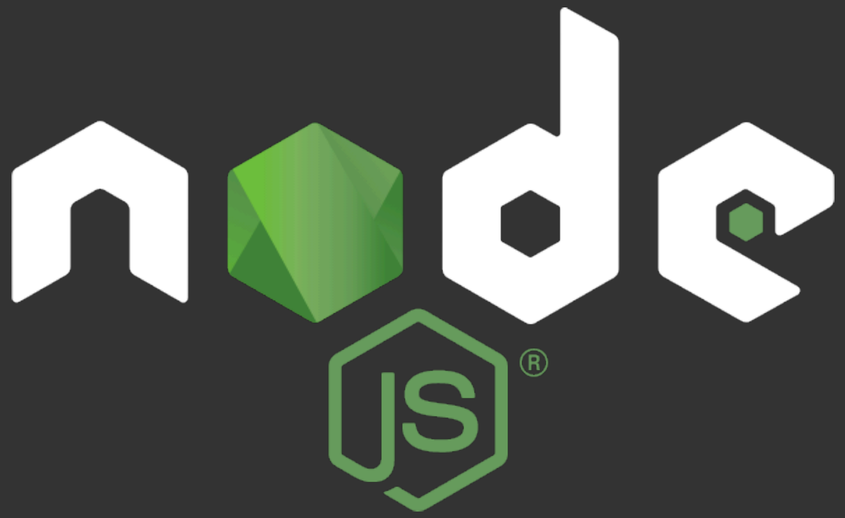
Do not try to understand everything!



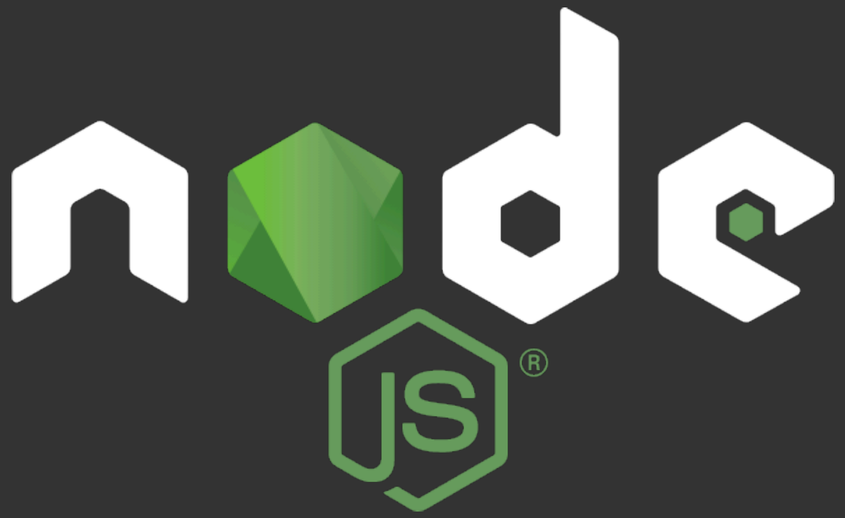
Øvelse

Hello Node.js





Why do we need it?

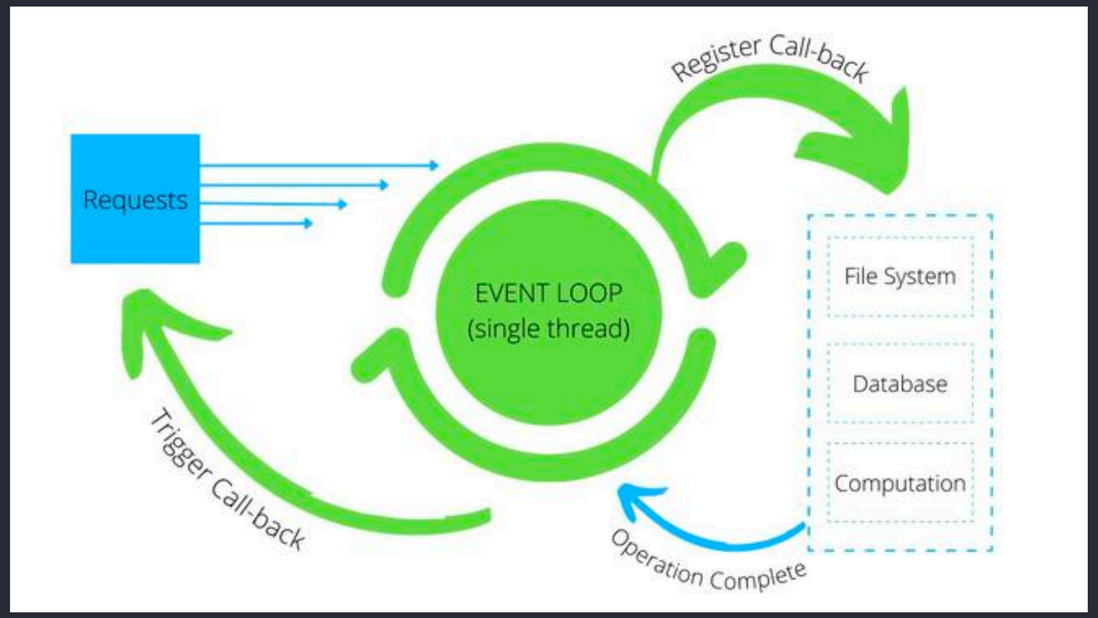


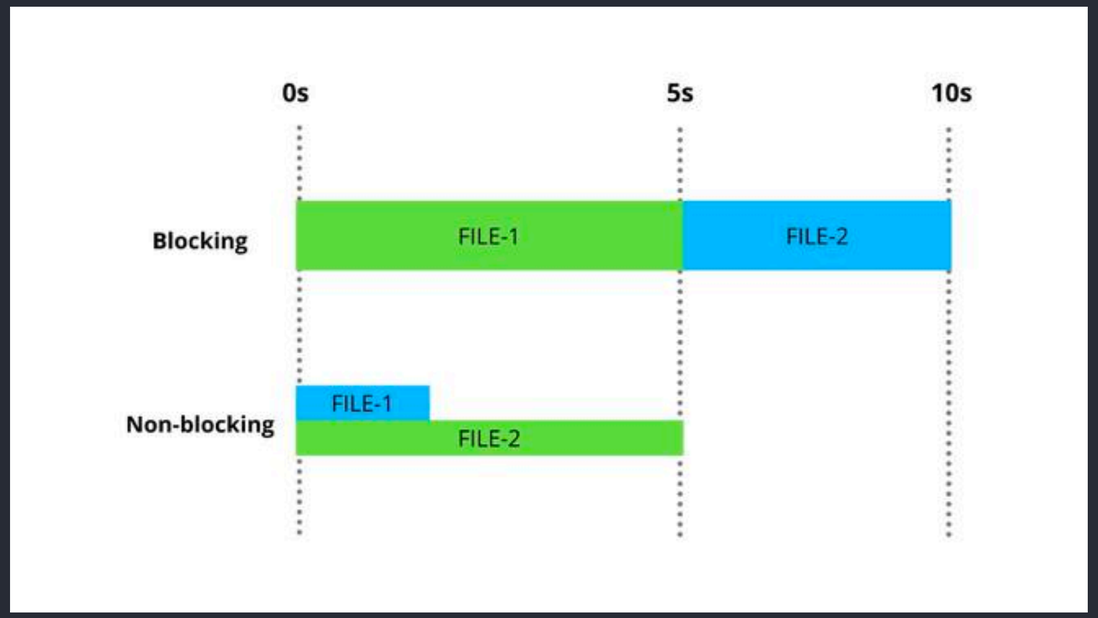
Why do we need it?

- In pairs, discuss and investigate
- Look at code and previous Node projects
- Ask Google and ChatGPT

Runtime
Environment

- The Node.js runtime environment is an open-source, cross-platform JavaScript runtime built on Google Chrome's V8 JavaScript engine.
- It allows developers to execute JavaScript code on the server-side, outside of the browser, enabling them to build scalable and high-performance applications.
- Node.js provides a range of features and capabilities that are particularly well-suited for building real-time, networked, and event-driven applications.



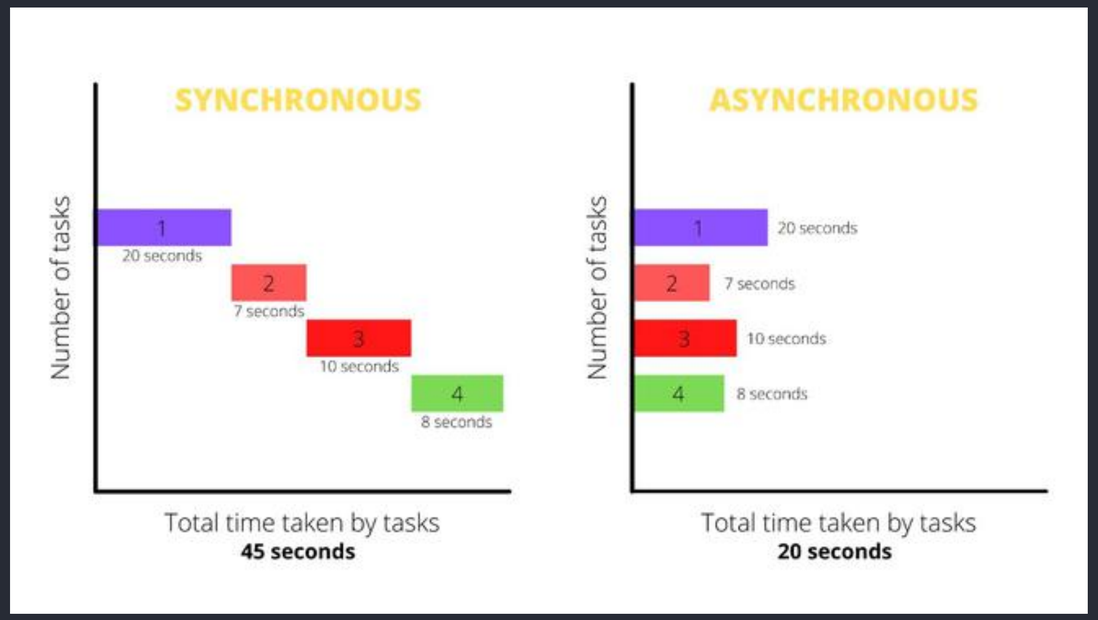


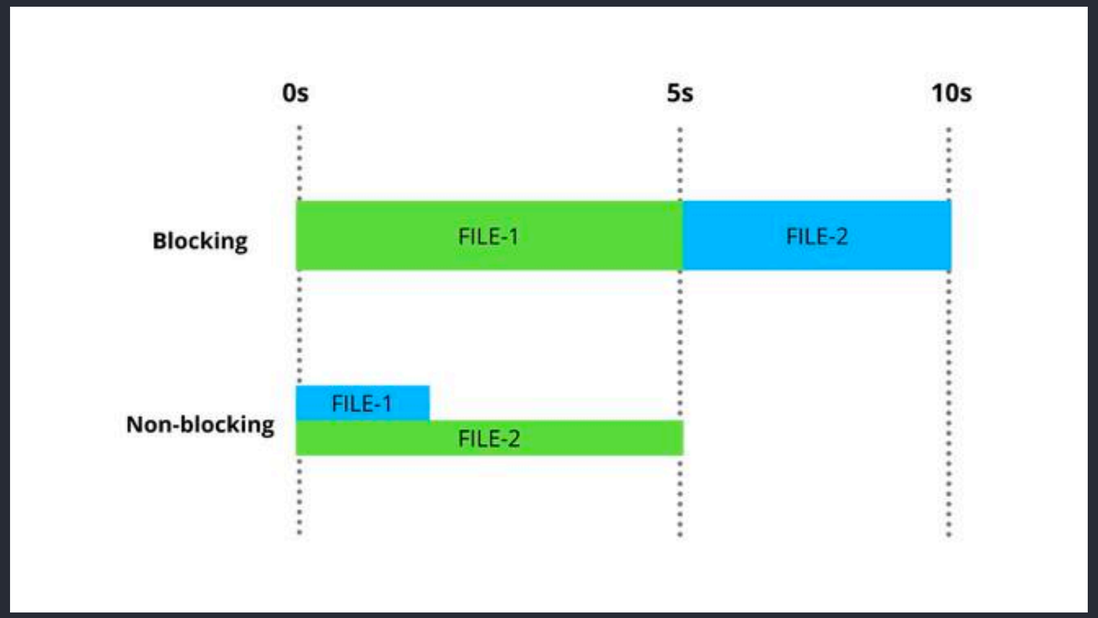
https://www.geeksforgeeks.org/how-node-js-overcome-the-problem-of-blocking-of-i-o-operations/

Runtime Environment

Non-blocking I/O

- Node.js operates using an event-driven, non-blocking I/O model, which means that it can handle multiple concurrent connections without requiring a separate thread for each connection.
- This design choice enables efficient handling of I/O-bound tasks, making it suitable for applications that involve a lot of network communication.





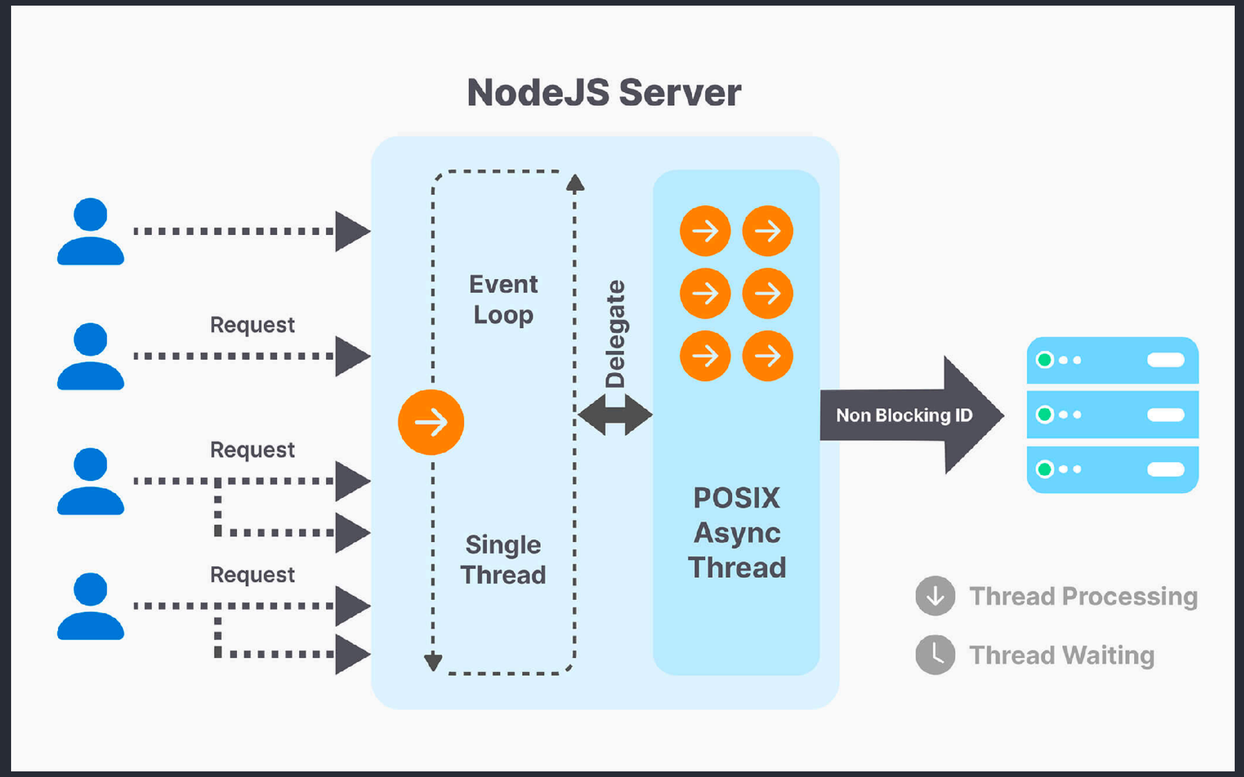
https://www.geeksforgeeks.org/how-node-js-overcome-the-problem-of-blocking-of-i-o-operations/

Runtime Environment

Asynchronous Programming

- Node.js heavily emphasizes asynchronous programming patterns.
- This allows developers to write code that doesn't block the execution of other tasks while waiting for certain operations to complete.
- This approach enhances the application's responsiveness and throughput.

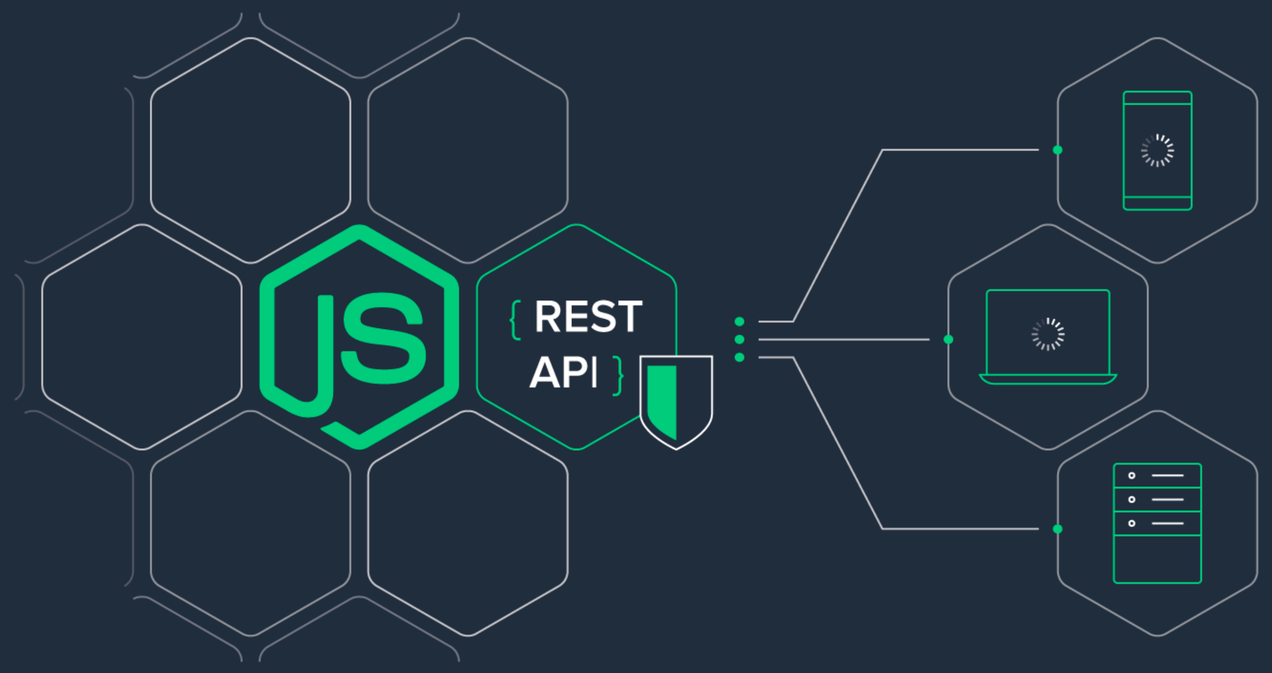
Runtime Environment



https://www.cygnismedia.com/blog/why-nodejs-is-the-top-choice-for-web-development/

Scalability

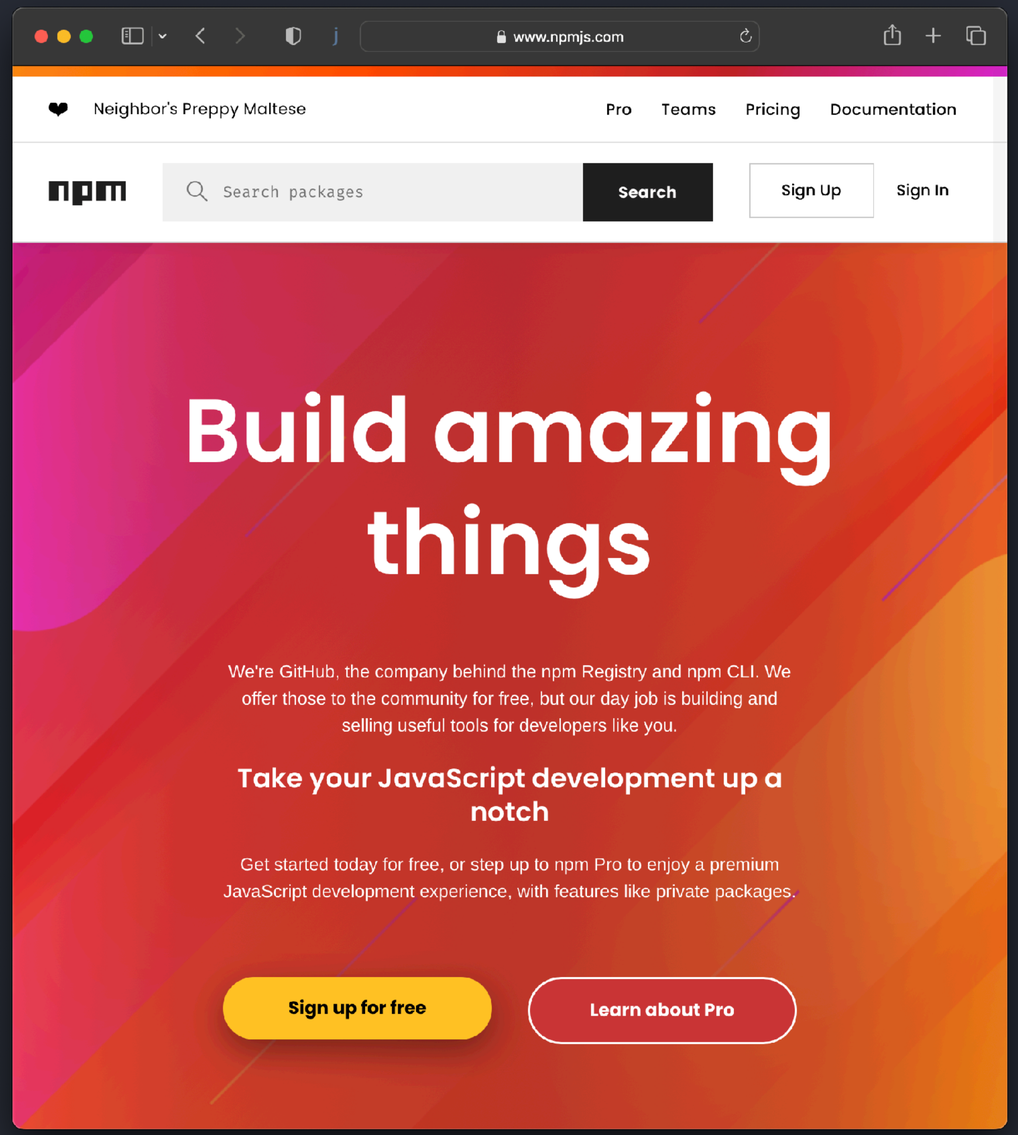
- Due to its non-blocking architecture, Node.js is well-suited for building scalable applications that can handle a large number of concurrent connections.
- This makes it a popular choice for real-time applications like chat applications, online gaming, and streaming services.



Runtime
Environment

Server-side Capabilities

- Node.js can function as a web server, handling incoming HTTP requests and serving responses.
- Developers can build web applications, APIs, and microservices using the same language for both client and server, which can streamline development and maintenance.

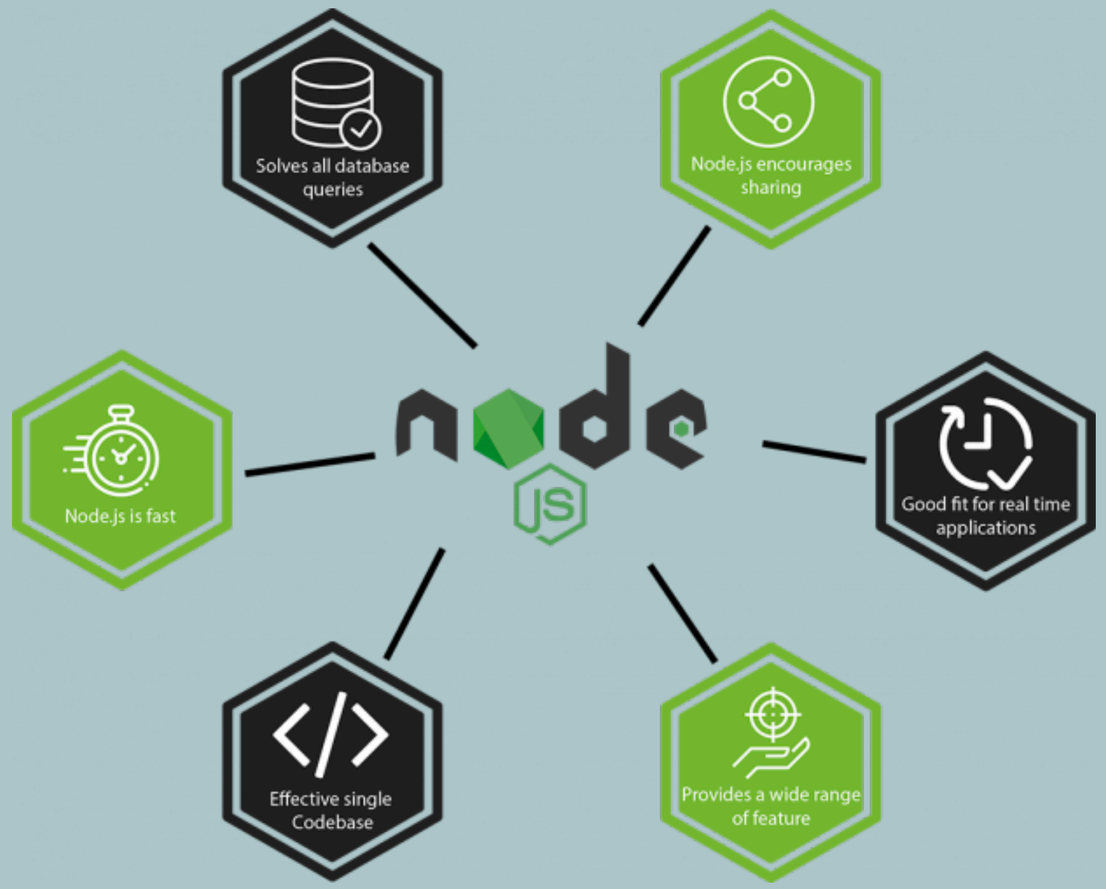


https://www.npmjs.com/

Runtime Environment

NPM (Node Package Manager)

- NPM is a powerful package manager that comes bundled with Node.js.
- It provides access to a vast ecosystem of open-source libraries and modules that can be easily integrated into Node.js applications, saving developers time and effort by avoiding the need to reinvent the wheel.



Runtime Environment

Community and Ecosystem

- Node.js has a large and active community that continuously contributes to its growth.
- The ecosystem includes various tools, frameworks, and libraries that extend its capabilities and make development more efficient.

Runtime
Environment

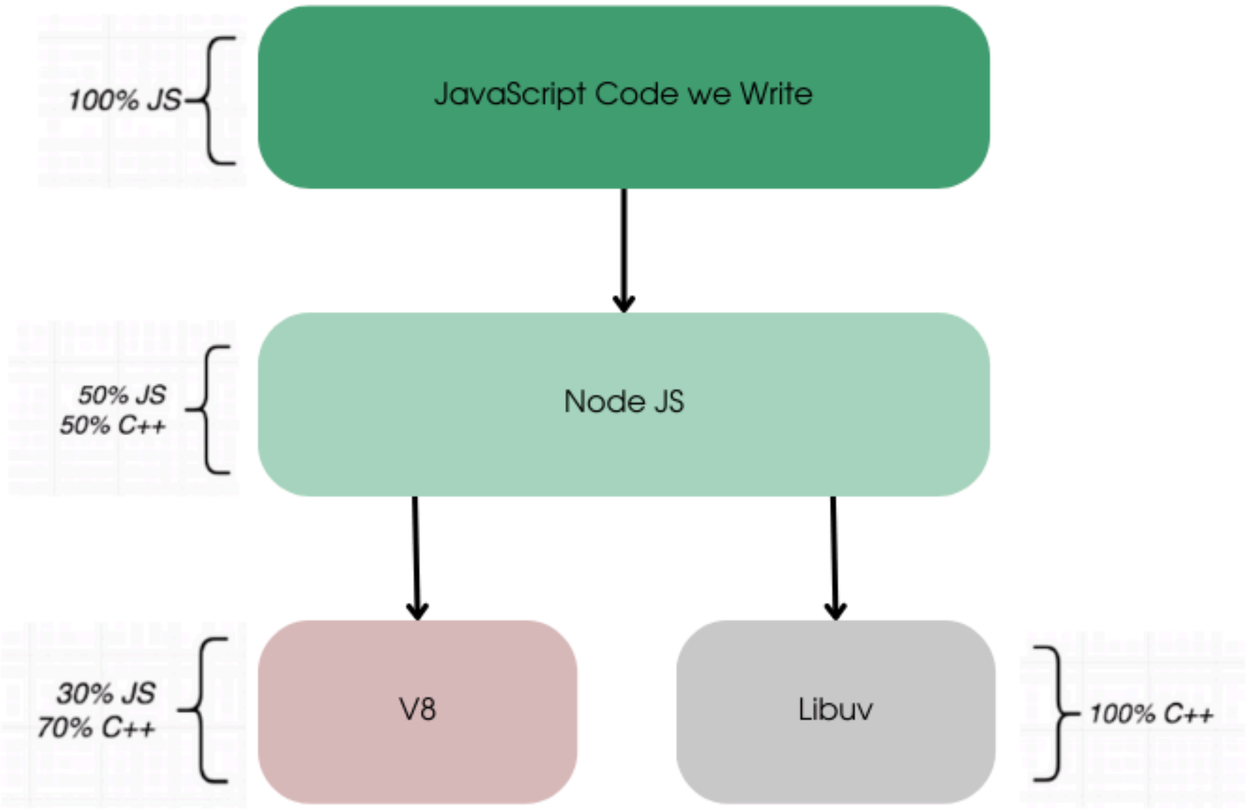
In summary, Node.js provides a runtime environment that allows developers to use JavaScript beyond the browser, enabling them to create high-performance, scalable, and real-time applications for various use cases.

... we can leverage our existing **JavaScript** **expertise** to develop backend applications, server apps, REST APIs, etc.

Runtime
Environment

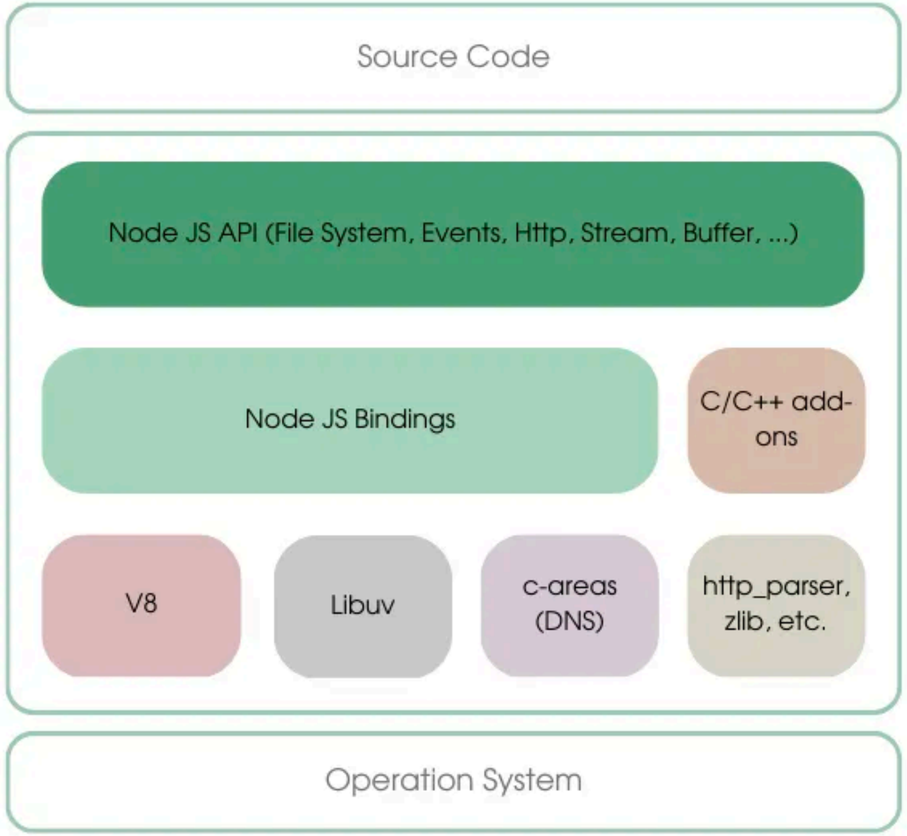
node JS = Runtime Environment + JavaScript Library

Node.js Architecture



https://levelup.gitconnected.com/nodejs-runtime-environment-libuv-library-event-loop-thread-pool-5f5ecadc0318

Node.js Architecture



https://levelup.gitconnected.com/nodejs-runtime-environment-libuv-library-event-loop-thread-pool-5f5ecadc0318

Node.js also comes with a toolbox called NPM, which has all sorts of ready-made tools (like Lego blocks) that developers can use to build their programs. Instead of building everything from scratch, you can use these tools to save time and make your app even better.

`npm` is a package manager for Node.js packages (JS modules)

- Node.js packages contains all the files you need for a module.
- Modules are JavaScript libraries you can include in your project.

My Project

NODE-USERS-REST-API

- node_modules
- .gitignore
- app.js
- data.json
- package-lock.json
- package.json

npm

Express JS

mysql2

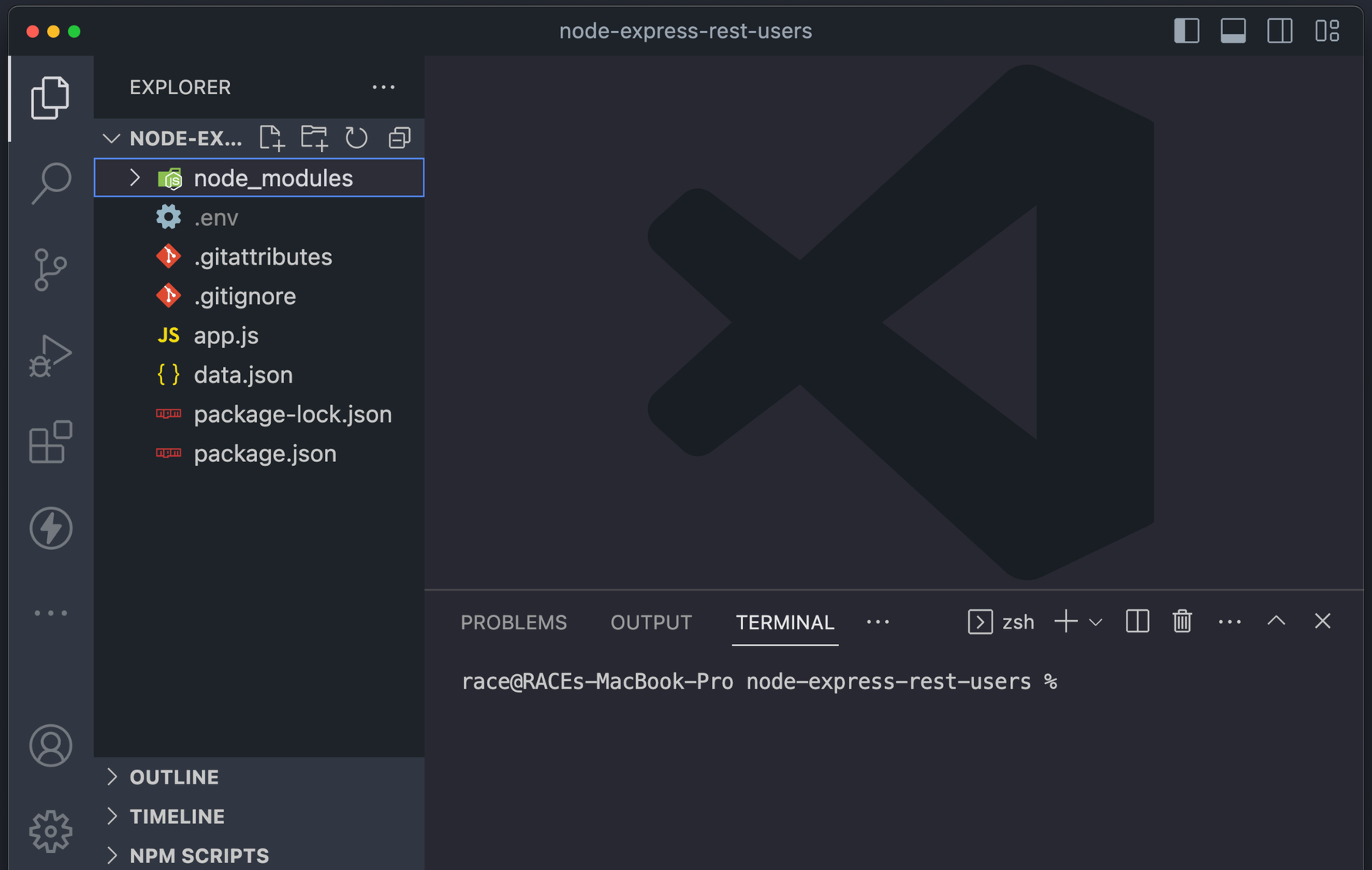
CORS

Remix

Mongoose

And many other
JavaScript libraries

node_modules



npm install {package-name}

package.json — node-express-rest-users

EXPLORER

NODE-EX...

- node_modules
- .env
- .gitattributes
- .gitignore
- app.js
- data.json
- package-lock.json
- package.json

OUTLINE

TIMELINE

NPM SCRIPTS

package.json > {} dependencies

```
10      "author": "Rasmus Cederdorff",
11      "license": "ISC",
12      "dependencies": {
13          "cors": "^2.8.5",
14          "express": "^4.18.2"
15      }
16  }
17
```

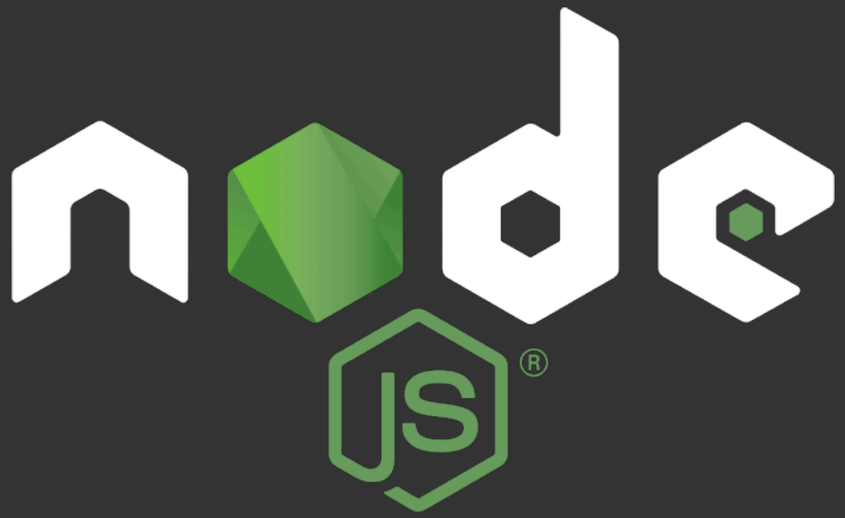
PROBLEMS OUTPUT TERMINAL ··· zsh

```
race@RACEs-MacBook-Pro node-express-rest-users % npm install express
```

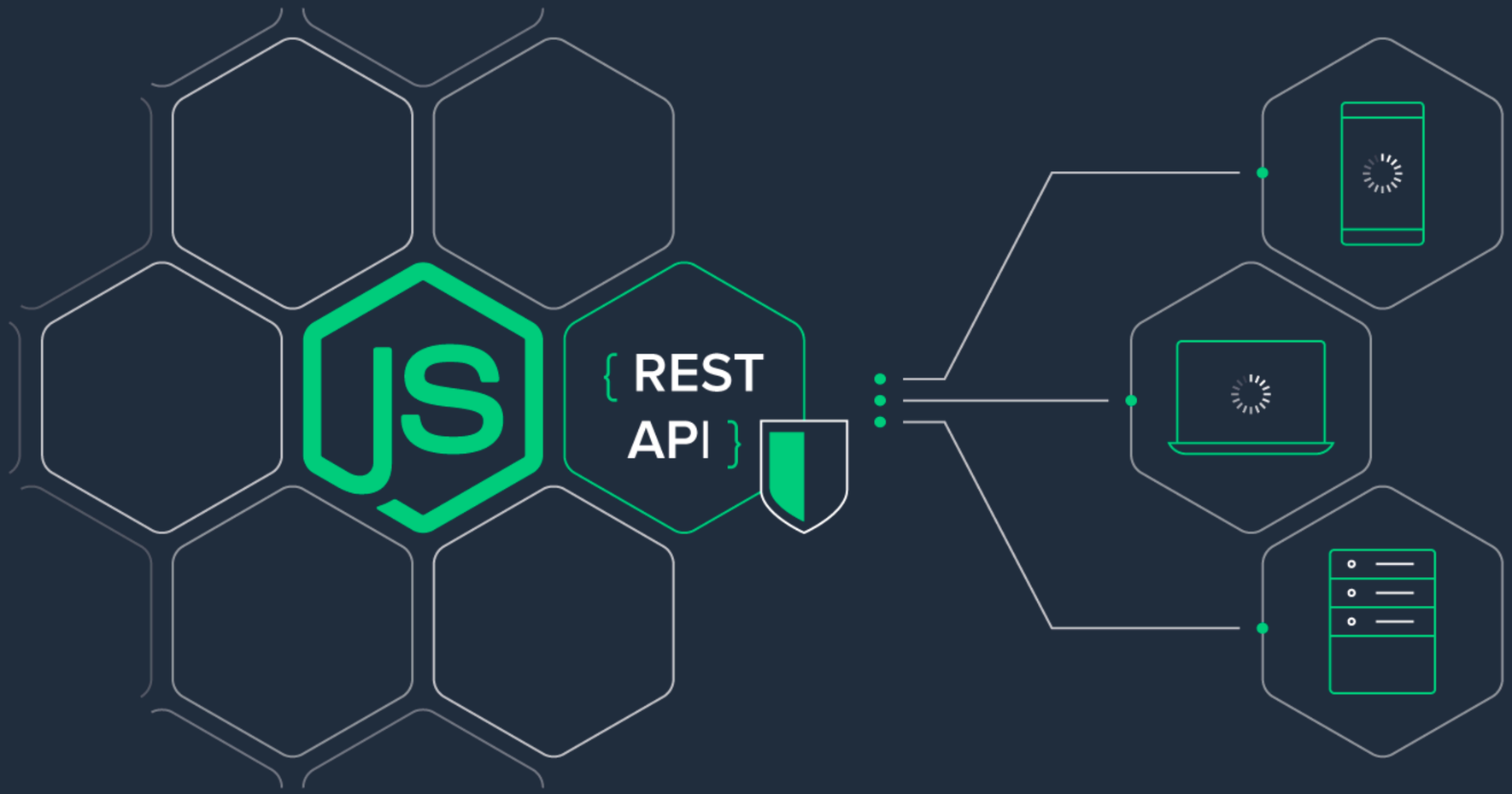
npm consists of

1. Website to discover packages
2. Command line interface (CLI)
3. A registry - database with JS software/ libraries (node modules)

https://docs.npmjs.com/about-npm/



Node.js HTTP Module


JS
{ REST
API }

Node.js HTTP Module

w3schools.com/nodejs/nodejs_http.asp

Apps KEA DAT DAT.js IT Arkitektur EAAA BOOKS Code & Templates

Menu Sign Up Log in

CSS JAVASCRIPT SQL PYTHON JA

The Built-in HTTP Module

Node.js has a built-in module called HTTP, which allows Node.js to transfer data over the Hyper Text Transfer Protocol (HTTP).

To include the HTTP module, use the `require()` method:

```
var http = require('http');
```

Node.js as a Web Server

The HTTP module can create an HTTP server that listens to server ports and gives a response back to the client.

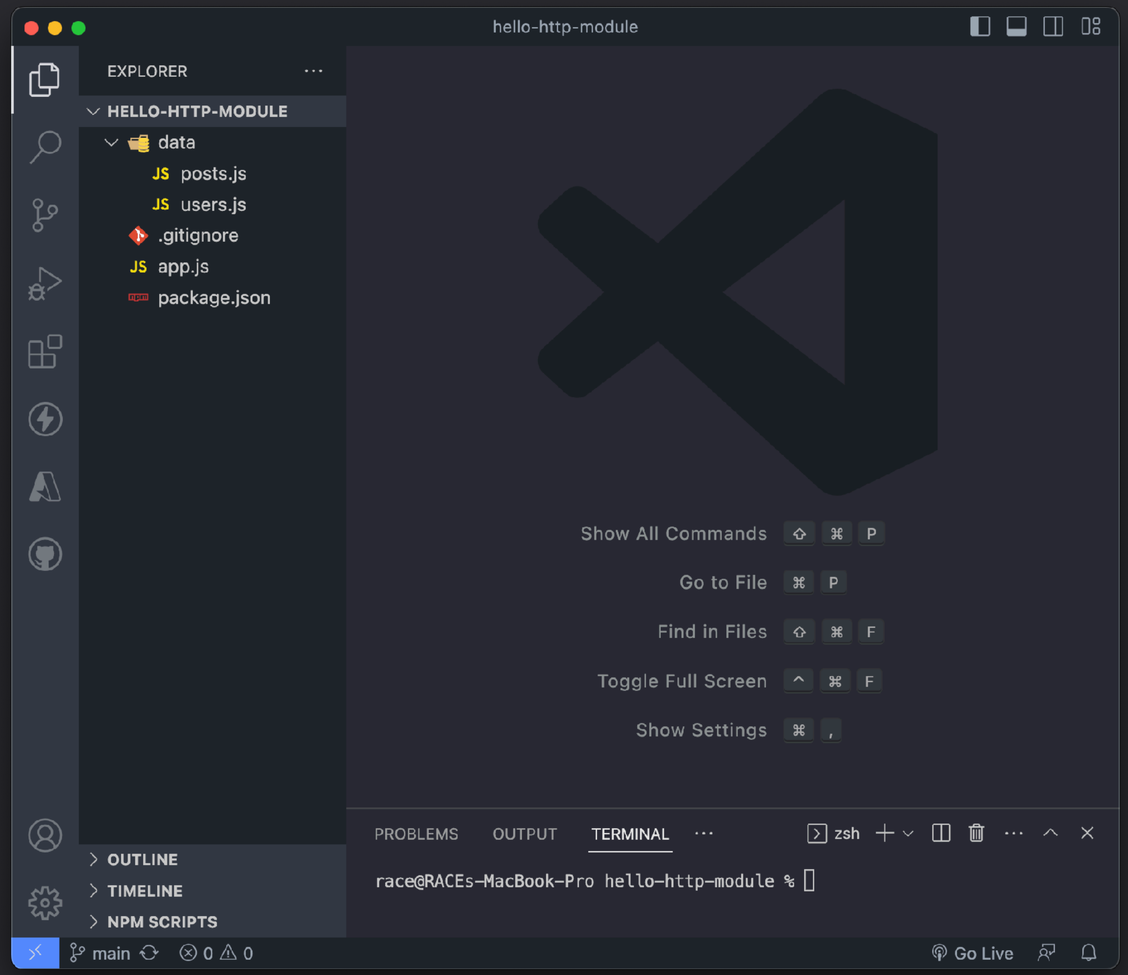
Use the `createServer()` method to create an HTTP server:

https://www.w3schools.com/nodejs/nodejs_http.asp

```
// Import the built-in Node.js 'http' module.
import http from "http";

// Here, we create an HTTP server using 'createServer' function.
// Takes a callback function that is called when a page is requested.
const app = http.createServer((request, response) => {
    // Set status code and headers for the response.
    response.statusCode = 200;
    response.setHeader("Content-Type", "text/plain");
    // Send a message as the response.
    response.end("Working with HTTP Module and routing");
});

// Here, we define the port number the server should listen on.
const port = 3000;

// Finally, we start the server by calling the 'listen' function.
app.listen(port, () => {
    // Once the server is started, we display a message in the terminal.
    console.log(`Server is running at http://localhost:${port}`);
});
```

Øvelse

Hello HTTP Module



POSTMAN

Getting Started with Postman

- Postman is a software tool for working with APIs.
- We can use it as client to test our backend API’s and to test APIs in general.
- It helps developers send requests and receive responses from APIs.
- Postman tests API functionality and documents how to use APIs.
- It simplifies API development and testing.
- Developers collaborate using Postman to work together effectively.

Web Development

POSTMAN

FIREBASE REST API

DATA SOURCE

FETCH(GET, POST, PUT, DELETE)

REQUEST

JSON DATA

RESPONSE

users

CLIENT

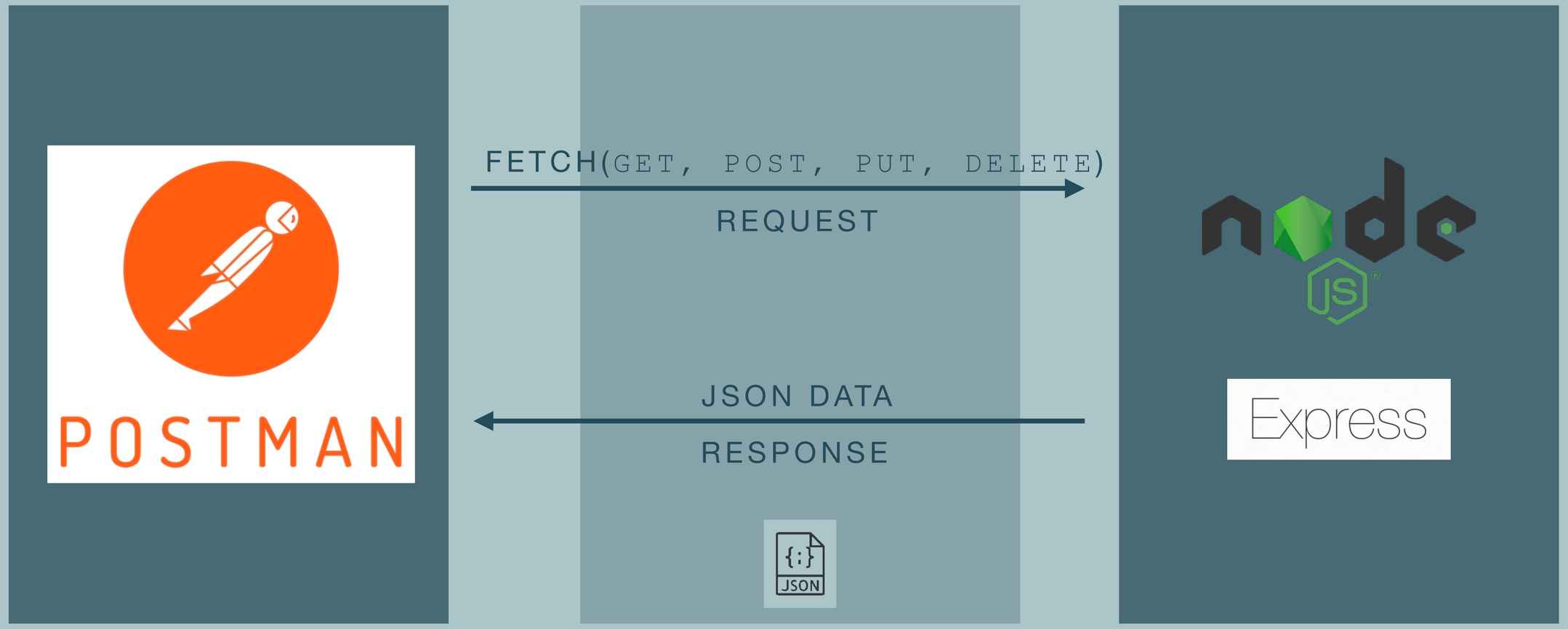
SERVER

Web Development

POSTMAN

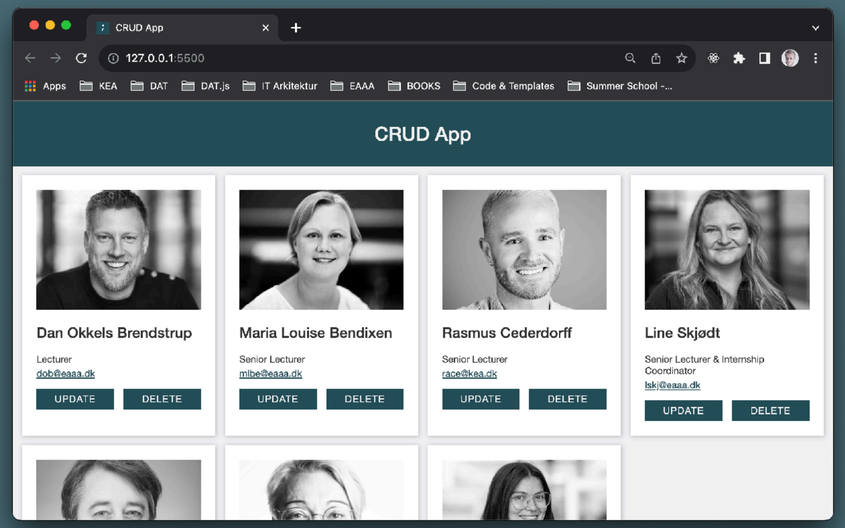
FIREBASE REST API

DATA SOURCE

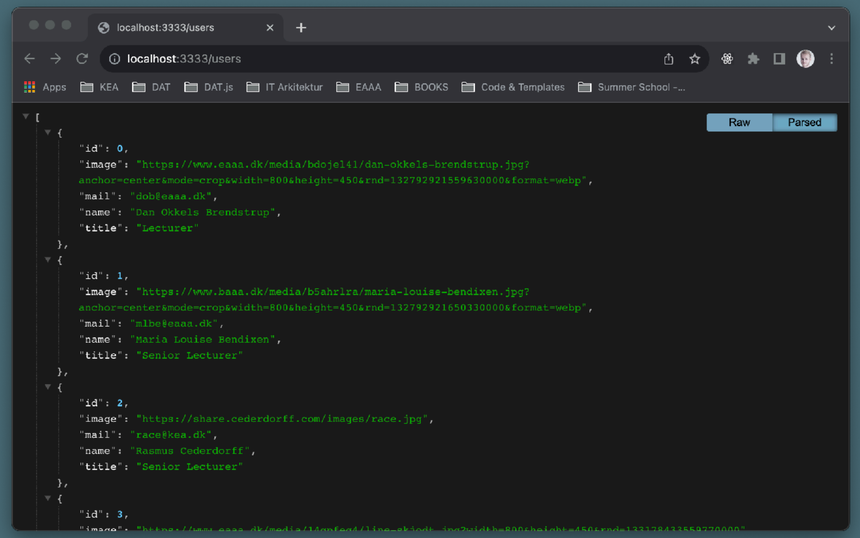


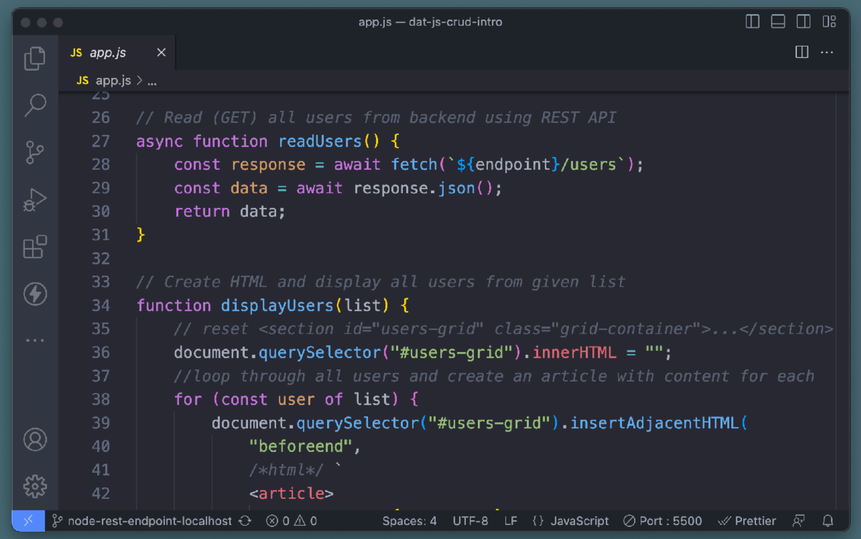
CLIENT

SERVER

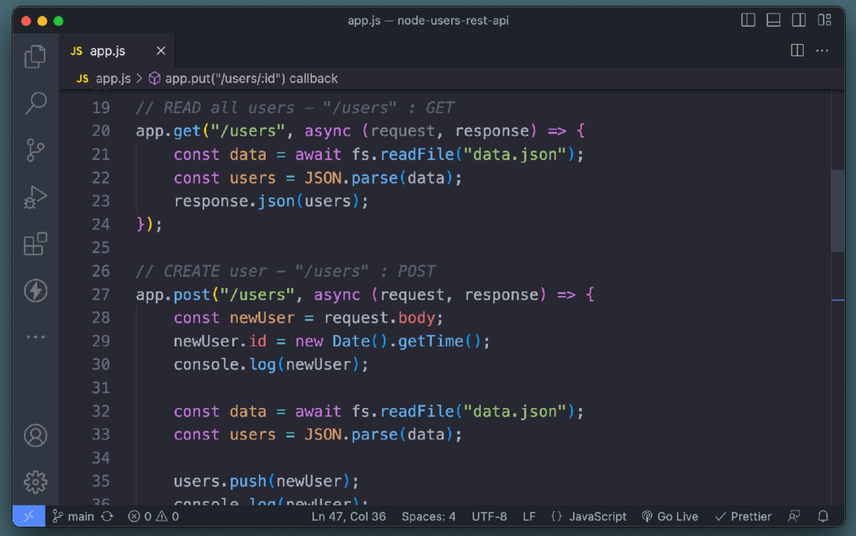


REQUEST



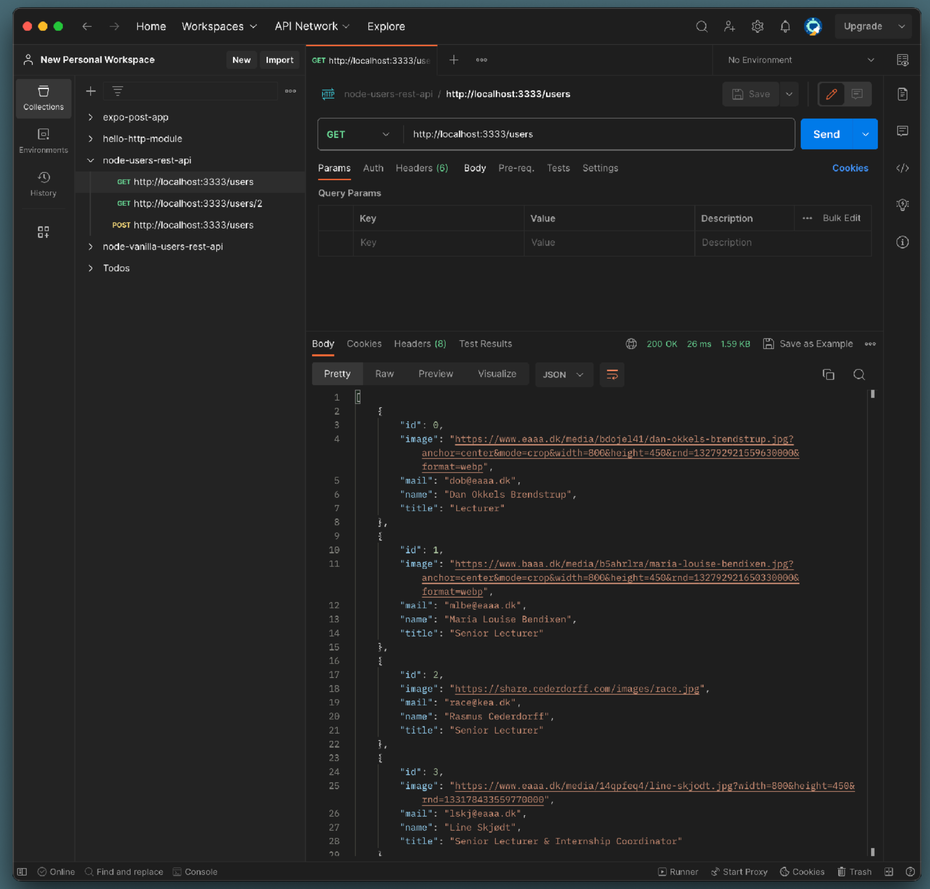


RESPONSE



Frontend

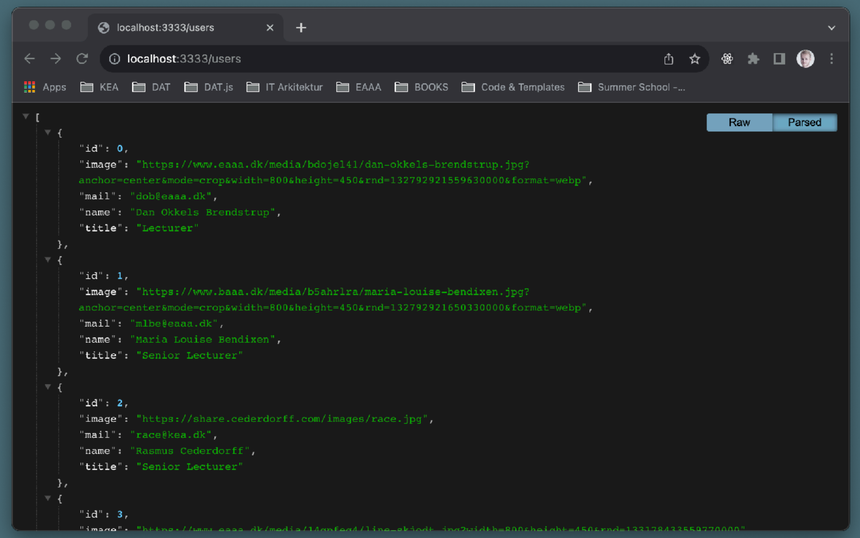
Backend

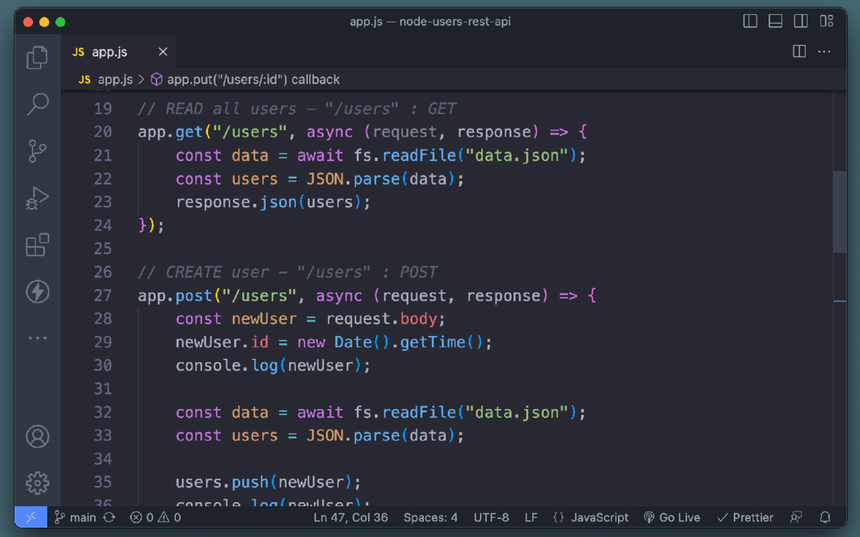


Client

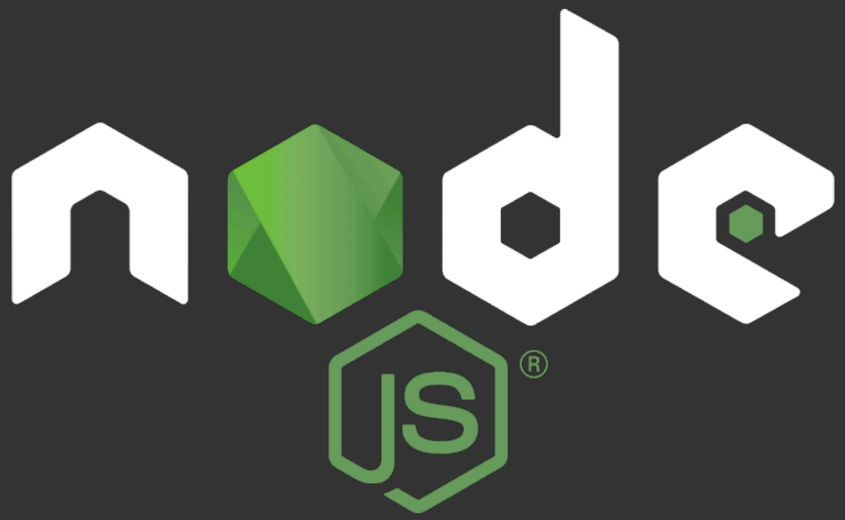
REQUEST

RESPONSE



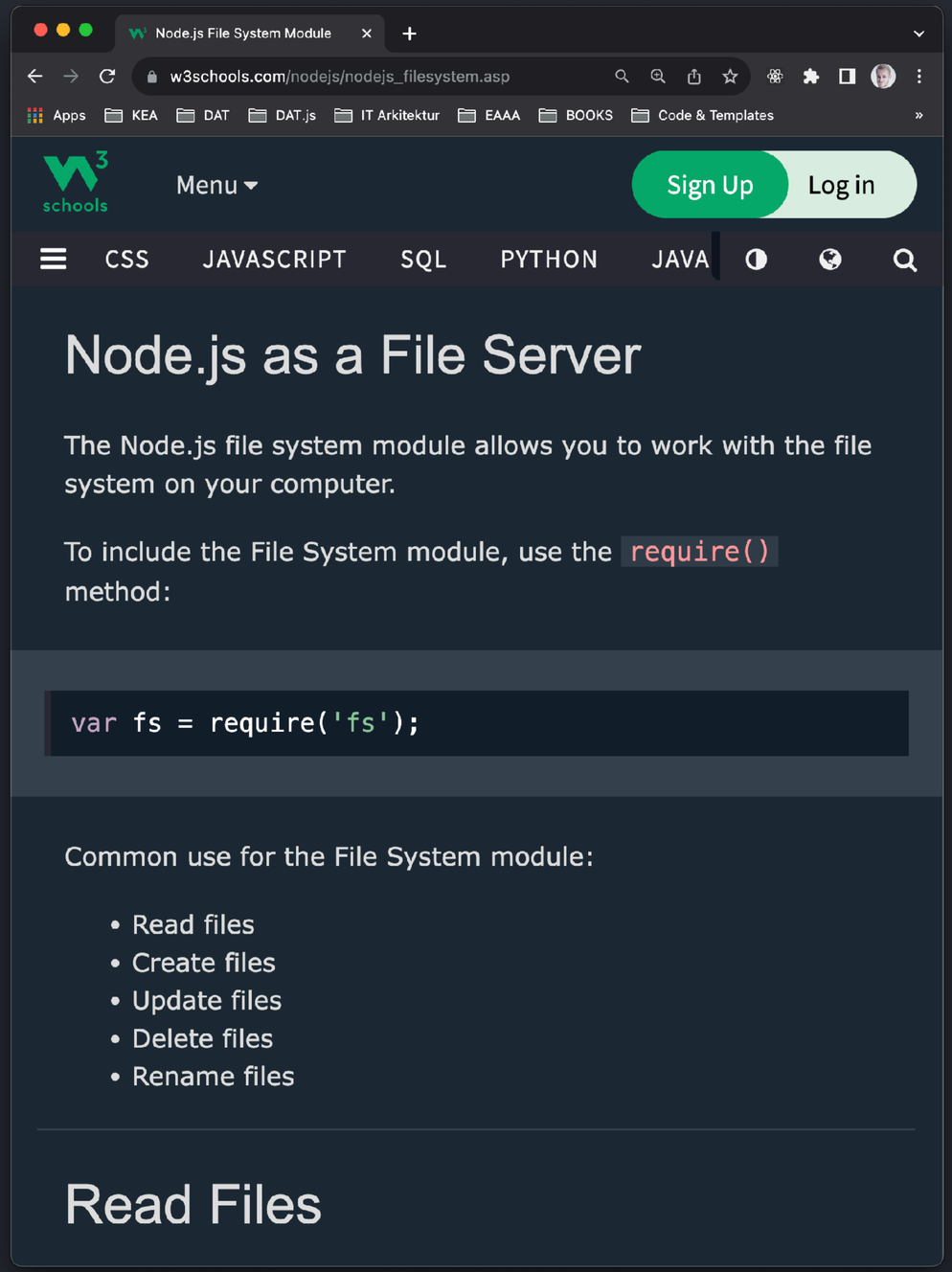


Backend



Node.js File System

Allows us to work with the file system on the computer or server

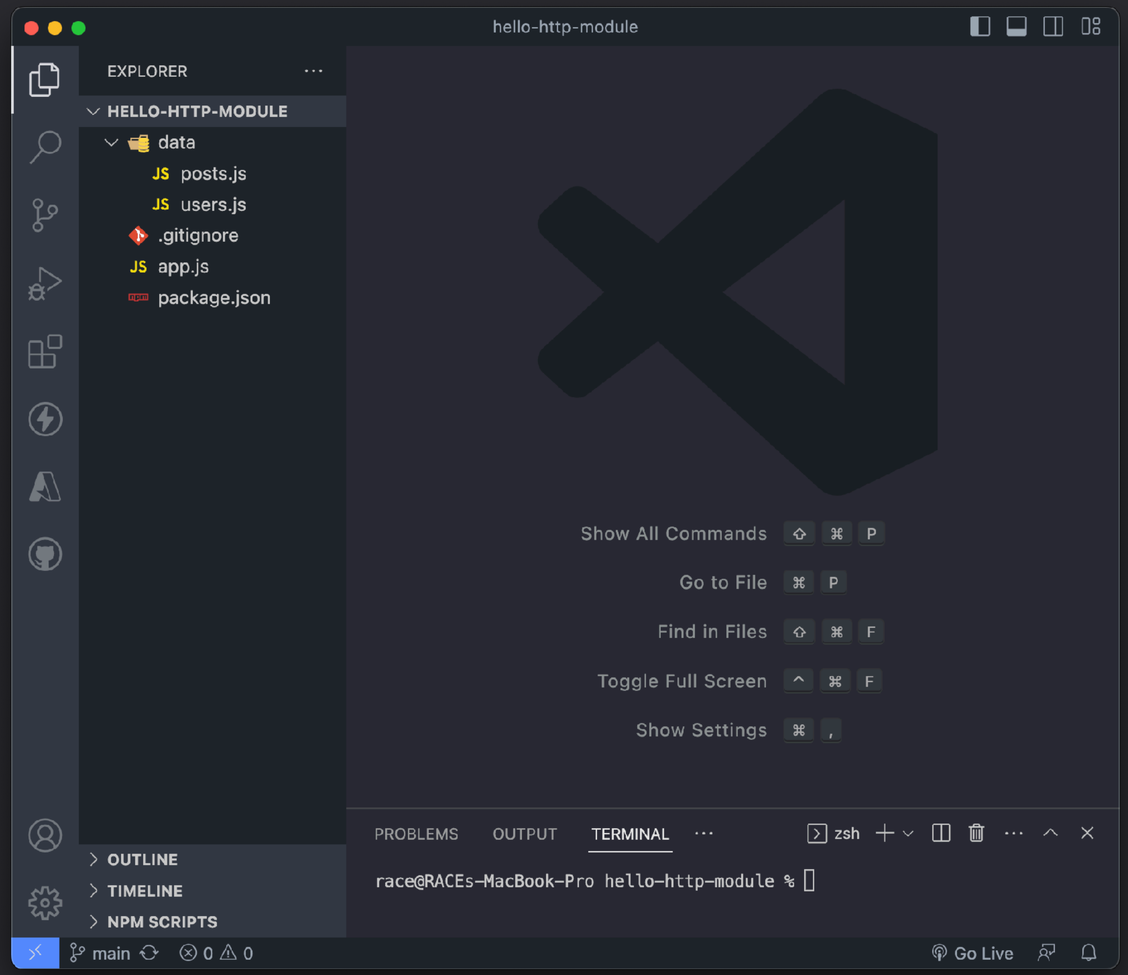


https://www.w3schools.com/nodejs/nodejs_filesystem.asp

```
import fs from "fs/promises";

// Læs fra JSON
const json = await fs.readFile("data/users.json");
console.log(json);
// Parse til JavaScript
const users = JSON.parse(json);
console.log(users);
// Tilføj "user" til "users"
users.push({ name: "Peter Lind", title: "Senior Lecturer" });
// Konverter users til JSON igen
const usersJSON = JSON.stringify(users);
// Skriv til JSON-fil
await fs.writeFile("data/users.json", usersJSON);
```

Øvelse

Node.js File System



Wrap Up

- NPM & JavaScript Libraries
  - Node.js NPM
  - Node.js Upload Files
  - Node.js Send an Email
  - Brug ChatGPT og Google

With your own words

- What’s Node.js?
- Important features of Node.js
- What did we build?
- What improvements can be made?
- How can we simplify the implementation of the different concepts?

Introduction to

Express JS

Express JS

- A popular, minimalist web application framework for building APIs and web applications using Node.js.
- It simplifies the process of building web applications and APIs by providing a set of tools and utilities.

Express JS

A minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications

Express JS

Express.js is known for its flexibility and ease of use, making it an ideal choice for developers who want to build web applications quickly and efficiently. With its built-in middleware functions, routing system, and template engines, Express.js simplifies the process of building complex web applications.

My Project

NODE-USERS-REST-API

- node_modules
- .gitignore
- app.js
- data.json
- package-lock.json
- package.json

npm

Express JS

mysql2

CORS

Remix

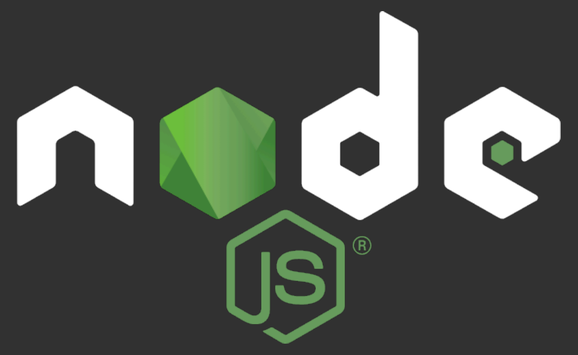
Mongoose

And many other
JavaScript libraries

Why Express.js?

HTTP Module

—>



+

express

Simple GET Request

```
const app = http.createServer(async (request, response) => {
    // READ all users - "/users" : GET
    if (request.url === "/users" && request.method === "GET") {
        const data = await fs.readFile("data.json");
        response.writeHead(200, { "Content-Type": "application/json" });
        response.end(data);
    }
```

—>

```
// READ all users - "/users" : GET
app.get("/users", async (request, response) => {
    const data = await fs.readFile("data.json");
    const users = JSON.parse(data);
    response.json(users);
});
```

“Simple” POST Request

```javascript
const app = http.createServer(async (request, response) => {
    // "/users" : POST
    if (request.url === "/users" && request.method === "POST") {
        // get the data sent along
        const newUser = JSON.parse(await getReqData(request));
        // user object with generated dummy id and parsed userData
        const user = {
            id: new Date().getTime(),
            ...newUser
        };
        const data = await fs.readFile("data.json");
        const users = JSON.parse(data);
        users.push(user);
        const usersJSON = JSON.stringify(users);
        await fs.writeFile("data.json", usersJSON);

        // set the status code and content-type
        response.writeHead(200, { "Content-Type": "application/json" });
        //send the user
        response.end(usersJSON);
    }
});

async function getReqData(req) {
    return new Promise((resolve, reject) => {
        let body = "";
        req.on("data", chunk => (body += chunk));
        req.on("end", () => resolve(body));
        req.on("error", reject);
    });
}
```

—>

```javascript
// CREATE user - "/users" : POST
app.post("/users", async (request, response) => {
    const newUser = request.body;
    newUser.id = new Date().getTime();

    const data = await fs.readFile("data.json");
    const users = JSON.parse(data);

    users.push(newUser);
    fs.writeFile("data.json", JSON.stringify(users));
    response.json(users);
});
```

- **Single-page applications (SPAs)**
  Express.js creates SPAs that load all resources on a single page, ensuring a seamless user experience.
- **Mobile applications**
  Express.js builds mobile apps using web technologies like HTML, CSS, and JavaScript.
- **RESTful APIs**
  Express.js crafts RESTful APIs for data manipulation and access by other applications.
- **Server-side rendering**
  Express.js enables server-side rendering, rendering pages on the server before sending to the client.
- **Real-time applications**
  Express.js constructs real-time apps, facilitating ongoing client-server communication.
- **Microservices**
  Express.js constructs microservices, small independent units combined to create larger applications.

Express JS

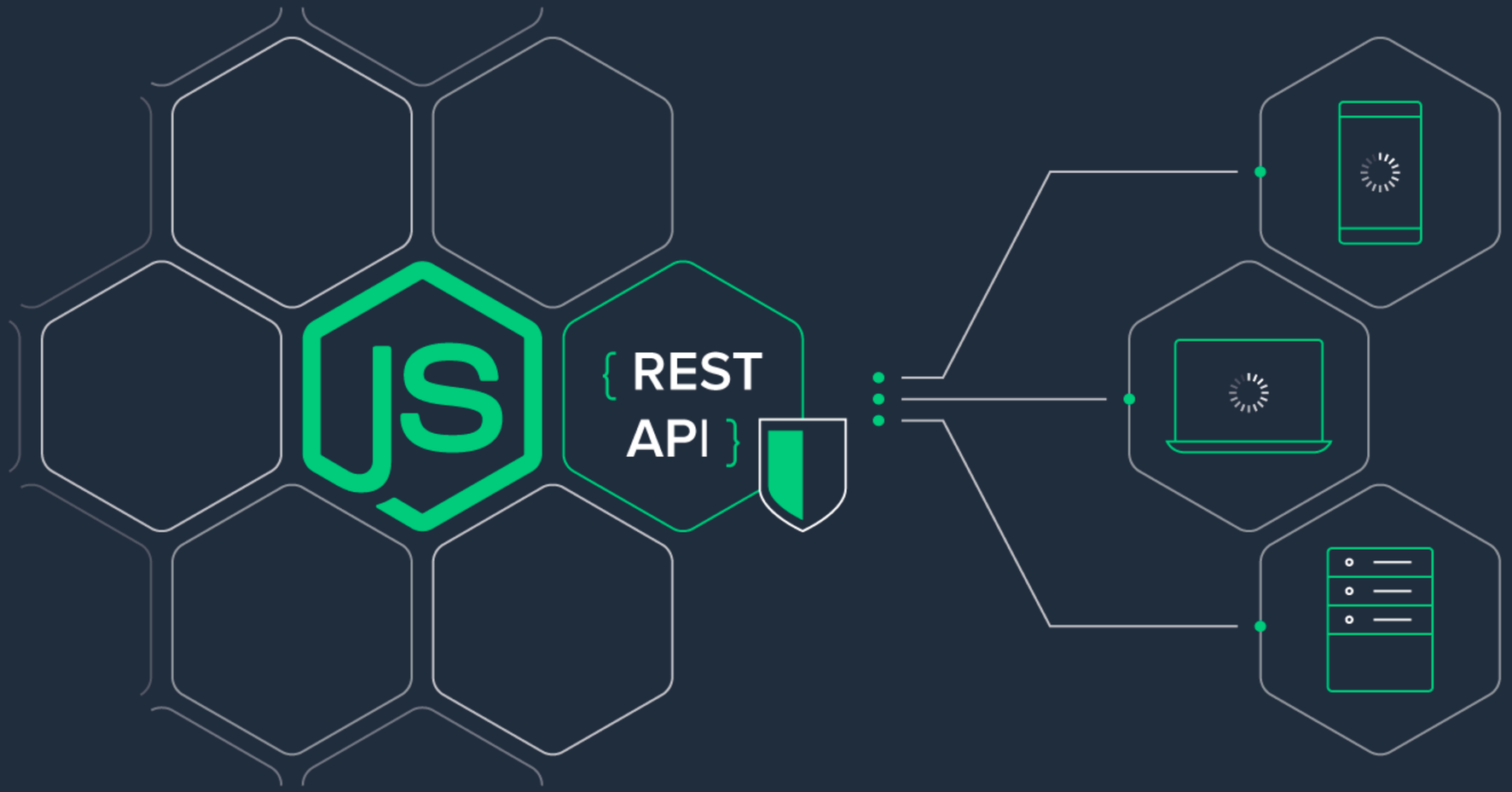
What is it used for?

- **Minimalistic Framework**
  Express is a lightweight and minimal web application framework for Node.js.
- **Routing**
  It offers a powerful routing system to handle different HTTP methods and URL patterns.
- **Middleware**
  Express uses middleware to handle various tasks, such as authentication, logging, and data parsing.
- **HTTP Utility Methods**
  Provides easy-to-use methods for interacting with the HTTP protocol.
- **Extensibility**
  Allows integration of third-party middleware and extensions for added functionality.
- **Static File Serving**
  Simplifies serving static files like CSS, JavaScript, and images.
- **Error Handling**
  Offers error handling mechanisms for a better user experience.
- **Flexible for APIs**
  Suitable for building RESTful APIs with its routing and middleware capabilities.
- **Easy to Use**
  Express.js is easy to use and requires minimal configuration, making it ideal for developers who want to build web applications quickly.

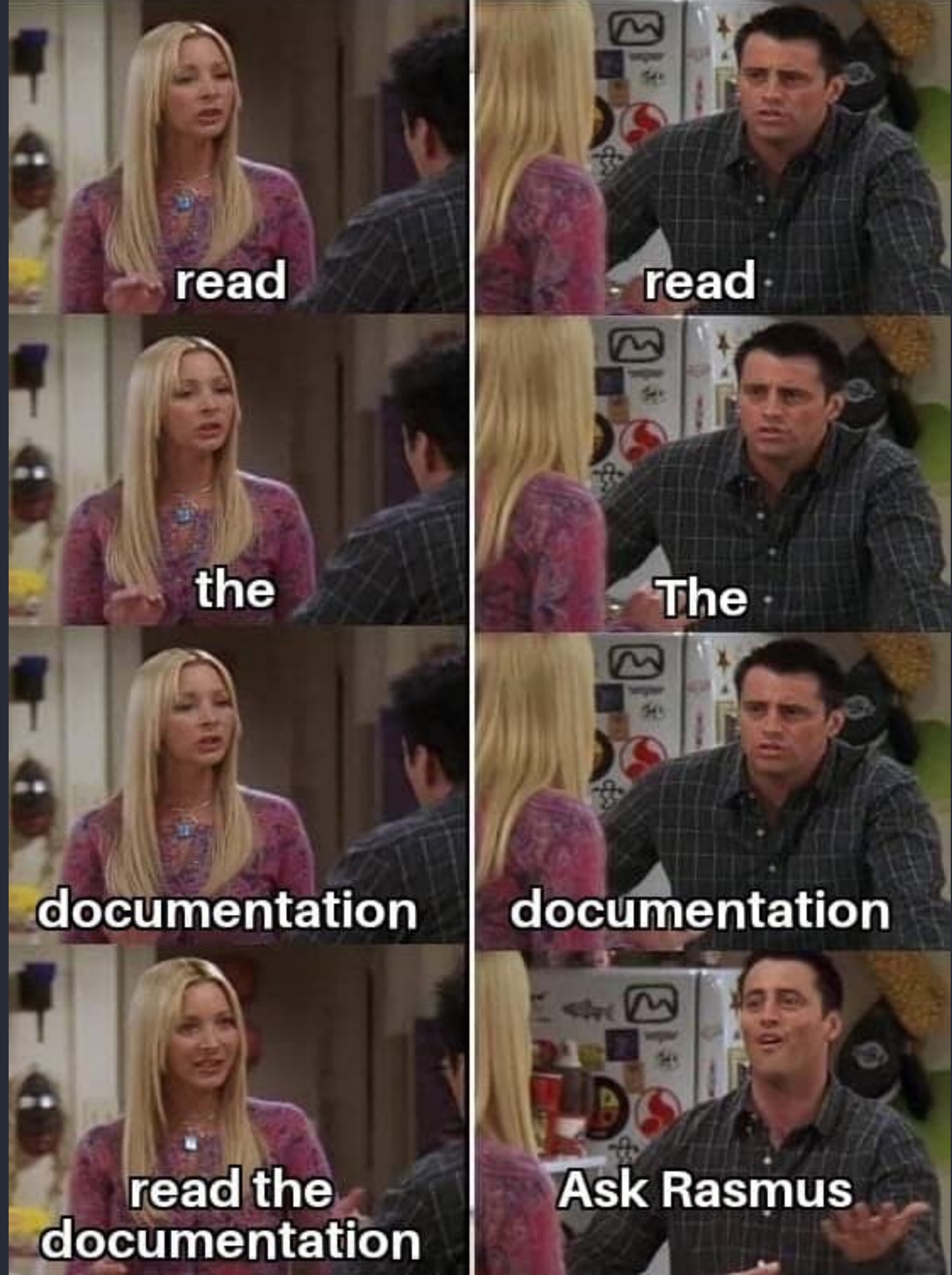
Express JS

Key Features

Streamlines and simplifies the implementation of REST API



But RACE, how do you know all that stuff?



Express JS

Routing

What’s a Router?

What’s a Router?

“It is the piece of software in charge to organize the states of the application, switching between different views.”

Routes organize how your application handles different requests, making your code structured and maintainable.

https://medium.com/@fro_g/routing-in-javascript-d552ff4d2921

Server

Client

Incoming request URL

Router

GET “/posts”

POST “/posts”

PUT “/posts”

DELETE “/posts”

Based on the request URL and method, the router decides what to do

```
app.get("/posts", (req, res) => {
    // return all posts
});

app.post("/posts", (req, res) => {
    const post = req.body;
    // Create a new post based on th
});

app.put("/posts/:id", (req, res) =>
    const postId = req.params.id;
    const post = req.body;
    // Find and update the post with
});

app.delete("/posts/:id", (req, res)
    const postId = req.params.id;
    // Find and delete the post with
});
```

Routes in Express.js

- Routes are a fundamental concept that helps you define **how your web application responds to different types of HTTP requests** at different URLs.
- Think of routes as the paths your application can take based on the URLs that users request.

Routes in Express.js

- The Express routing is built upon Node.js HTTP Module and it **simplifies the process of handling different URL paths and HTTP methods** in your web application.
- Express routing enables you to define routes for different endpoints, **simplifying request handling**.
- This enhances and streamlines code organization, and readability, and lets you focus on building functionality rather than intricate routing logic.

Routes in Express.js

- Routes determine how your application responds to specific URLs and HTTP methods.
- Each route combines an HTTP method (like GET, POST) with a URL path.
- When a client request matches a route, a corresponding route handler function is executed to process the request and send a response.

```javascript
import express from "express";
const app = express();

// Define routes using app.METHOD(PATH, HANDLER)
app.get("/posts", (req, res) => {
    // return all posts
});

app.post("/posts", (req, res) => {
    const post = req.body;
    // Create a new post based on the data in req.body
});

app.put("/posts/:id", (req, res) => {
    const postId = req.params.id;
    const post = req.body;
    // Find and update the post with postId and data from req.body
});

app.delete("/posts/:id", (req, res) => {
    const postId = req.params.id;
    // Find and delete the post with postId
});

// ...and so on

app.listen(3000, () => {
    console.log("Server is running on port 3000");
});
```

Express JS

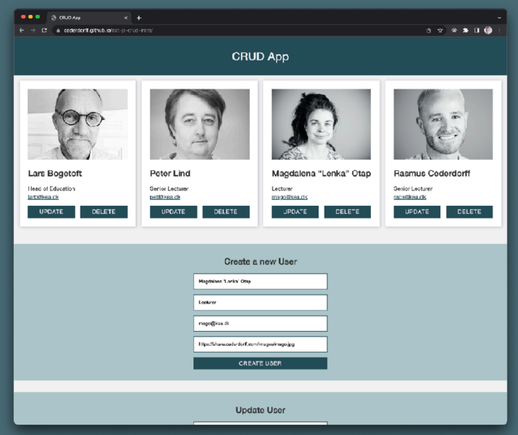
Middleware, JSON Request and Response

Web Development

WEB APP

REST API

NODE.JS



FETCH(GET, POST, PUT, DELETE)

REQUEST {JSON DATA}

RESPONSE {JSON DATA}

Express

CLIENT

SERVER

- In Express.js, middleware refers to functions that are **executed during the request-response cycle** of a web application. These functions have access to the request and response objects and can modify them, perform tasks, or control the flow of the application.
- Express.js allows you to add middleware at the application level using `app.use()` or at specific route levels. Middleware functions are executed in the order they are added, so the order matters.
- For instance, using `app.use(express.json())` is a middleware that automatically parses incoming JSON data from requests, making it accessible as JavaScript objects in your route handlers. This simplifies handling JSON data.
- Similarly, `app.use(cors())` is middleware that enables Cross-Origin Resource Sharing (CORS), allowing your server to respond to requests from different domains. This is essential for security when your frontend and backend are on separate domains.
- In both cases, middleware helps simplify and enhance the functionality of your Express application by handling common tasks automatically before your route handlers are executed.

app.js — node-users-rest-api

app.js > ...

```javascript
app.use(express.json()); // to parse JSON bodies
app.use(cors()); // Enable CORS for all routes

app.listen(port, () => {
    console.log(`App running on http://localhost:${port}`);
});

app.get("/", (request, response) => {
    response.send("Node.js Users REST API 🎉");
});

// READ all users - "/users" : GET
app.get("/users", async (request, response) => {
    const data = await fs.readFile("data.json");
    const users = JSON.parse(data);
    response.json(users);
});

// CREATE user - "/users" : POST
app.post("/users", async (request, response) => {
    const newUser = request.body;
    newUser.id = new Date().getTime();
    console.log(newUser);

    const data = await fs.readFile("data.json");
    const users = JSON.parse(data);

    users.push(newUser);
    console.log(newUser);
    fs.writeFile("data.json", JSON.stringify(users));
    response.json(users);
});
```

main+ ⊗ 0 ⚠ 0 Spaces: 4 UTF-8 LF {} JavaScript Go Live Prettier

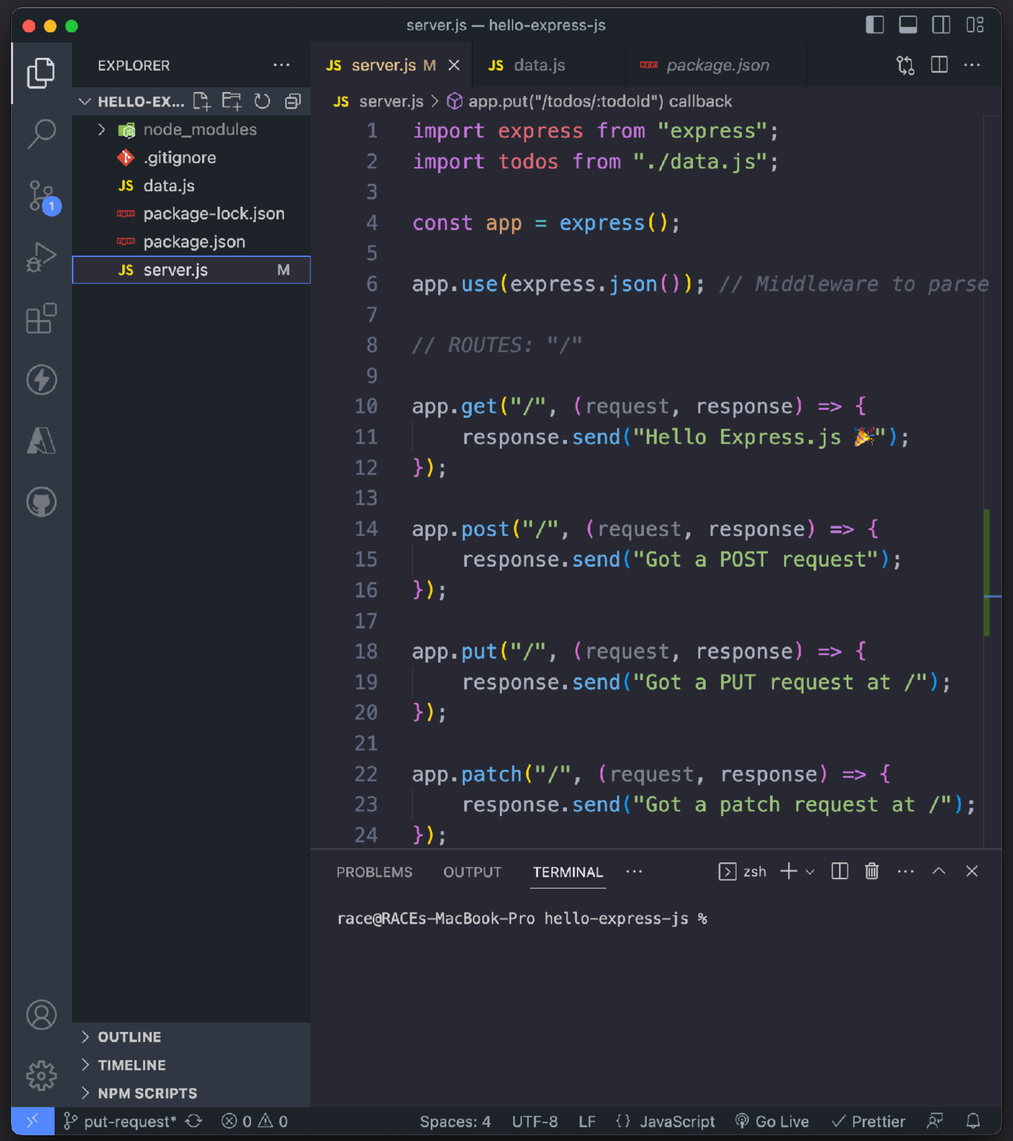
100 SECONDS OF

CORS

CROSS-ORIGIN RESOURCE SHARING

Exercises

- Getting Started with Express.js
- Node.js og MySQL



Code Every Day