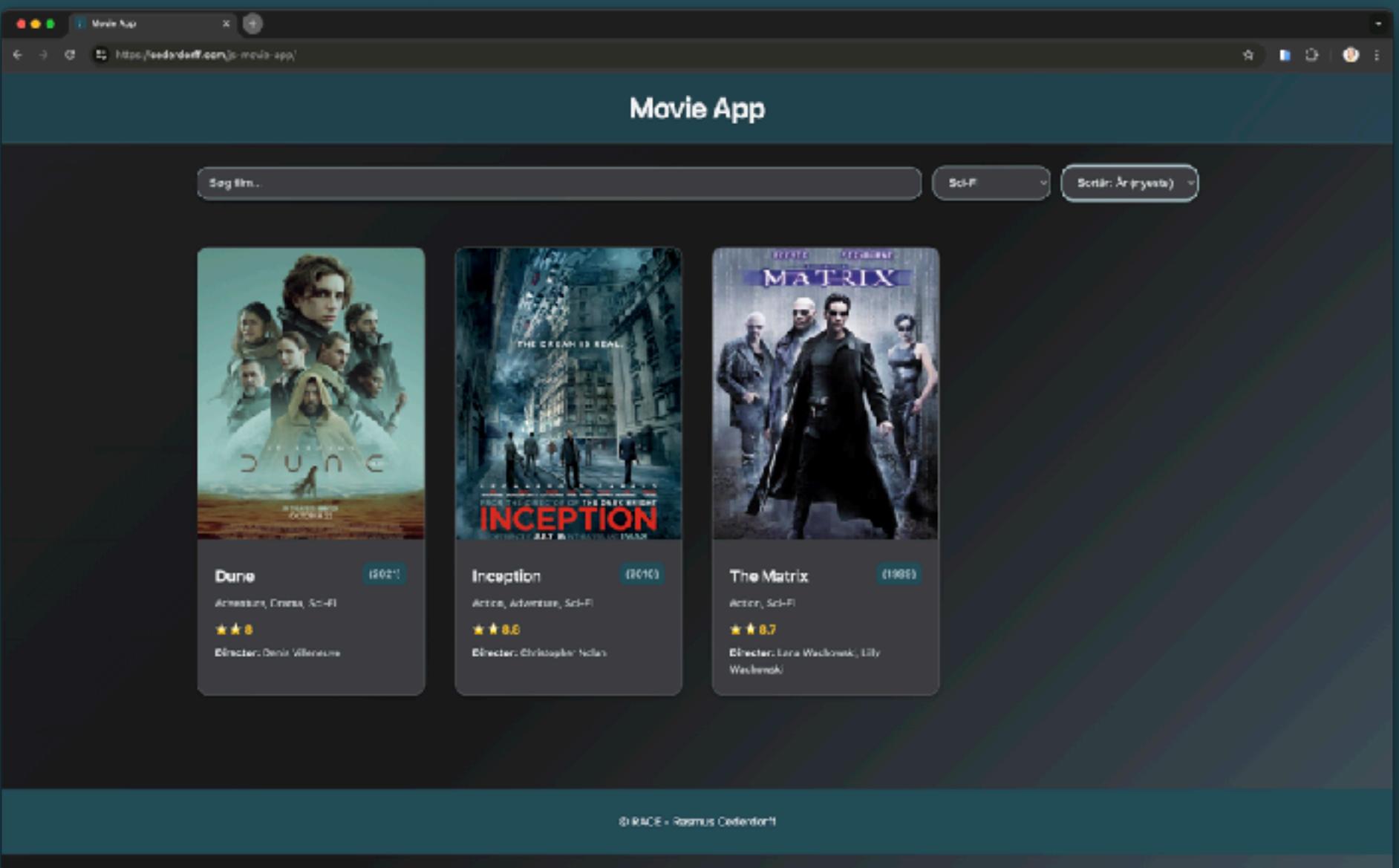
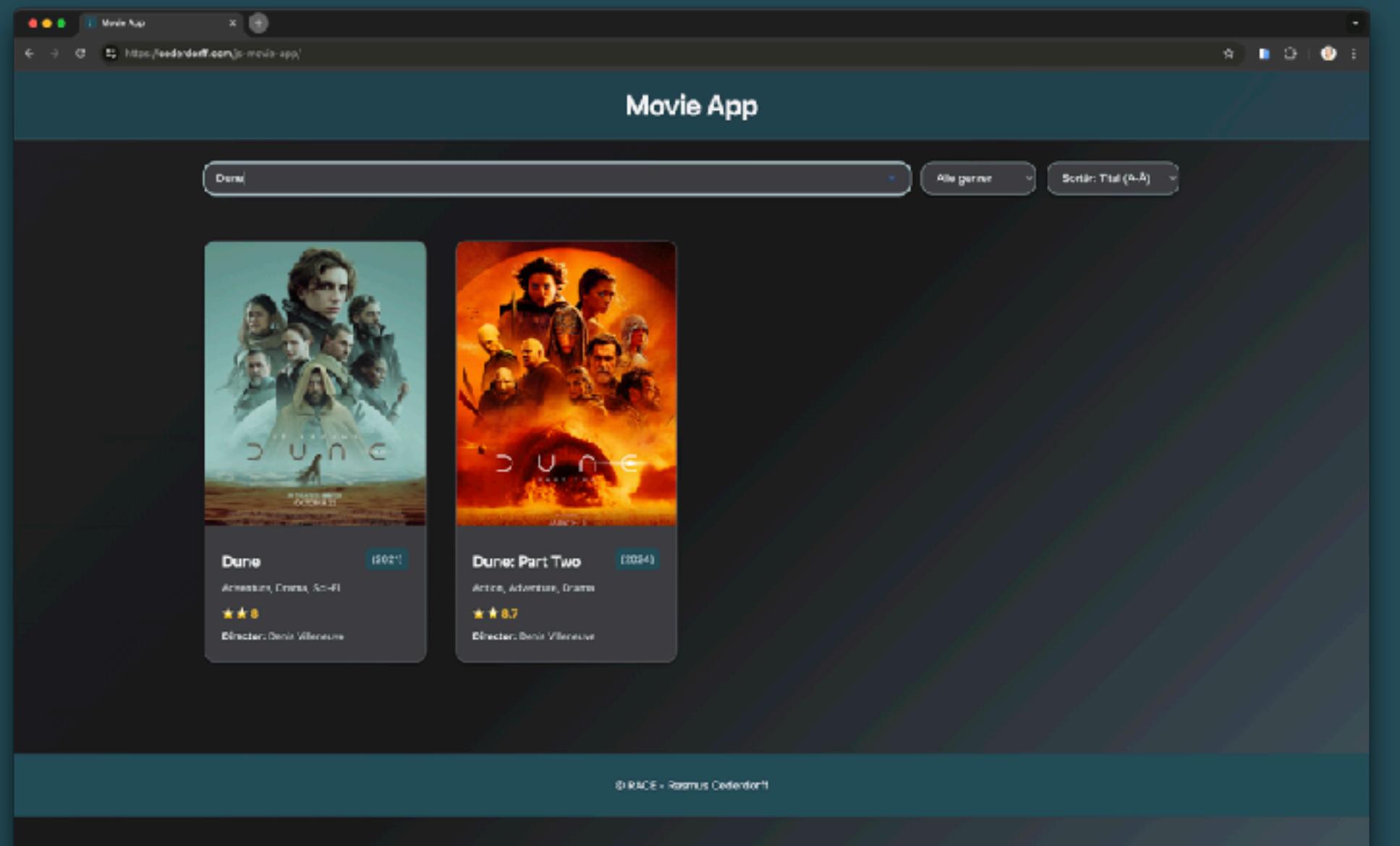
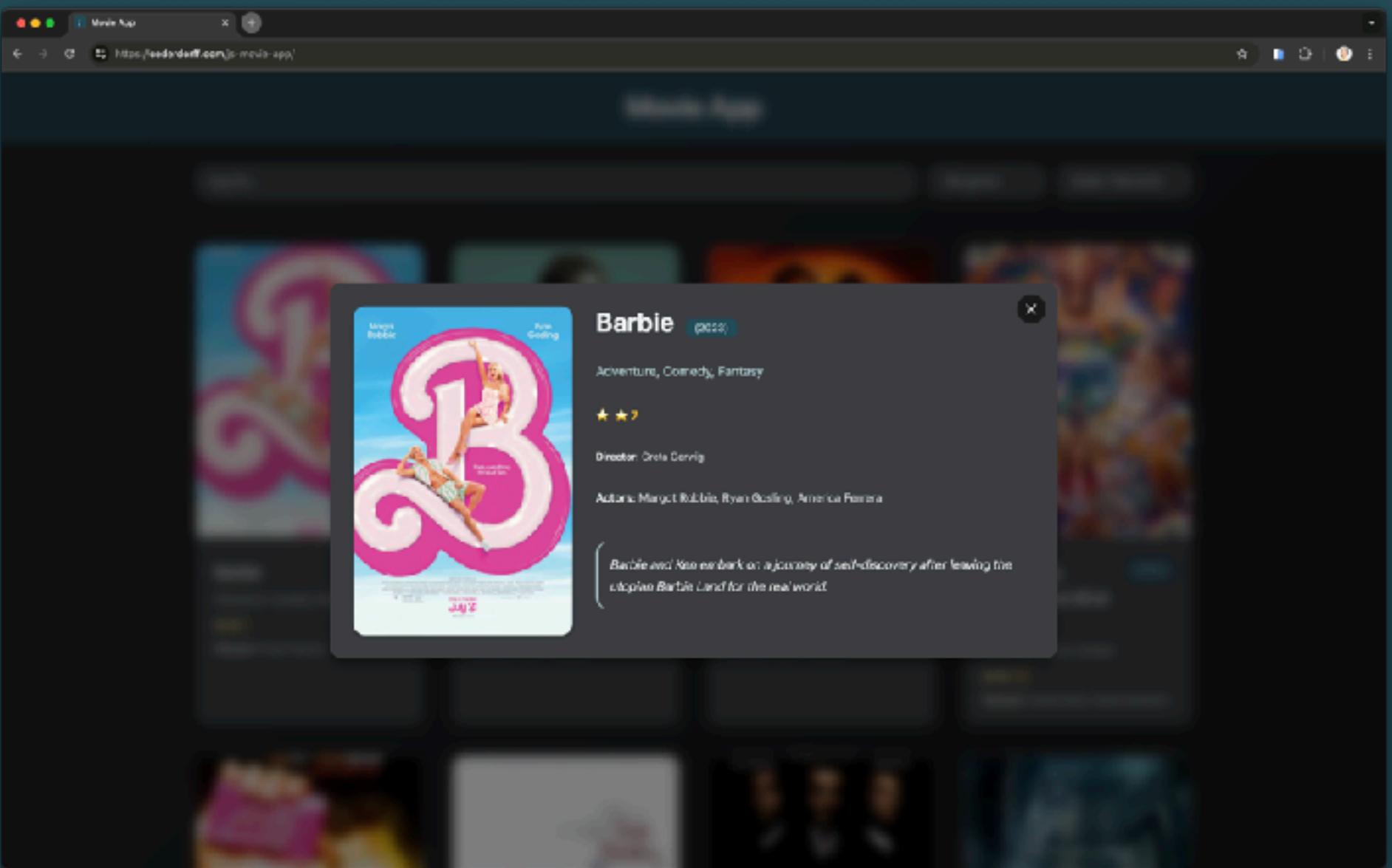
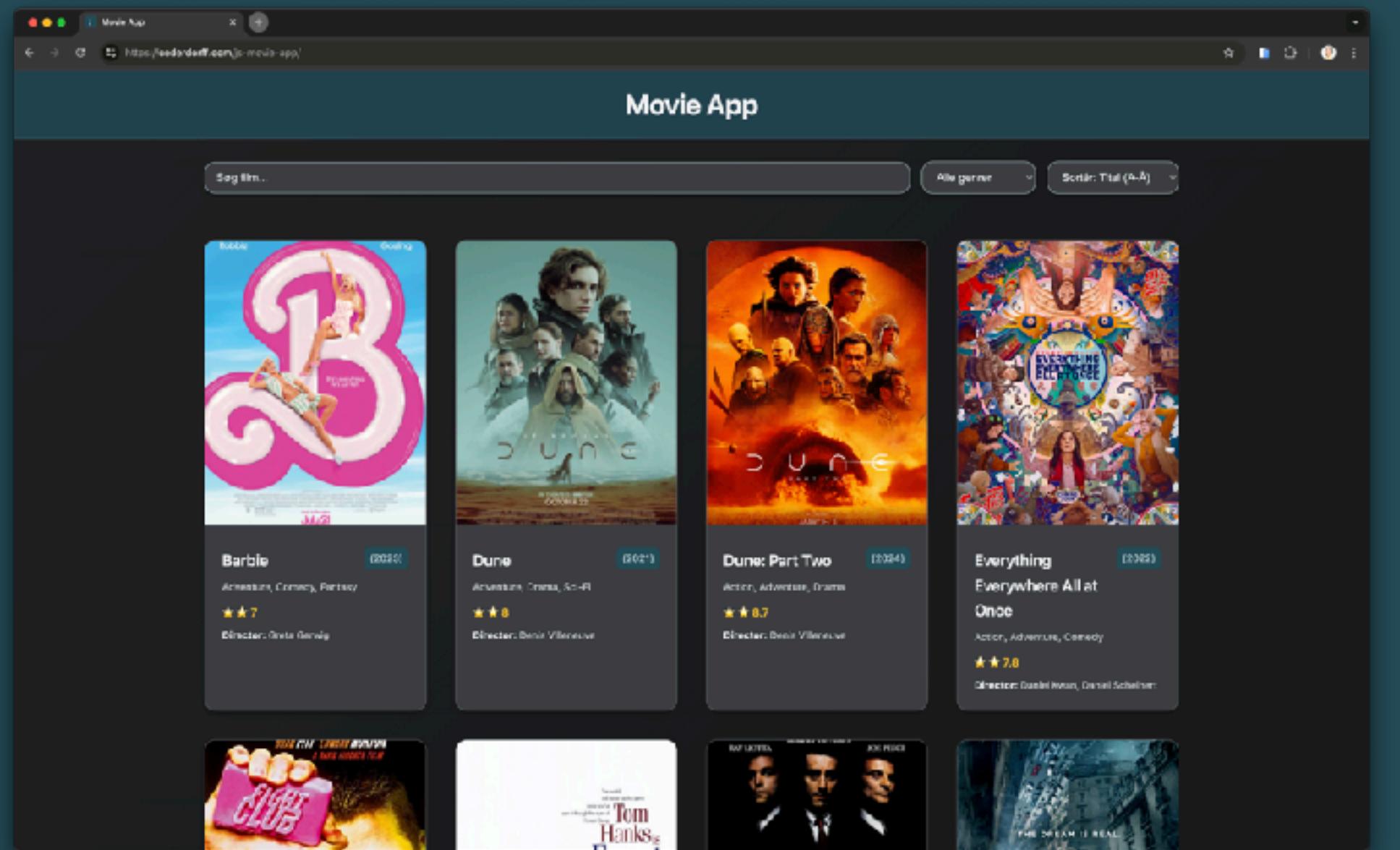


# 1. Semester - JavaScript

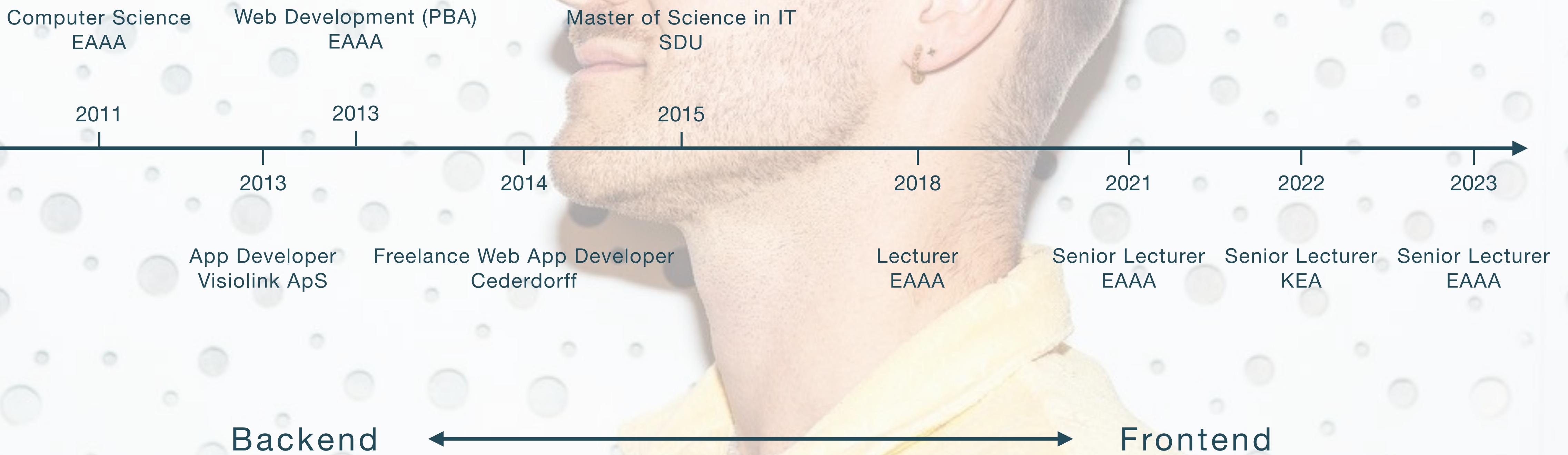
Client-Server, Variabler, Datatyper, Template String, Objects, Arrays, Funktioner, DOM-Manipulation, CSS Grid & Fetch



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.







I'm Rasmus Cederdorff  
Holstebro  
Alicia & Ida

From Holstebro  
I'm into sports  
Love (apple) gadgets, to take  
pictures & interior design  
projects

What's with my arm?





**“Learning to code requires a HUGE  
investment of time and energy.”**

<https://medium.com/martinssoft/learn-javascript-c1cca9db9015>

“

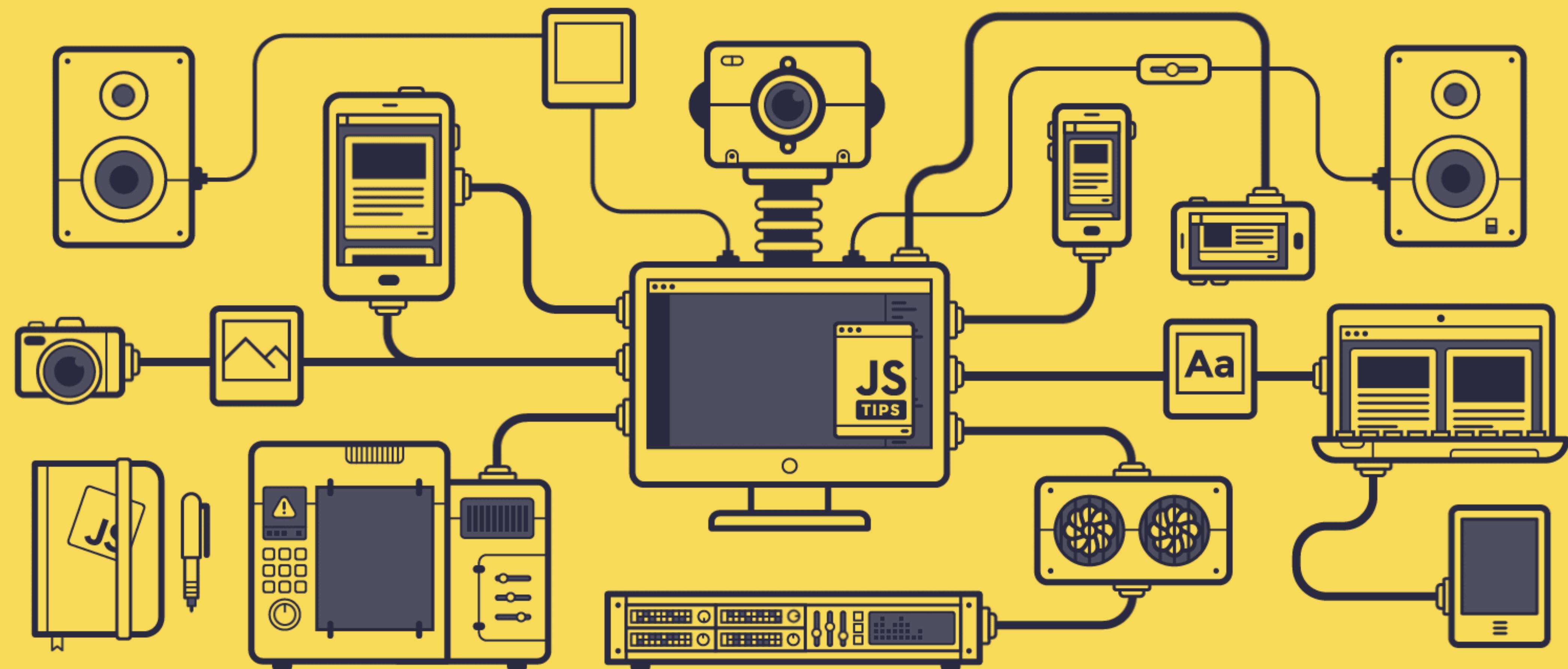
The cool thing about JavaScript is  
that you can create stuff that will put  
a smile on your face, as a developer.

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>



# **100** ***SECONDS OF***

**JS**

# JS

## 101



index.html x

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>Page Title</title>
5      <link rel="stylesheet" href="styles.css" />
6    </head>
7    <body>
8      <h1>This is a Heading</h1>
9      <p>This is a paragraph.</p>
10     <button onclick="tryMe()">Try me</button>
11     <script src="app.js"></script>
12   </body>
13 </html>
14
```

# What is JavaScript?

.. is the world's most popular programming language.

... is the programming language of the Web.

... is easy to learn.

app.js x

```
1  function tryMe() {
2    document.body.style.backgroundColor = "red";
3    document.body.style.color = "white";
4  }
5
```

... can change content of a webpage (HTML content).

... can change styling of HTML.

<https://www.w3schools.com/js/default.asp>

index.html x

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>Page Title</title>
5      <link rel="stylesheet" href="styles.css" />
6    </head>
7    <body>
8      <h1>This is a Heading</h1>
9      <p>This is a paragraph.</p>
10     <button onclick="tryMe()">Try me</button>
11     <script src="app.js"></script>
12   </body>
13 </html>
```

# With JavaScript we are able to

... build dynamic web pages and web apps.

... fetch content/ data from a backend (web service, data source, etc. ) through an API.

app.js x

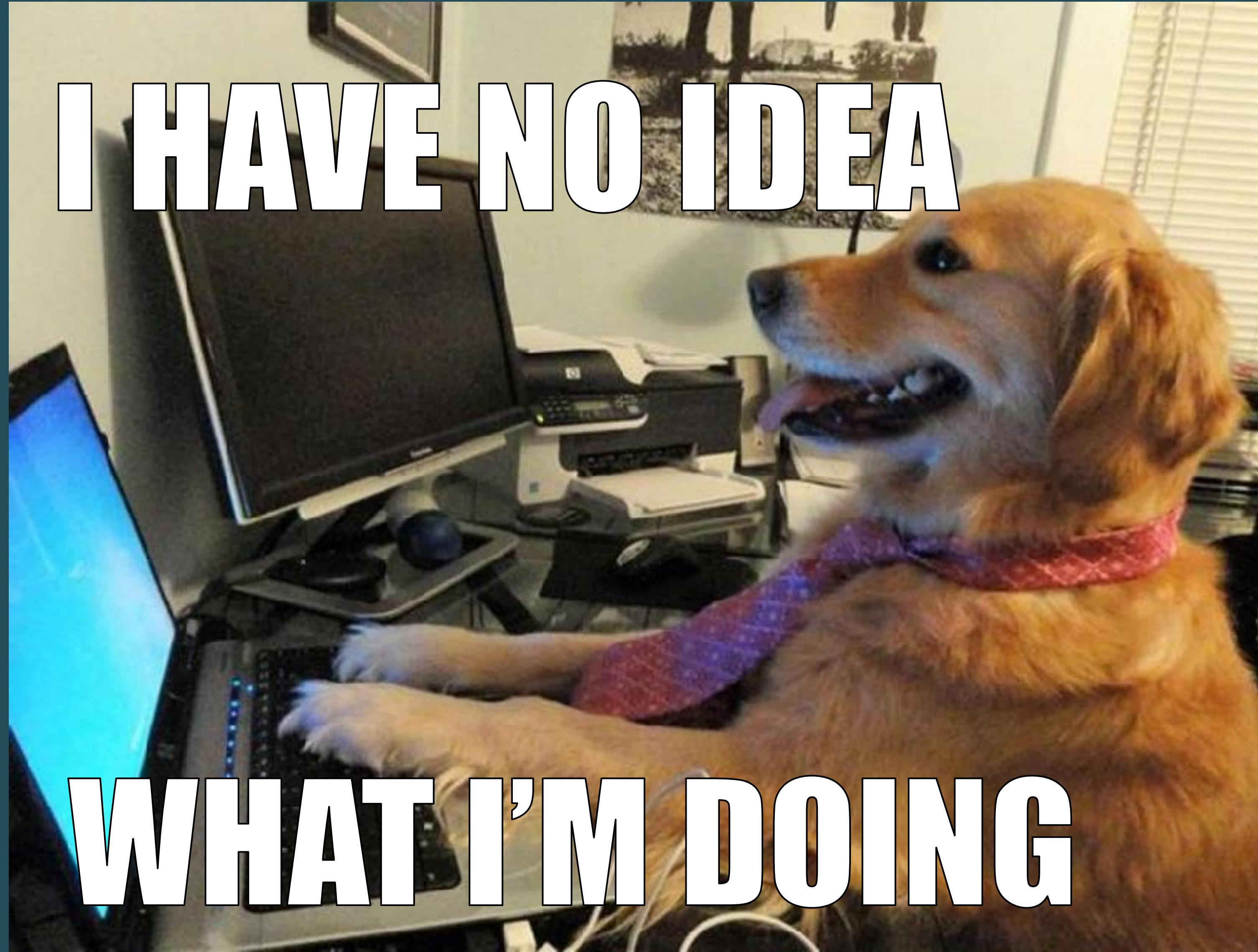
```
1  function tryMe() {
2    document.body.style.backgroundColor = "red";
3    document.body.style.color = "white";
4  }
5
```

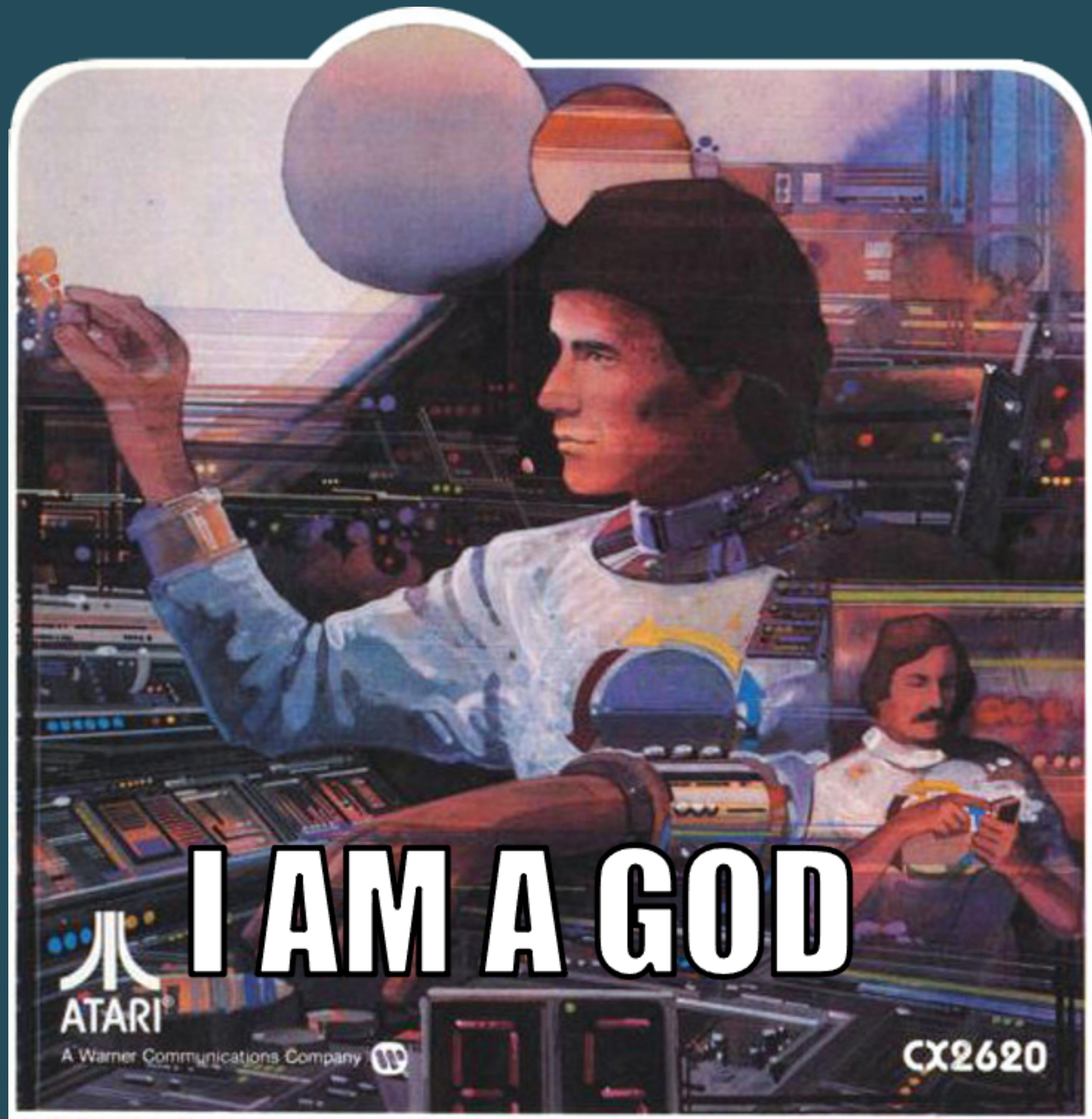
... do DOM-manipulation.

... build and develop anything 

I HAVE NO IDEA

WHAT I'M DOING



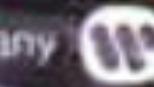


I AM A GOD



ATARI

A Warner Communications Company

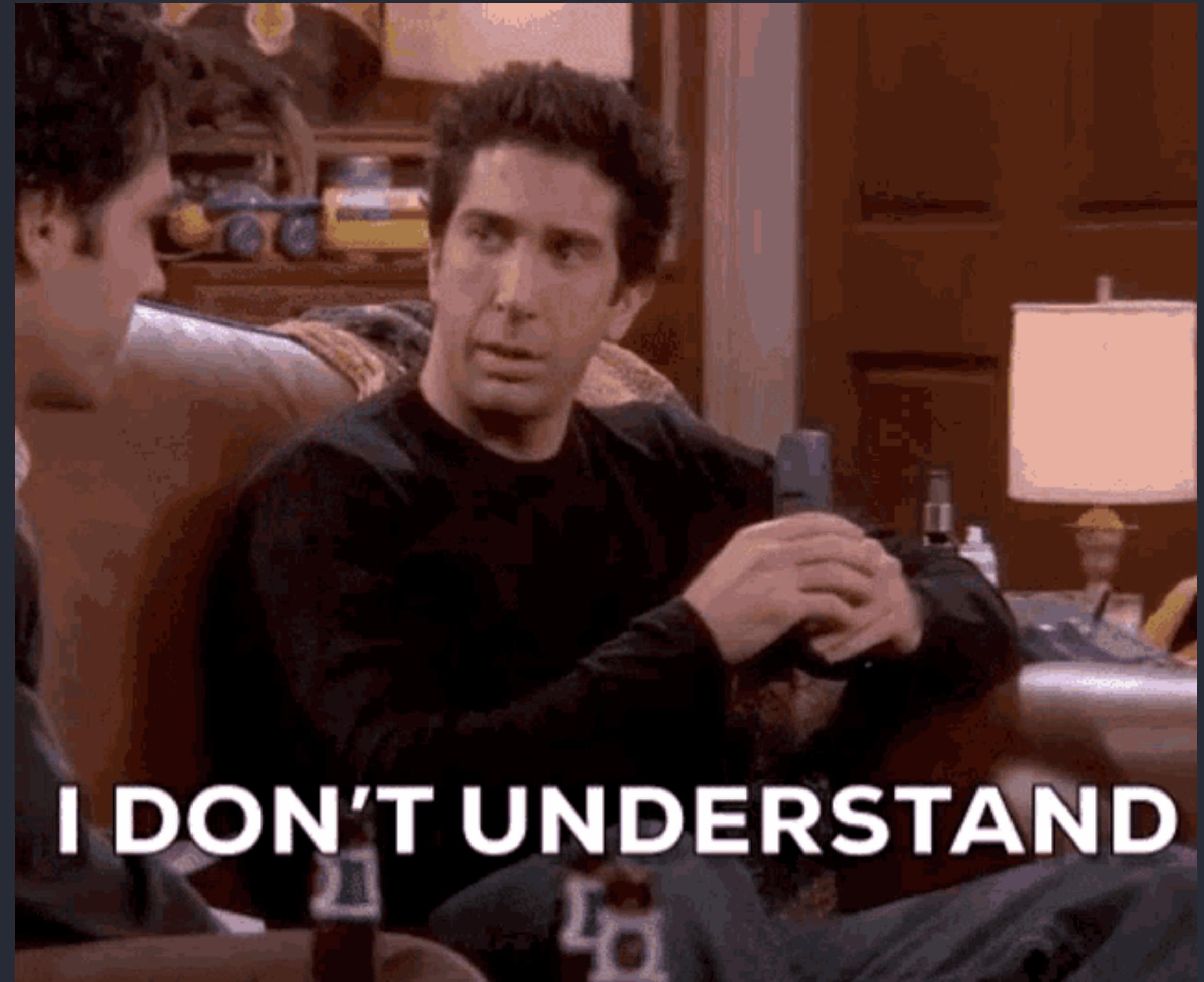


CX2620

JUST DO IT.



Please,  
Do not try to  
understand  
everything!



I DON'T UNDERSTAND

# Accept the gaps

– understanding grows with practice

# Chef on TV



# Real Chef



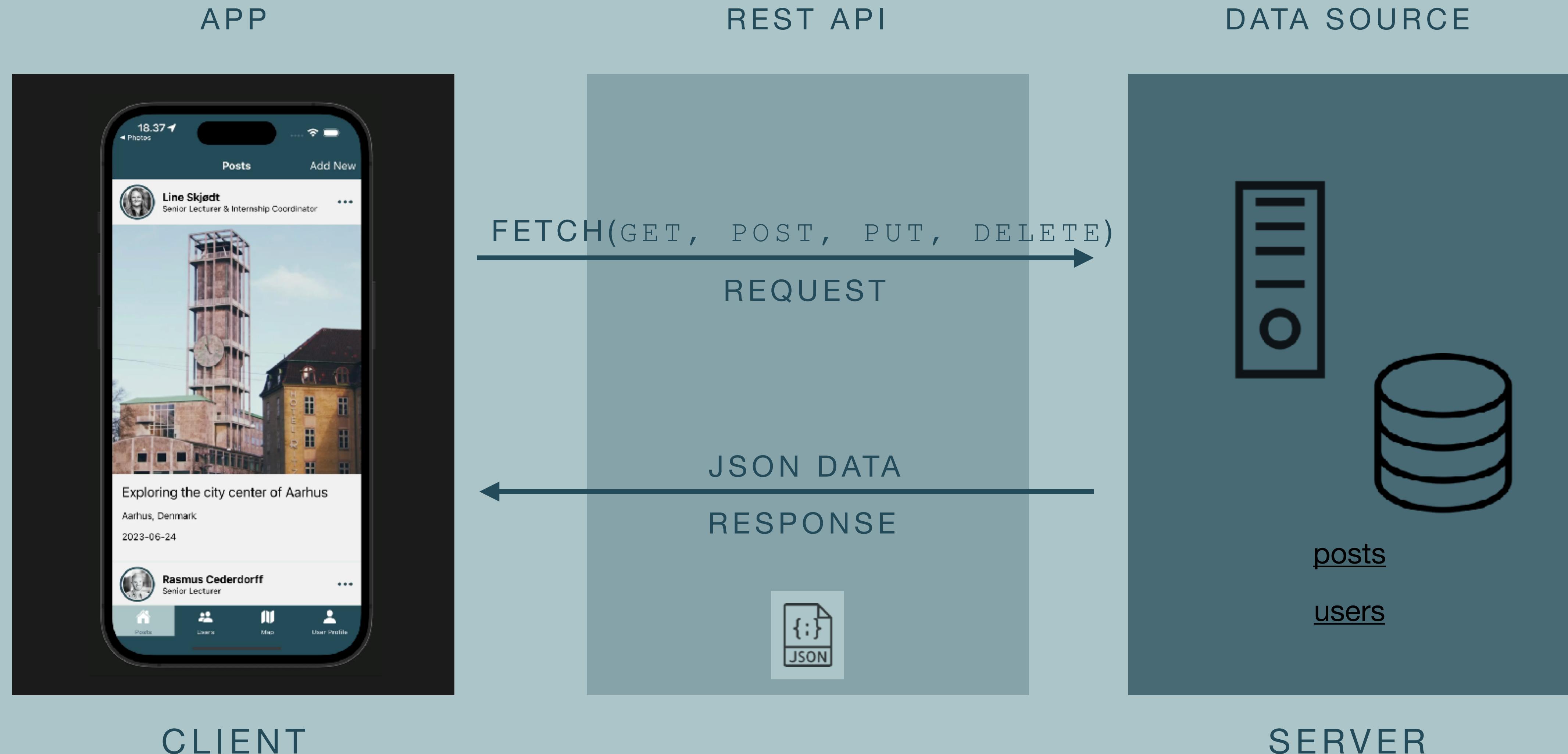
# IxD - Interaction Design

Se det som en smagsprøve på IxD

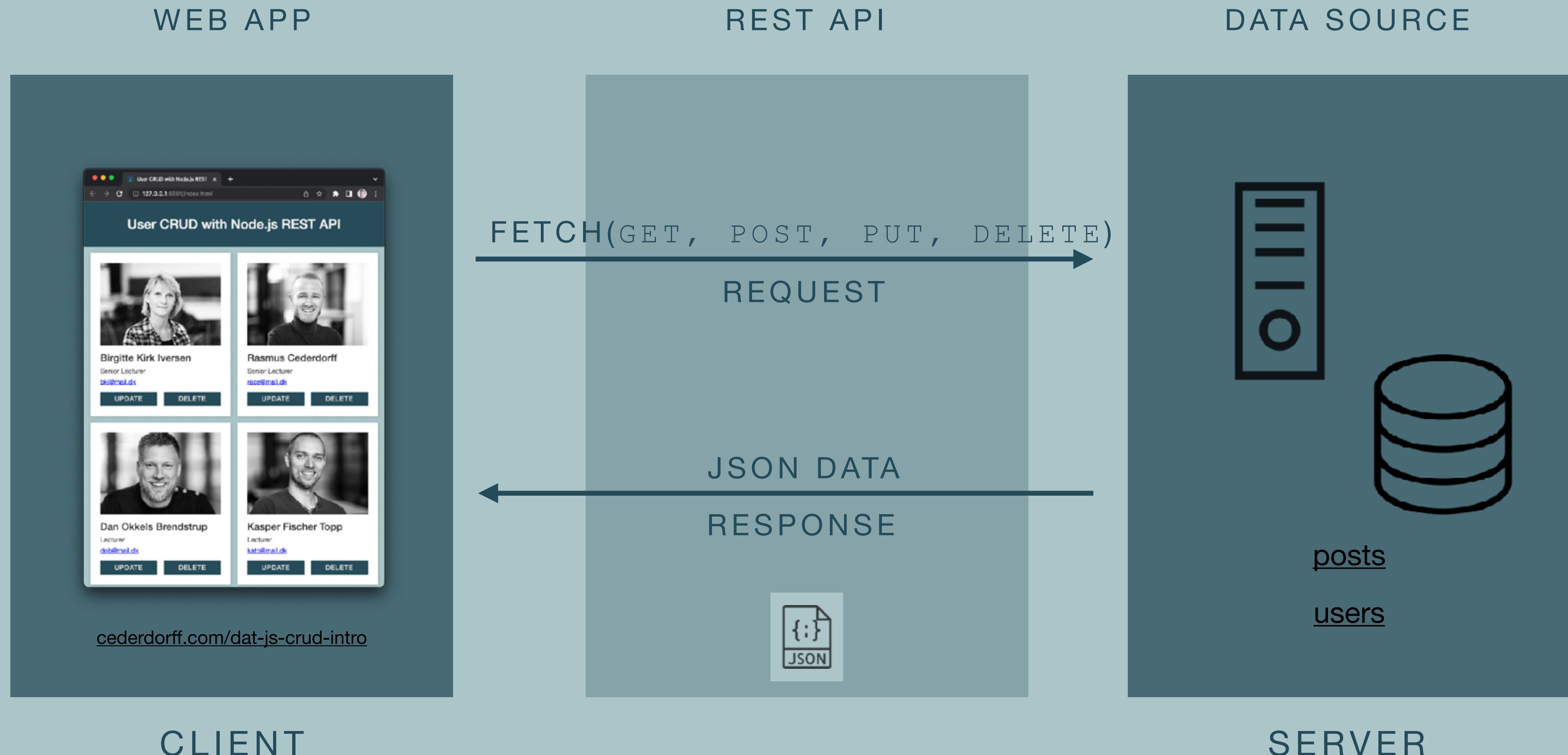
# Client Server

The Basic Architecture of the Web

# Client Server Architecture

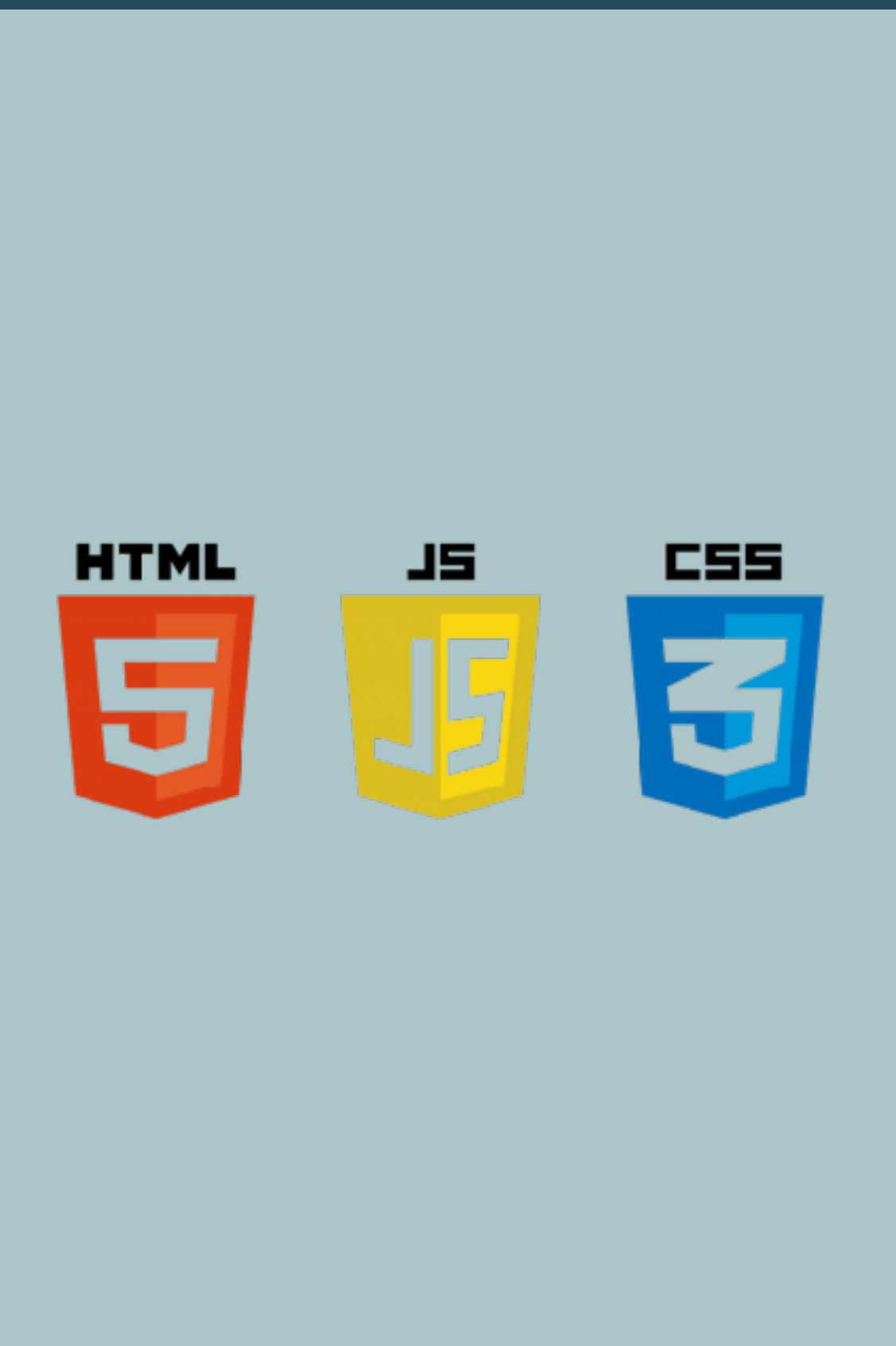


# Web Development



# Webudvikling

FRONTEND



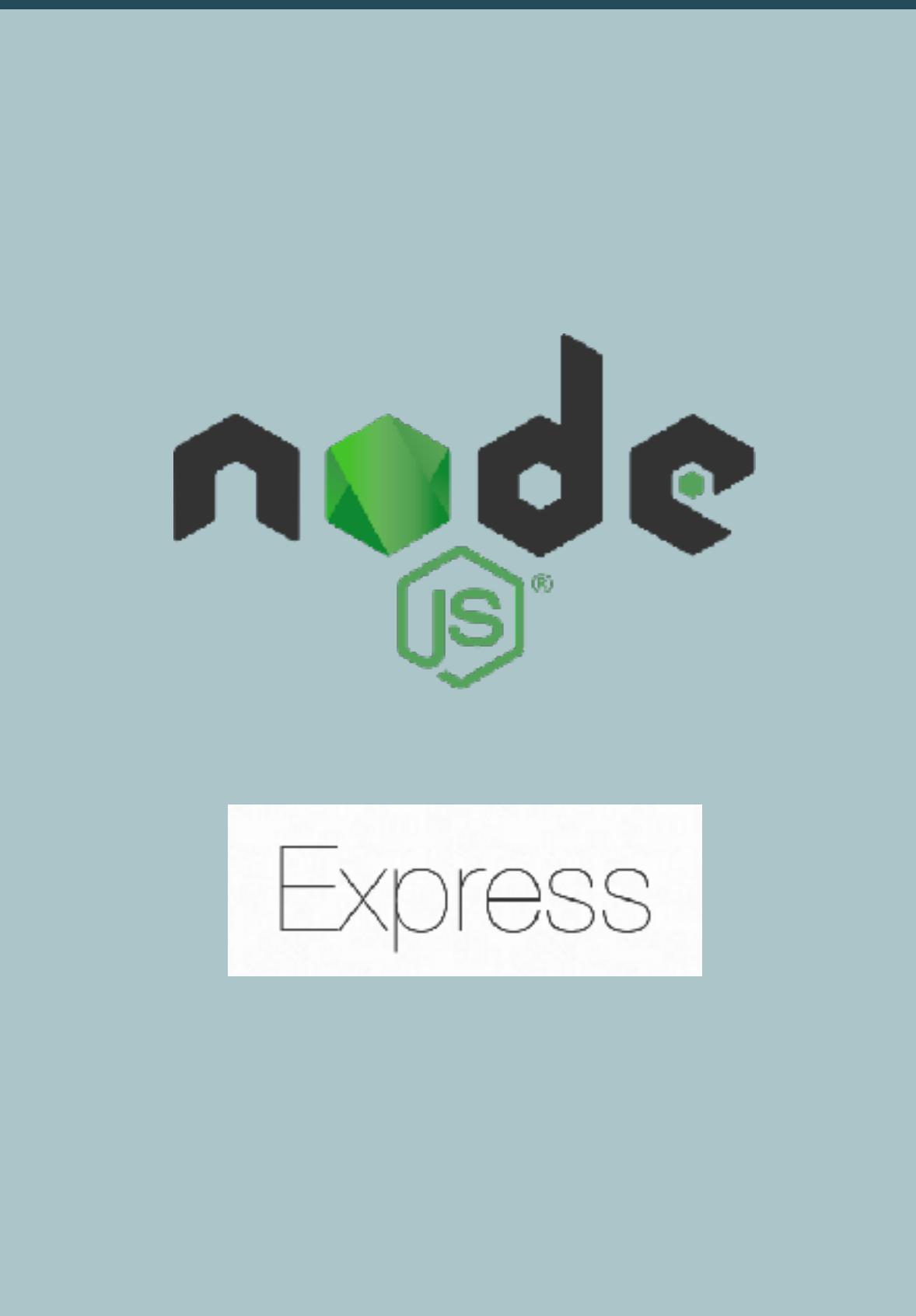
REST API

FETCH(GET, POST, PUT, DELETE)  
REQUEST

JSON DATA  
RESPONSE

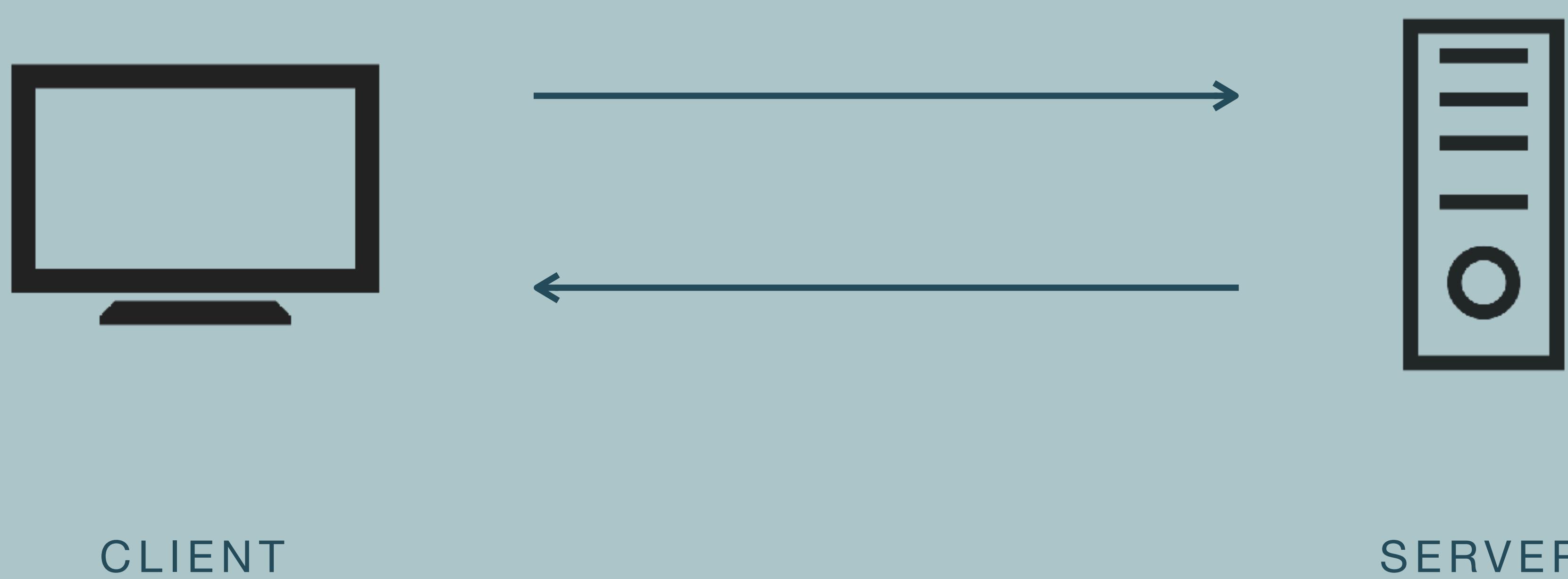


BACKEND



# Client-Server Model

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



# Client-Server Model

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

**Browsers**  
Or any type of  
program or device

CLIENT

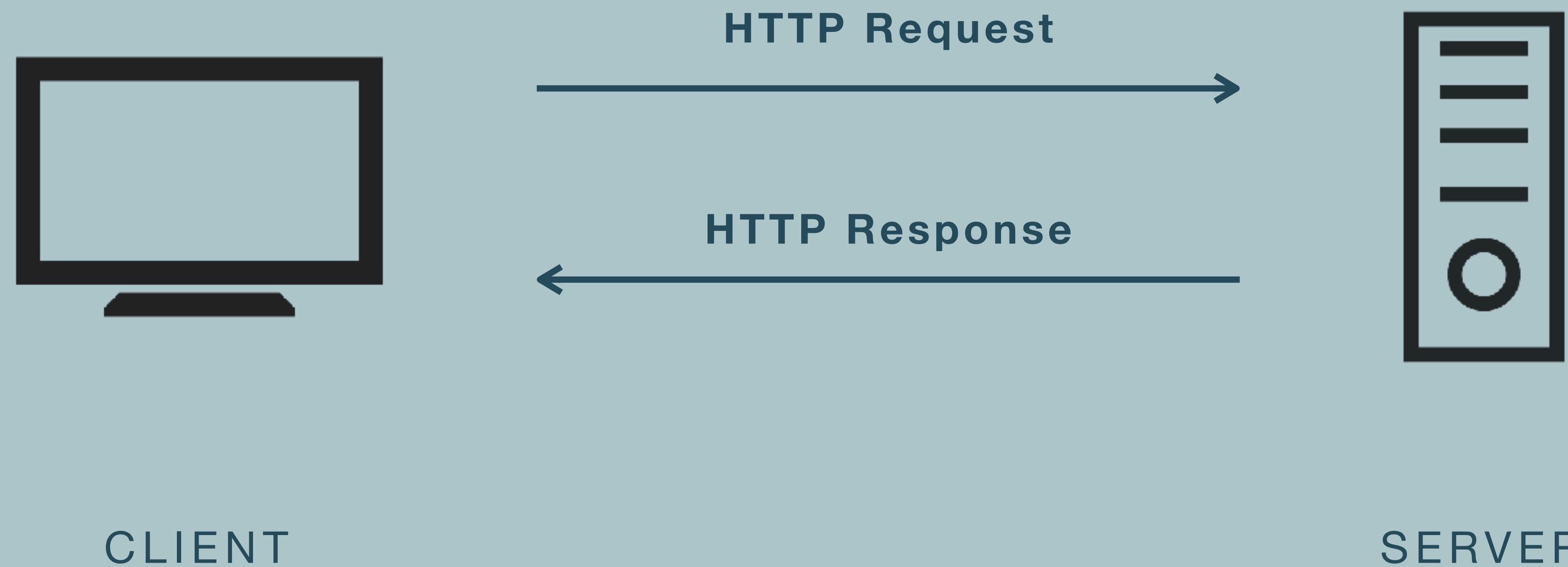


**Cloud Computers**  
Often computers  
in the cloud

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

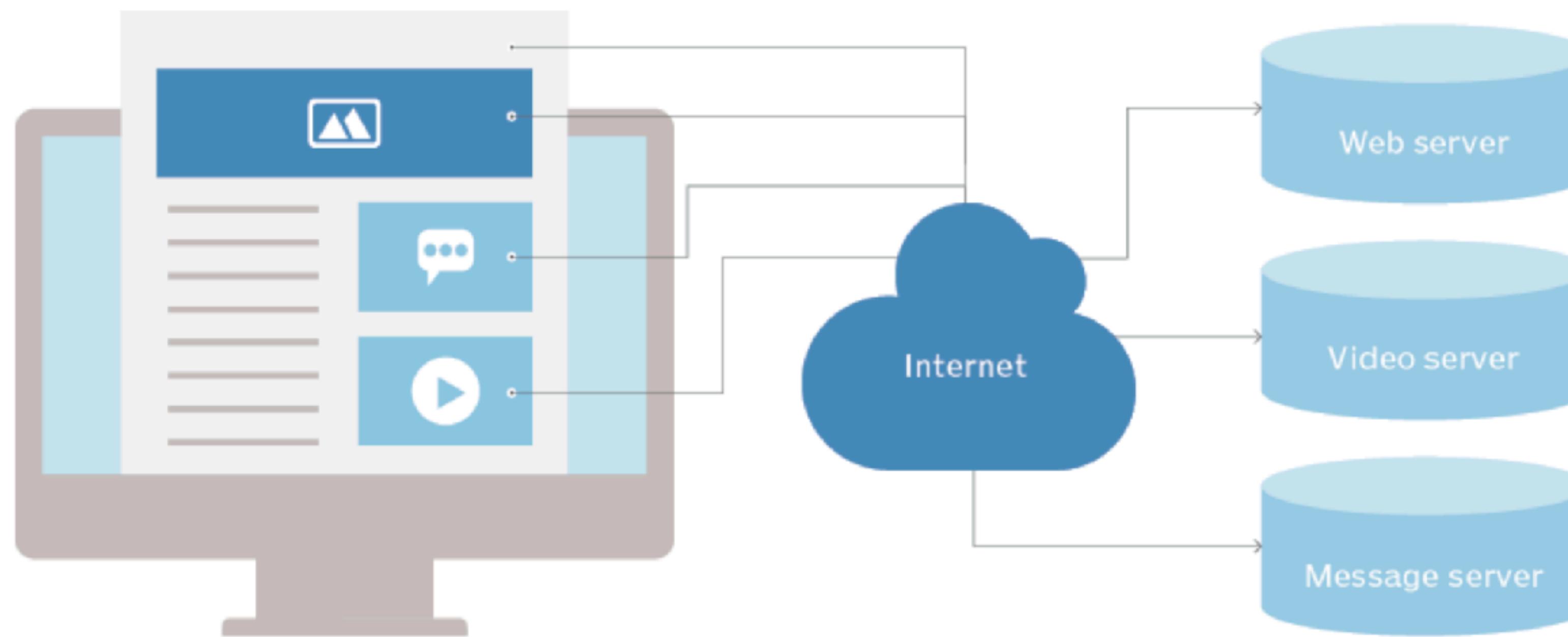
---

## 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).

# How HTTP Works



<https://www.techtarget.com/whatis/definition/HTTP-Hypertext-Transfer-Protocol>

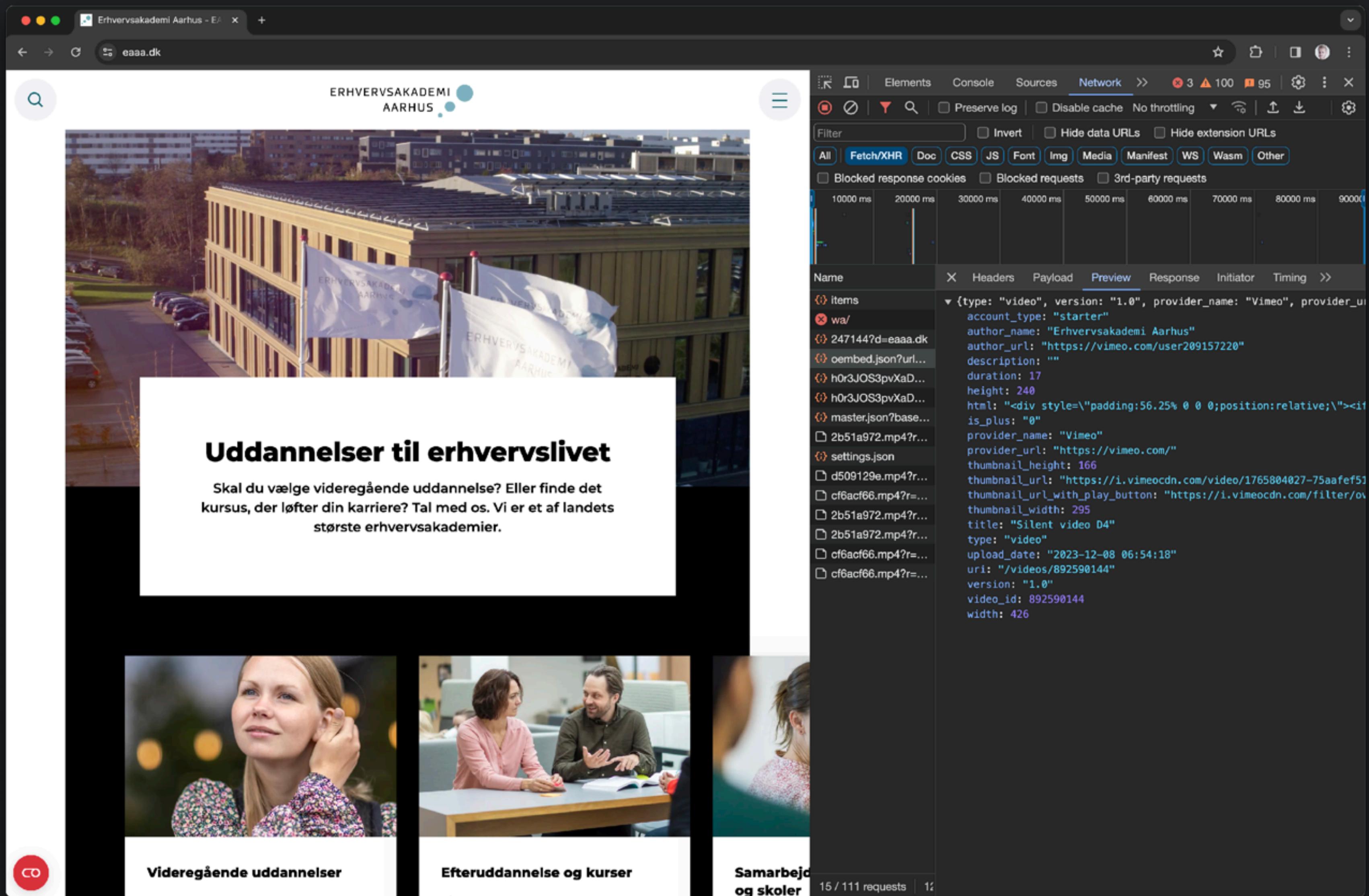
# Network Tab

The screenshot illustrates the Network tab of a browser's developer tools, overlaid on a live website. The website's content includes a header for 'ERHVERVSAKADEMI AARHUS', a main image of two people working, a central text block about education, and three smaller images below it.

**Network Tab Overview:**

- Timeline:** Shows the sequence of requests over time, with markers for 1000 ms, 2000 ms, 3000 ms, 4000 ms, 5000 ms, and 6000 ms.
- Table Headers:** Name, Method, Status, Type, Initiator, Size, T.., Waterfall.
- Table Data:** A list of requests including:
  - 1.gif?dgi=70fb8fdd-4...
  - heatmaps.js
  - page-correct.js
  - ?a=h0r3JOS3pvXaDa...
  - favicon.ico
  - h0r3JOS3pvXaDabSjt...
  - h0r3JOS3pvXaDabSjt...
  - cast\_sender.js?loadC...
  - 1765804027-75aafef...
  - master.json?base64\_i...
  - 1765804027-75aafef...
  - cast\_framework.js
  - cast\_sender.js
  - settings.json
  - widgetIcon.min.js
  - 2b51a972.mp4?r=dX...
  - d509129e.mp4?r=dX...
  - cf6acf66.mp4?r=dXM...
  - 2b51a972.mp4?r=dX...
  - 2b51a972.mp4?r=dX...
  - cf6acf66.mp4?r=dXM...
  - cf6acf66.mp4?r=dXM...
  - collect?v=2&tid=G-D...
- Metrics:** 91 requests, 12.8 MB transferred, 16.2 MB resources, Finish: 5.47 s, DOMContentLoaded: 290 ms.

# Network Tab



ERHVERVSAKADEMI  
AARHUS

Kom til u-days 20.-22. februar

Besøg vores 29 videregående uddannelser – og bliv klogere på dit studievalg.

Læs om u-days her

Uddannelser til erhvervslivet

Tal du vælge videregående uddannelse? Eller finde det kursus, der løfter din karriere? Tal med os. Vi er et af landets største erhvervsakademier.

Network Tab

Name	Method	Status	Type	Initiator	Size	T...
www.eaaa.dk	GET	200	docu...	Other	23.2 ...	4...
main.css?v=1Tq49VSe...	GET	200	style...	www.eaaa.dk/	(me...)	0...
logo-da-small.svg	GET	200	svg+...	www.eaaa.dk/	(me...)	0...
logo-da-big.svg	GET	200	svg+...	www.eaaa.dk/	(me...)	0...
main.js?v=g2A4y0KTr...	GET	200	script	www.eaaa.dk/	(me...)	0...
gtm.js?id=GTM-P3PZXT	GET	200	script	(index):36	(me...)	0...
JTUSIg1_i6t8kCHKm...	GET	200	font	main.css	(me...)	0...
videregående-uddann...	GET	200	webp	(index):2506	(me...)	0...
efteruddannelser-og...	GET	200	webp	(index):2536	(me...)	0...
erhvervsakademি-aar...	GET	200	webp	(index):2566	(me...)	0...
data:image/svg+xml;...	GET	200	svg+...	main.css	(me...)	0...
videobanner_cover_im...	GET	200	jpeg	(index):2761	(me...)	0...
data:image/svg+xml;...	GET	200	svg+...	main.css	(me...)	0...
afae0c2c100d6b5123...	GET	200	png	main.css	(me...)	0...
items	GET	200	fetch	main.js?v=g2A...	61 B	1...
data:image/svg+xml;...	GET	200	svg+...	main.css	(me...)	0...
data:image/svg+xml;...	GET	200	svg+...	main.css	(me...)	0...
data:image/svg+xml;...	GET	200	svg+...	main.css	(me...)	0...
data:image/svg+xml;...	GET	200	svg+...	main.css	(me...)	0...
uc.js?cbid=70fb8fd...	GET	200	script	gtm.js?id=GT...	26.5 ...	3...
ad03e075-63e2-4fb0...	GET	304	script	VM300:2	308 B	7...
20ab1c93b09b00989...	GET	200	script	main.js?v=g2A...	(me...)	0...
81f573bfa84afb36...	GET	200	script	main.js?v=g2A...	(me...)	0...
oembed.json?url=http...	GET	200	xhr	20ab1c9...js:2	2.3 kB	2...
base4 min.html	GET	200	docu...	click	1	

102 requests | 12.9 MB transferred | 17.4 MB resources | Finish: 2.59 s | DOMCo

# Network Tab

## Investigating Network Activity in the Browser

# Variabler

const & let

# Variables

... are used to store data (values, objects, collections) in the memory

# Variables

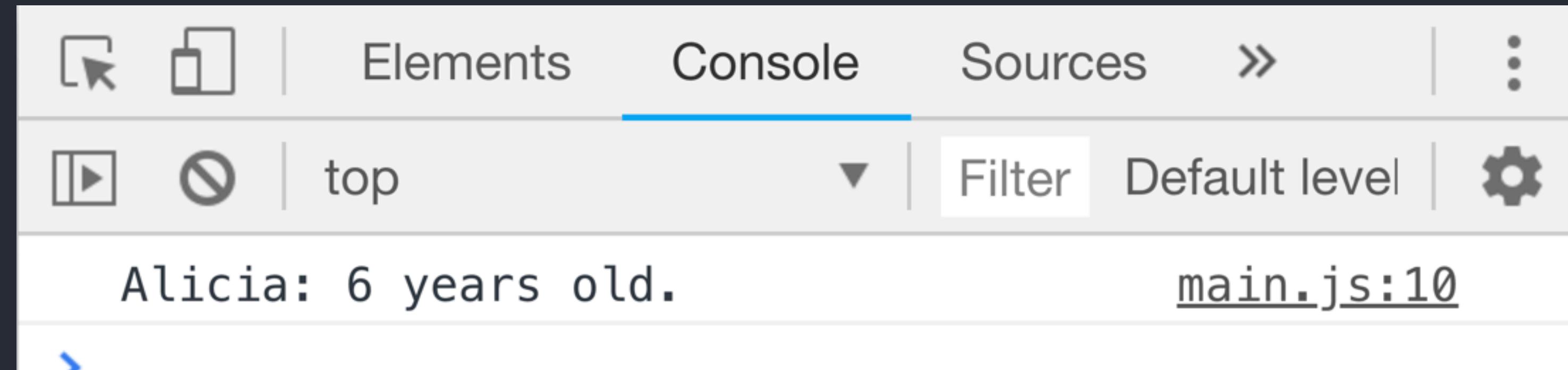
... a simple way to store, get and set data in  
your code.

# Variables

Store data in the memory

```
let name = "Alicia";
let age = 6;

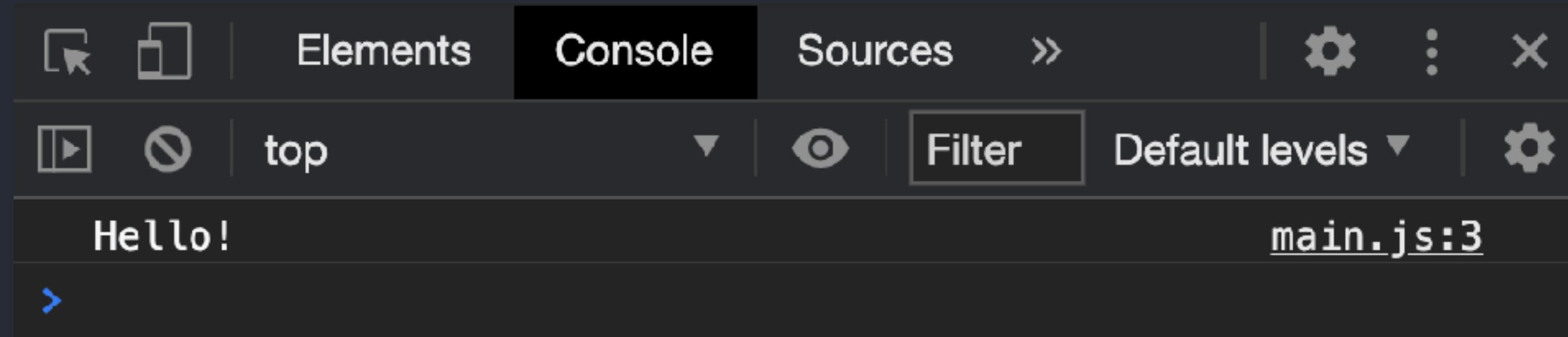
console.log(name + ": " + age + " years old.");
```



# Variables

Store data in the memory

```
let message = "Hello!";
console.log(message);
```

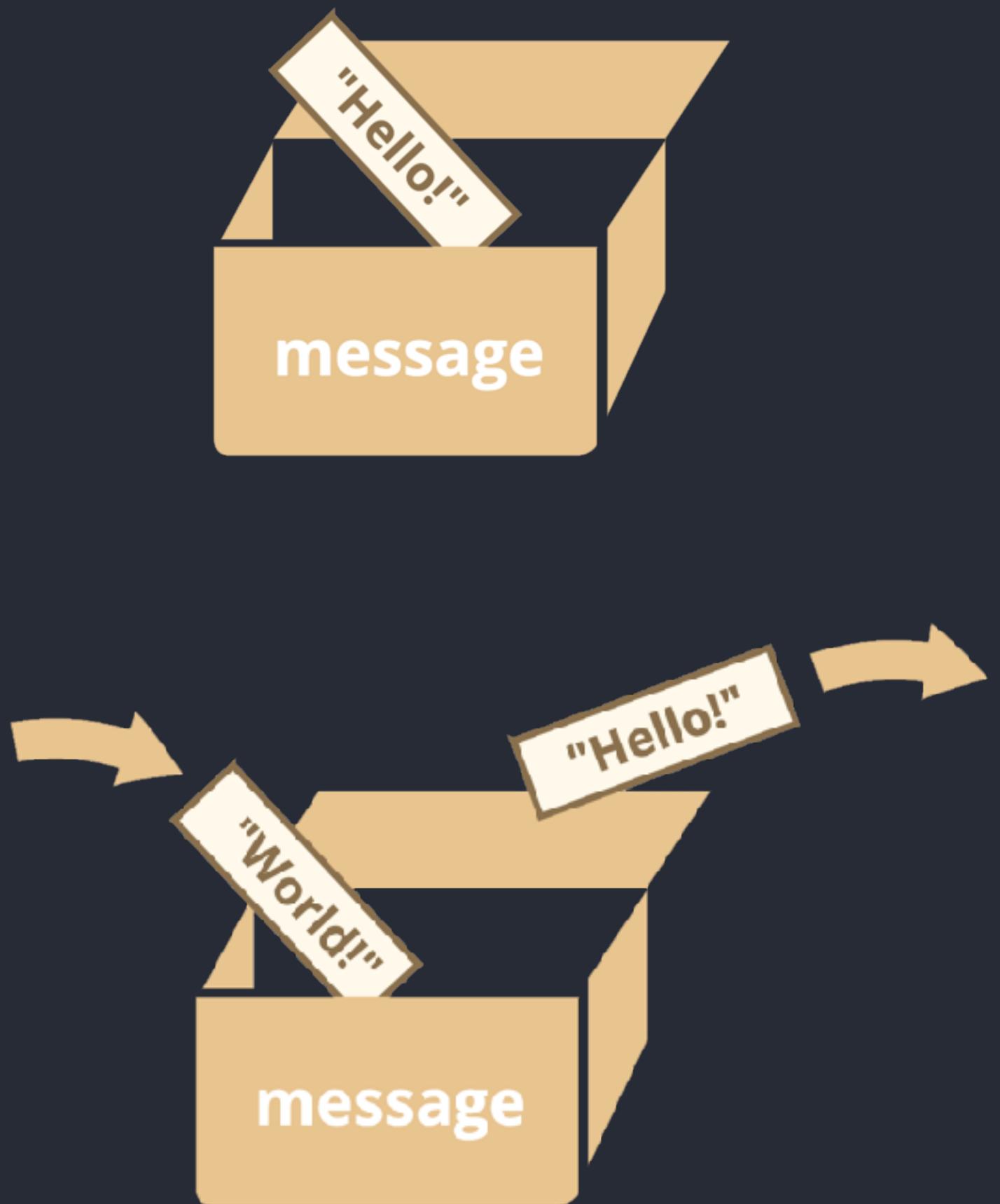


# Variables

Store data in the memory

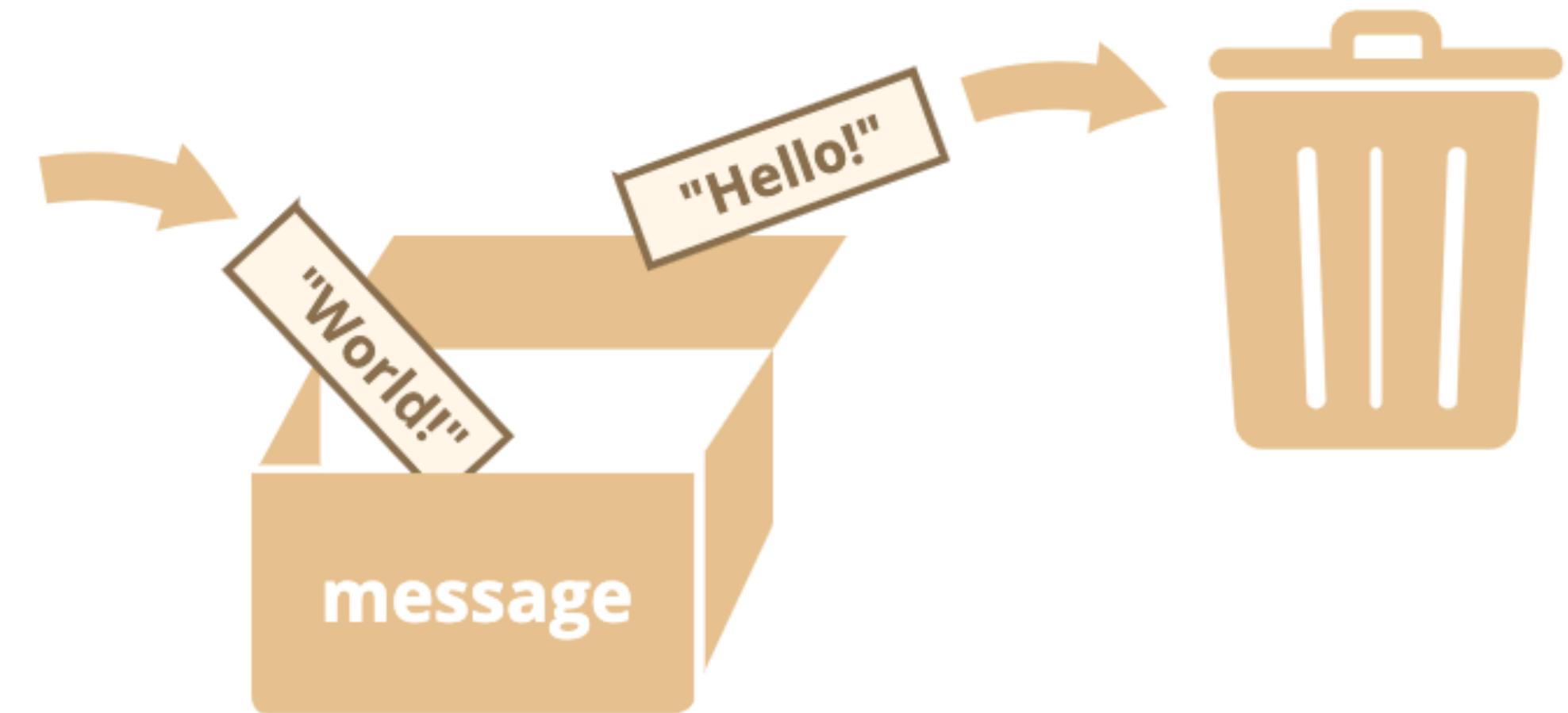
```
let message = "Hello!";
console.log(message);
```

```
message = "World!";
console.log(message);
```



# Variable

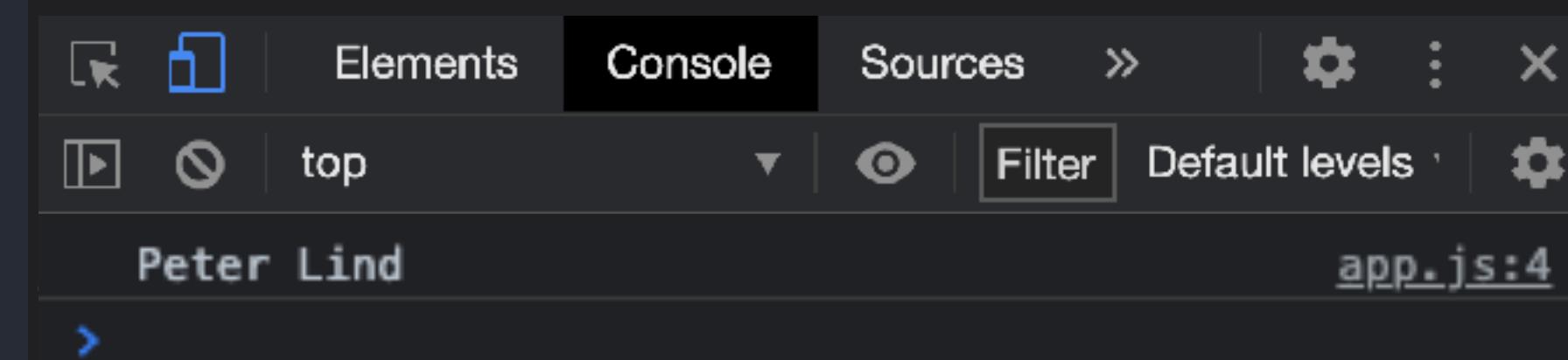
A variable is a “named storage” and stored in the memory of the browser.



We can change the value of the variables as many times as we want.

# Variables & UI

```
let fullName = "Peter Lind";
console.log(fullName);
document.querySelector("#fullName_container").textContent = fullName;
```



# Variables & UI

```
let fullName = "Peter Lind";
console.log(fullName);
document.querySelector("#fullName_container").textContent = fullName;
```

```
<body>
  <header>
    <h1 id="fullName_container"></h1>
  </header>
  <script src="app.js"></script>
</body>
```



# Variables & UI

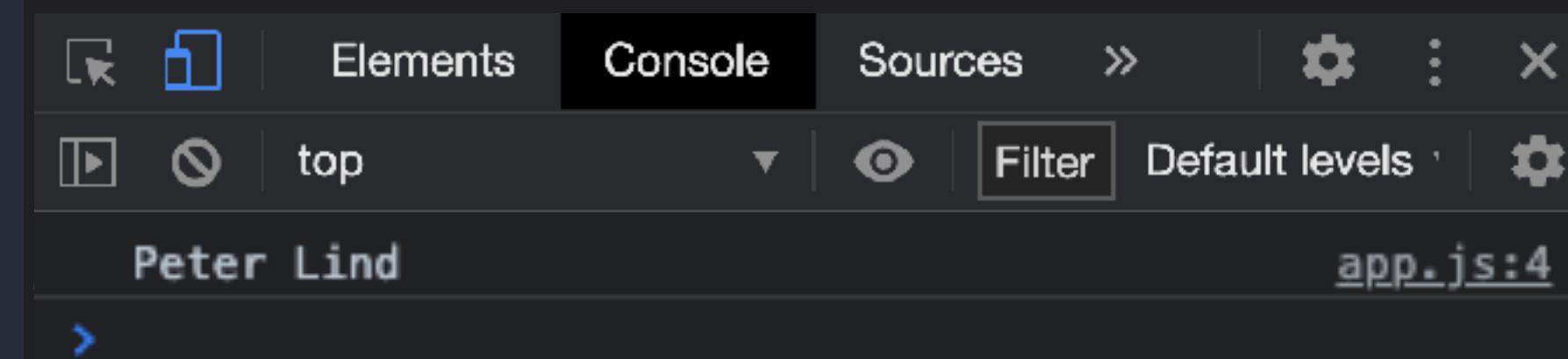
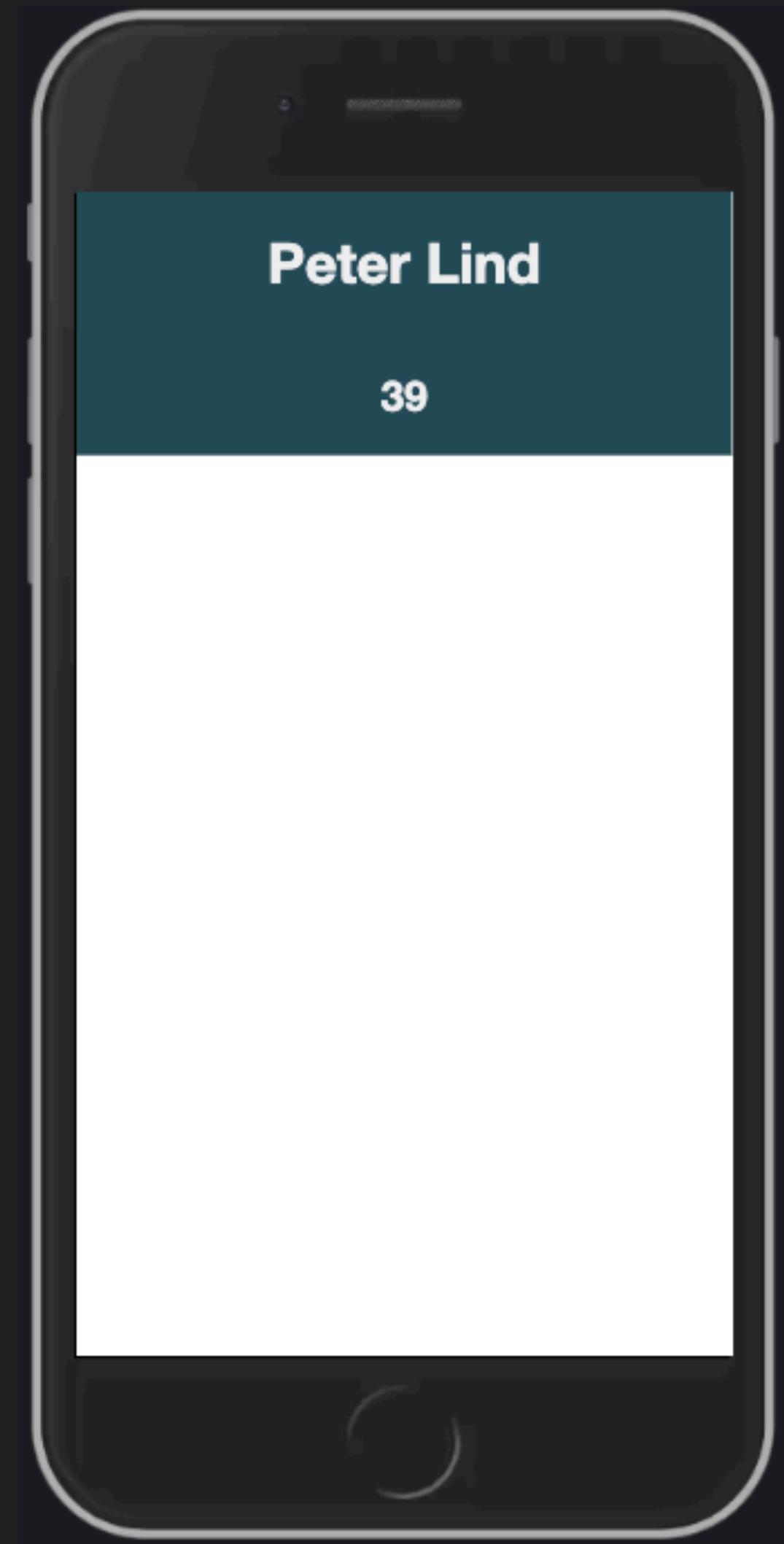
```
let fullName = "Peter Lind";
console.log(fullName);
document.querySelector("#fullName_container").textContent = fullName;
```

## .textContent

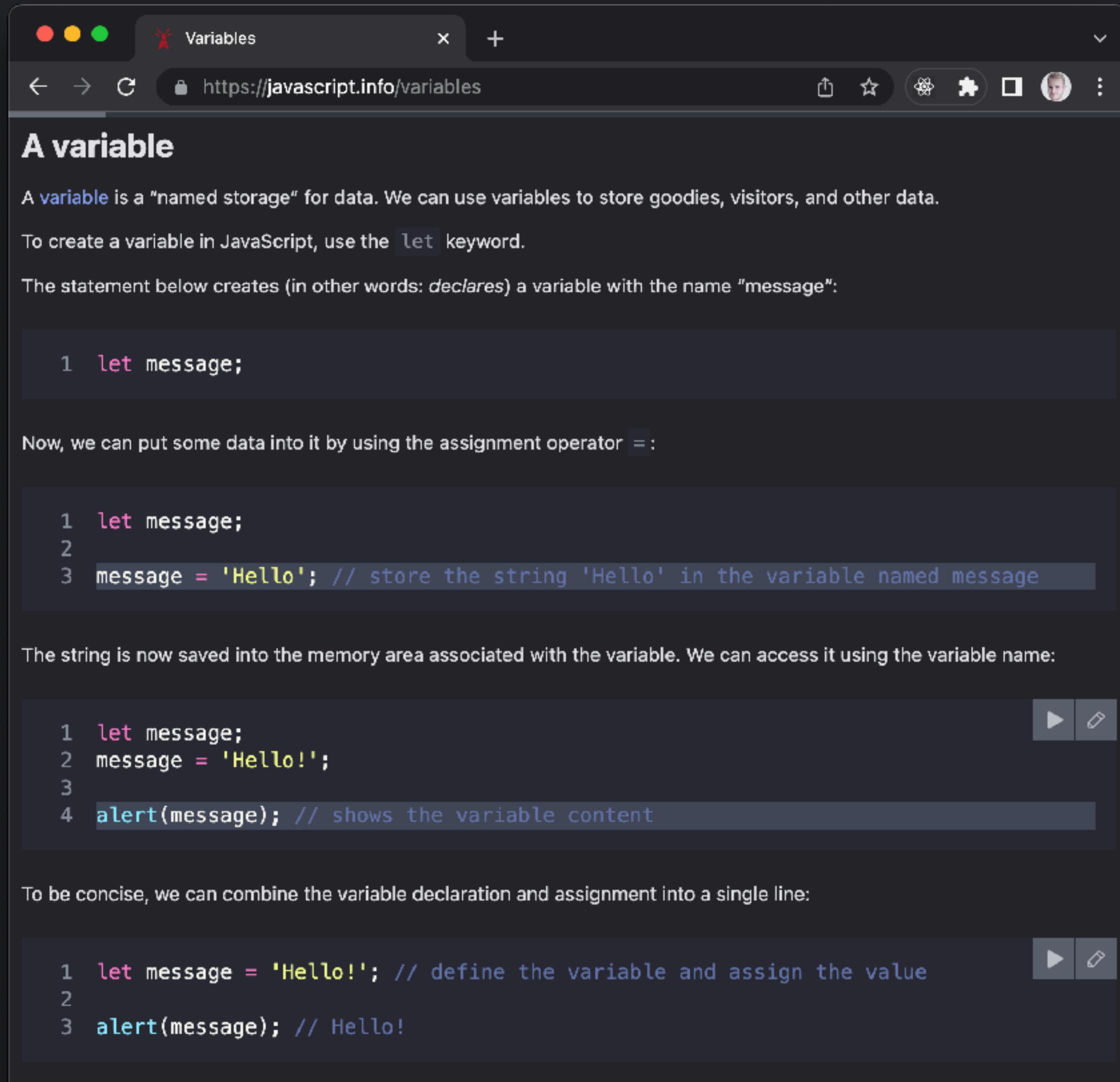
Get or set the text content of a given (HTML) element

# Variables & UI

```
let fullName = "Peter Lind";
let age = 39;
console.log(fullName, age);
document.querySelector("#fullName_container").textContent = fullName;
document.querySelector("#age_container").textContent = age;
```



# JavaScript.info/Variables



A screenshot of a web browser window displaying the [JavaScript.info/Variables](https://javascript.info/variables) page. The browser has a dark theme. The title bar says "Variables". The address bar shows the URL. The page content starts with a section titled "A variable". It explains what a variable is and how to create one using the `let` keyword. It includes a code snippet:

```
1 let message;
```

It then shows how to assign data to the variable using the assignment operator `=`:

```
1 let message;
2
3 message = 'Hello'; // store the string 'Hello' in the variable named message
```

The explanatory text states: "The string is now saved into the memory area associated with the variable. We can access it using the variable name:"

```
1 let message;
2 message = 'Hello!';
3
4 alert(message); // shows the variable content
```

To be concise, we can combine the variable declaration and assignment into a single line:

```
1 let message = 'Hello!'; // define the variable and assign the value
2
3 alert(message); // Hello!
```

Read, read,  
read,  
The docs



# var vs let

THE DIFFERENCE IS THE SCOPING

VAR IS FUNCTION-WIDE OR GLOBAL SCOPE

LET IS BLOCK SCOPED

VAR TOLERATES REDECLARATION

<https://javascript.info/variables>

<https://javascript.info/var>

```
// Example 1
// "var" has no block scope
if (true) {
| var test1 = true; // use "var" instead of "let"
}
console.log(test1); // true, the variable lives after if

// Example 2
if (true) {
| let test2 = true; // use "let"
}
console.log(test2); // Error: test is not defined

// Example 3
for (var i = 0; i < 10; i++) {
| // ...
}
console.log(i); // 10, "i" is visible after loop, it's a global variable
```

```
// "var" tolerates redeclarations
var user1 = "Pete";
var user1 = "John"; // this "var" does nothing (already declared)
// ...it doesn't trigger an error
console.log(user1); // John

let user2;
let user2; // SyntaxError: 'user' has already been declared
```

var-vs-let

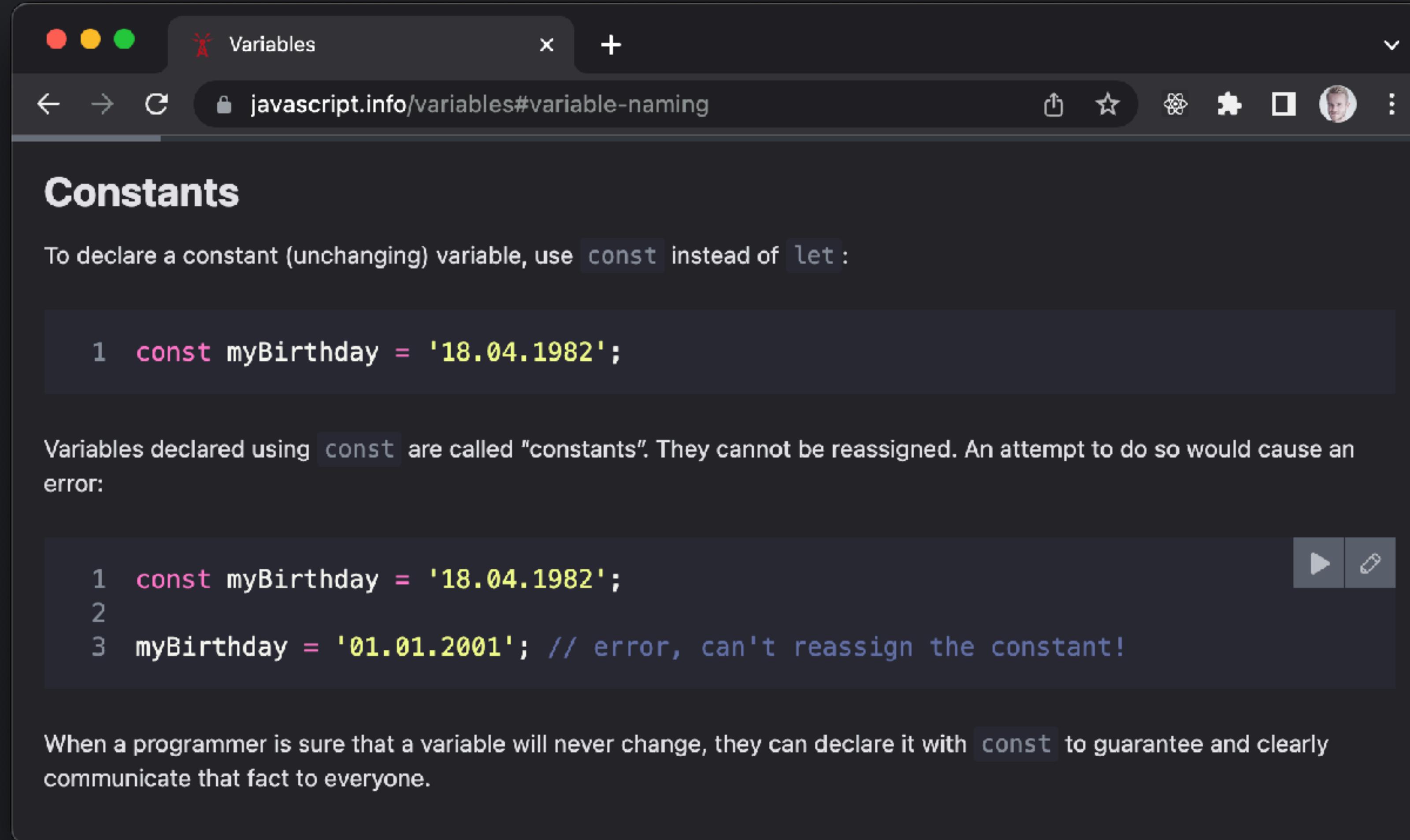
# Const

Const is an unchanging variable.

```
const myBirthday = "12-03-1990";
myBirthday = "12-03-1989";
// Uncaught TypeError: can't reassign the constant!
```

const cannot be reassigned.  
If you try to, an error will be thrown.

# And the doc says...



The screenshot shows a dark-themed web browser window with the title bar "Variables". The address bar contains the URL "javascript.info/variables#variable-naming". The main content area displays the following text and code examples:

## Constants

To declare a constant (unchanging) variable, use `const` instead of `let`:

```
1 const myBirthday = '18.04.1982';
```

Variables declared using `const` are called "constants". They cannot be reassigned. An attempt to do so would cause an error:

```
1 const myBirthday = '18.04.1982';
2
3 myBirthday = '01.01.2001'; // error, can't reassign the constant!
```

When a programmer is sure that a variable will never change, they can declare it with `const` to guarantee and clearly communicate that fact to everyone.

# Const can't be reassigned

```
const myBirthday = "12-03-1990";
myBirthday = "12-03-1989"; // Uncaught TypeError: can't reassign the constant!

const person = {
    name: "Kasper",
    mail: "kato@eaaa.dk",
    age: 32
};

person.age = 33; // no error

person = {
    name: "Rasmus",
    mail: "race@eaaa.dk",
    age: 31
}; // Uncaught TypeError: can't reassign the constant!
```

Use let & const  
instead of var

<https://javascript.info/variables>  
<https://javascript.info/var>

# But which one?

<https://javascript.info/variables>  
<https://javascript.info/var>

Start with const

If needed, change to let

Never use var!

<https://javascript.info/variables>

<https://javascript.info/var>

The screenshot shows a web browser window with the following details:

- Title Bar:** Variables
- Address Bar:** javascript.info/variables#name-things-right
- Left Sidebar (Chapter):**
  - Chapter
  - JavaScript Fundamentals
  - Lesson navigation
  - A variable
  - A real-life analogy
  - Variable naming
  - Constants
  - Name things right
  - Summary
  - Tasks (3)
  - Comments
  - Share
- Right Content Area:**

## Name things right

Talking about variables, there's one more extremely important thing.

A variable name should have a clean, obvious meaning, describing the data that it stores.

Variable naming is one of the most important and complex skills in programming. A quick glance at variable names can reveal which code was written by a beginner versus an experienced developer.

In a real project, most of the time is spent modifying and extending an existing code base rather than writing something completely separate from scratch. When we return to some code after doing something else for a while, it's much easier to find information that is well-labeled. Or, in other words, when the variables have good names.

Please spend time thinking about the right name for a variable before declaring it. Doing so will repay you handsomely.

Some good-to-follow rules are:

  - Use human-readable names like `userName` or `shoppingCart`.
  - Stay away from abbreviations or short names like `a`, `b`, `c`, unless you really know what you're doing.
  - Make names maximally descriptive and concise. Examples of bad names are `data` and `value`. Such names say nothing. It's only okay to use them if the context of the code makes it exceptionally obvious which data or value the variable is referencing.
  - Agree on terms within your team and in your own mind. If a site visitor is called a "user" then we should name related variables `currentUser` or `newUser` instead of `currentVisitor` or `newManInTown`.

Sounds simple? Indeed it is, but creating descriptive and concise variable names in practice is not. Go for it.

<https://javascript.info/variables#name-things-right>

The screenshot shows a web browser window with a dark theme. The title bar says "Variables". The address bar shows the URL "javascript.info/variables". The main content area has a dark background and contains the following text:

## Variable naming

There are two limitations on variable names in JavaScript:

1. The name must contain only letters, digits, or the symbols `$` and `_`.
2. The first character must not be a digit.

Examples of valid names:

```
1 let userName;
2 let test123;
```

When the name contains multiple words, `camelCase` is commonly used. That is: words go one after another, each word except first starting with a capital letter: `myVeryLongName`.

What's interesting – the dollar sign `'$'` and the underscore `'_'` can also be used in names. They are regular symbols, just like letters, without any special meaning.

These names are valid:

```
1 let $ = 1; // declared a variable with the name "$"
2 let _ = 2; // and now a variable with the name "_"
3
4 alert($ + _); // 3
```

Examples of incorrect variable names:

```
1 let 1a; // cannot start with a digit
2
3 let my-name; // hyphens '-' aren't allowed in the name
```

<https://javascript.info/variables#variable-naming>

# Name things right (or wrong?)

1. Create a variable with the name of our planet. How would you name such a variable?
2. Create a variable to store the name of a current visitor to a website. How would you name that variable?

<https://javascript.info/variables>

# Global Variables

Variables outside a function (and scopes) are global variables

# [JavaScript.info/function-basics#local-variables](https://JavaScript.info/function-basics#local-variables)

## Local variables

A variable declared inside a function is only visible inside that function.

For example:

```
1 function showMessage() {  
2   let message = "Hello, I'm JavaScript!"; // local variable  
3  
4   alert( message );  
5 }  
6  
7 showMessage(); // Hello, I'm JavaScript!  
8  
9 alert( message ); // <-- Error! The variable is local to the function
```

## Scopes:

- Local Variable
- Global Variable

## Outer variables

A function can access an outer variable as well, for example:

```
1 let userName = 'John';  
2  
3 function showMessage() {  
4   let message = 'Hello, ' + userName;  
5   alert(message);  
6 }  
7  
8 showMessage(); // Hello, John
```

## Global variables

Variables declared outside of any function, such as the outer `userName` in the code above, are called *global*.

Global variables are visible from any function (unless shadowed by locals).

It's a good practice to minimize the use of global variables. Modern code has few or no globals. Most variables reside in their functions. Sometimes though, they can be useful to store project-level data.

# Datatyper

Application Programming Interface

# Data Types

In JavaScript there are two main data types:

- **Primitive values** like strings, numbers and booleans.
- **Objects** with properties.

```
1 let str = "Hello";
2 let str2 = 'Single quotes are ok too';
3 let phrase = `can embed another ${str}`;
```

```
1 let n = 123;
2 n = 12.345;
```

```
1 let user = new Object(); // "object constructor" syntax
2 let user = {}; // "object literal" syntax
```

In JavaScript, a value always has a certain type like a string, number, boolean, object, array, etc.

# Data Types

Variables can hold 7  
different kinds of values  
(data types)

- only one at a time ...

**Boolean**

**Number**

**String**

**Object**

**Null**

**Undefined**

**Symbol**

# Datatypes - when to use which one?

Boolean	Values that are true or false – conditions
Number	Numbers! Points, counting, <b>math-operations</b> , measuring ...
String	Text! Input-fields – HTML and CSS-attributes ... URLs (Phone-number)
Object	Everything (Arrays, Functions) Combined/collected data (of the other types)
Null	No datatype yet – no value!
Undefined	Something that hasn't defined before – <u>don't set things to undefined!</u>
Symbol	Something complicated

# Datatyper

1. Prøv disse og test med  
console.log
2. Hvordan ser vi værdien?
3. Hvordan ser vi typen?
4. Hvordan definerer du,  
hvilken type, du vil  
gemme i en variabel?

```
const bool = true;
const num = 41;
const str = "Peter";
const obj = {
  cats: 2,
  cars: 1
};
const nothing = null;
let undf;
const symbol = Symbol("symbol");
```

# Strings

In JavaScript textual data is stored as strings.

Strings are defined with either ‘single quotes’, “double quotes” or `backticks`.

```
let single = 'single-quoted';
```

```
let double = "double-quoted";
```

```
let backticks = `backticks`;
```

```
let fullName = "Peter Lind";
let age = 39;

let message = fullName + " is " + age + " years old.";
console.log(message); // Peter Lind is 39 years old.
```

## Concatenation

We can combine two or more variables in one string (variable).

```
let fullName = "Peter Lind";
let age = 39;

// single
console.log(fullName + ' is ' + age + ' years old.');
// double
console.log(fullName + " is " + age + " years old.");
// backticks
console.log(` ${fullName} is ${age} years old.`);
```

## Concatenation

Strings are defined with either ‘single quotes’, “double quotes” or `backticks`.

# String quotes

1. Test the code to the right.
2. What is the output of the script to the right?

```
"use strict";  
  
let fullName = "Alicia Keys";  
  
console.log(`hello ${1}`); // ?  
console.log(`hello ${"name"} `); // ?  
console.log(`hello ${fullName}`); // ?
```

# Data Type Conversion

In JavaScript, we never specify the type of a variable (String, Number, Boolean, Object, etc.).

We set the value and use the value as the type we expect it to be.

```
const firstName = "Rasmus";
const lastName = "Cederdorff";

let age = 32;

let isSeniorLecture = true;

const obj = {
    cats: 0,
    cars: 1
};

const kids = ["Alicia", "Ida"];
```

It happens when we do:

- Concatenation ( + )
- Comparisons ( ==, !=, <, >, <=, >= )
- Calculations ( -, \*, /, % )

## Automatic conversion

So often JavaScript has to do automatic conversion.

Everything in HTML is strings, so when writing or reading numbers, they have to be converted.

Most of the time this happens automatically.

# “Semi-automatic” conversion

`String( value)` converts a value to a string, usually by calling `value.toString()`

`Number( value)` converts a value to a number, usually by calling `value.valueOf()`

`Boolean( value)` converts a value to a boolean, by unknown magic means!

Note:

`Boolean(0)` is false, but

`Boolean("0")` is true

It is always possible to convert something into a string –  
but it might not be possible to convert everything into a number!

# Numeric conversion

1. How do you convert to Number?

```
let string = "1234";  
console.log(string);  
// number??
```

# String conversion

1. How do you convert to String?

```
let number = 1234;  
console.log(number);  
// string??
```

# Numeric conversion 2

1. Try yourself!

2. What is the output in  
the console?

```
console.log(Number(" 123  ")); // ?  
console.log(Number("123z")); // ?  
console.log(Number(true)); // ?  
console.log(Number(false)); // ?
```

Type Conversions

← → C 🔒 javascript.info/type-conversions

EN  Buy EPUB/PDF  ☰ ⚡ 🔍

Chapter

JavaScript Fundamentals

Lesson navigation

String Conversion

Numeric Conversion

Boolean Conversion

Summary

Comments

Share  

Edit on GitHub

Type Conversions

The JavaScript language → JavaScript Fundamentals January 24, 2023

## Type Conversions

Most of the time, operators and functions automatically convert the values given to them to the right type.

For example, `alert` automatically converts any value to a string to show it. Mathematical operations convert values to numbers.

There are also cases when we need to explicitly convert a value to the expected type.

 Not talking about objects yet

In this chapter, we won't cover objects. For now, we'll just be talking about primitives.

Later, after we learn about objects, in the chapter [Object to primitive conversion](#) we'll see how objects fit in.

## String Conversion

String conversion happens when we need the string form of a value.

For example, `alert(value)` does it to show the value.

We can also call the `String(value)` function to convert a value to a string:

```
1 let value = true;
2 alert(typeof value); // boolean
3
4 value = String(value); // now value is a string "true"
5 alert(typeof value); // string
```

String conversion is mostly obvious. A `false` becomes `"false"`, `null` becomes `"null"`, etc.

## Numeric Conversion

Numeric conversion in mathematical functions and expressions happens automatically.

For example, when divisor `/` is applied to non-numbers:

# Read, read, read, The docs



# Manual conversion

You can omit  
the prefix  
`Number.` if you  
want

`Number.parseInt( string )`

Converts a string **to a number**, using the specified numerical system ([MDN](#))

`Number.prototype.toString( )`

Converts a number **to a string**, in the specified numerical system ([MDN](#))

*There's also: `parseFloat`, `toExponential`, `toFixed`, `toPrecision` for other kinds of conversion*

```
const n1 = 5;  
const n2 = 6;
```

```
const s1 = "4";  
const s2 = "9";
```

```
console.log(n1 + n2);  
console.log(s1 + s2);
```

```
console.log(n1 - n2);  
console.log(s1 - s2);
```

```
console.log(n1 + s2);  
console.log(s1 + n2);
```

```
console.log(n1 - s2);  
console.log(s1 - n2);
```

# Concatenation looks like addition

1. What will be the result of these calculations/concatenations?
2. Test them in the console!

# Automatic data type conversion with concatenation

String + string is a **concatenation**

Number + number is a **calculation**

But if the types are different, one gets converted to the other!

The challenge is to figure out which one gets converted to what ...

```
"" + 1 + 0;  
"" - 1 + 0;  
true + false;  
6 / "3";  
"2" * "3";  
4 + 5 + "px";  
"$" + 4 + 5;  
"4" - 2;  
"4px" - 2;  
" -9 " + 5;  
" -9 " - 5;  
null + 1;  
undefined + 1;  
" \t \n" - 2;
```

# Type Conversion

1. Test these expressions.
2. First: Try to guess the result, then check your guess in the console!
3. Explain to each other why your guess was correct/wrong.

# Template String

A template string (template literal) in JavaScript is a string enclosed in backticks (`) that allows embedding variables and expressions using \${ ... } and supports multi-line text.

# `template string`

*“Template literals are literals delimited with backtick (`) characters, allowing for multi-line strings, for string interpolation with embedded expressions, and for special constructs called tagged templates.”*

```
let name = "Alicia";
let age = 6;
```

```
console.log(name + " is " + age + " years old.");
```

```
console.log(` ${name} is ${age} years old. `);
```

Alicia is 6 years old.	<a href="#">main.js:10</a>
Alicia is 6 years old.	<a href="#">main.js:12</a>

# `template string`

A *template string (template literal)* in JavaScript is a string enclosed in backticks () that allows embedding variables and expressions using \${...} and supports multi-line text.

```
const navn = "Rasmus";
const alder = 34;

const tekst = `Hej, jeg hedder ${navn} og jeg er ${alder} år gammel.`;

console.log(tekst);
// Output: Hej, jeg hedder Rasmus og jeg er 34 år gammel.
```

```
const navn = "Rasmus";
const alder = 34;

const tekst = `Hej, jeg hedder ${navn} og jeg er ${alder} år gammel.`;
const tekstUdenTemplate = "Hej, jeg hedder " + navn + " og jeg er " + alder + " år gammel.";

console.log(tekst);
// Output: Hej, jeg hedder Rasmus og jeg er 34 år gammel.
```

# `template string`

## Backtick String / Template Literals

- Extended functionality
- Simplifies concatenating strings
- Embed values and expression into a string with \${...}
- Simplifies the syntax and the reading
- Let us create more readable HTML templates

```
let name = "Alicia";
console.log(`Hello, ${name}`);
```

Hello, Alicia

main.js:8

# `template string`

```
let name = "Alicia";  
let age = 6;
```

```
console.log(name + " is " + age + " years old.");
```

```
console.log(` ${name} is ${age} years old. `);
```

Alicia is 6 years old.  
Alicia is 6 years old.

[main.js:10](#)  
[main.js:12](#)

# `template string`

## REGULAR STRING EXPRESSION

```
function appendTeachers(teachers) {  
  for (let teacher of teachers) {  
    console.log(teacher);  
    document.querySelector("#grid-teachers").innerHTML +=  
      "<article>" +  
      "<img src='" + teacher.img + "'>" +  
      "<h3>" + teacher.name + "</h3>" +  
      teacher.position + "<br>" +  
      "<a href='mailto:" + teacher.mail + "'>" + teacher.mail + "</a>" +  
      "</article>";  
  }  
}
```

### TEACHERS



Birgitte Kirk Iversen

Senior Lecturer  
[bki@baaa.dk](mailto:bki@baaa.dk)



Michael Hvidtfeldt

Senior Lecturer  
[mhv@baaa.dk](mailto:mhv@baaa.dk)



Rasmus Cederdorff

Lecturer  
[race@baaa.dk](mailto:race@baaa.dk)

# `template string`

... EMBED VARIABLES AND EXPRESSIONS IN A STRING

```
function appendTeachers(teachers) {  
  for (let teacher of teachers) {  
    console.log(teacher);  
    document.querySelector("#grid-teachers").innerHTML +=  
      "<article>" +  
      "<img src='" + teacher.img + "'>" +  
      "<h3>" + teacher.name + "</h3>" +  
      teacher.position + "<br>" +  
      "<a href='mailto:" + teacher.mail + "'>" + teacher.mail + "</a>" +  
      "</article>";  
  }  
}
```



```
function appendTeachers(teachers) {  
  for (let teacher of teachers) {  
    console.log(teacher);  
    document.querySelector("#grid-teachers").innerHTML += `  
      <article>  
        <img src='${teacher.img}'>  
        <h3>${teacher.name}</h3>  
        ${teacher.position}<br>  
        <a href='mailto:${teacher.mail}'>${teacher.mail}</a>  
      </article>`;  
  }  
}
```

# `VS Code ES6 String HTML`

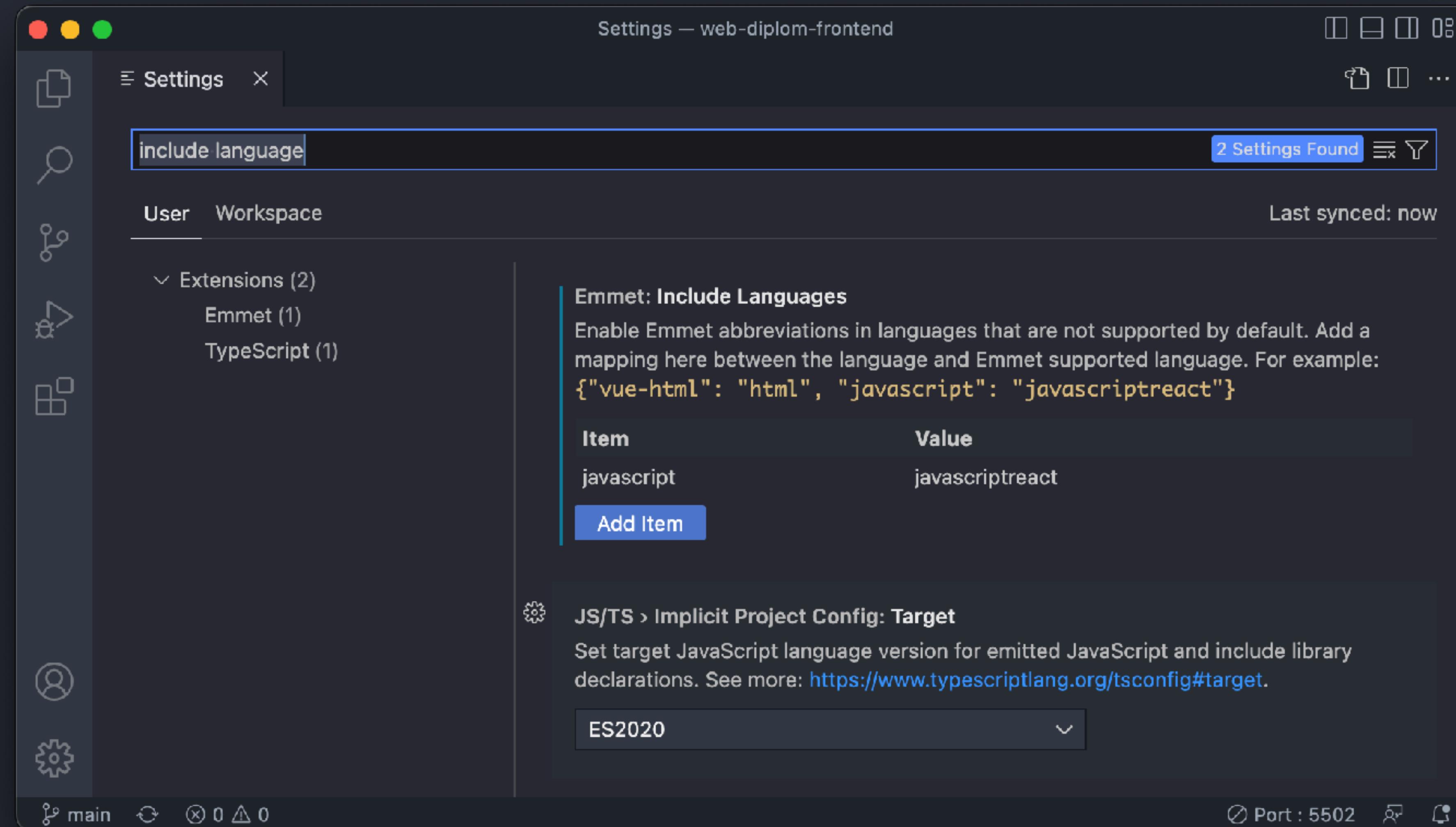
<https://marketplace.visualstudio.com/items?itemName=hjb2012.vscode-es6-string-html>

```
function appendTeachers(teachers) {
  for (let teacher of teachers) {
    console.log(teacher);
    document.querySelector("#grid-teachers").innerHTML += `
      <article>
        <img src='${teacher.img}'>
        <h3>${teacher.name}</h3>
        ${teacher.position}<br>
        <a href='mailto:${teacher.mail}'>${teacher.mail}</a>
      </article>`;
  }
}
```



```
function appendTeachers(teachers) {
  for (let teacher of teachers) {
    console.log(teacher);
    document.querySelector("#grid-teachers").innerHTML += /*html*/
      <article>
        <img src='${teacher.img}'>
        <h3>${teacher.name}</h3>
        ${teacher.position}<br>
        <a href='mailto:${teacher.mail}'>${teacher.mail}</a>
      </article>;
  }
}
```

# Add language support in template string



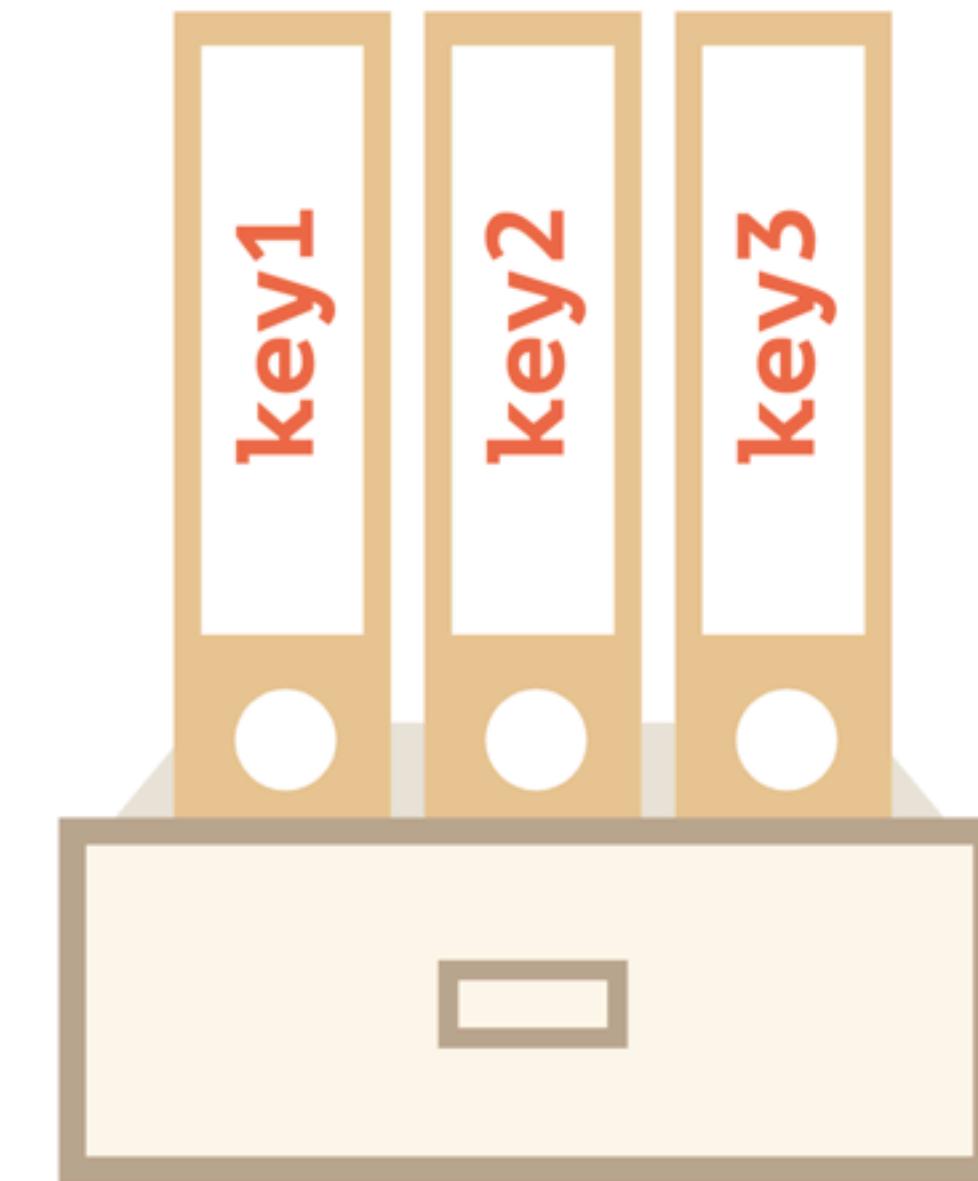
# Objects

A set of named values: `key:value`

# Objects

A set of named values

Objects are used to store keyed  
collections of various data



Containers for named values  
called properties. A property  
is a “key: value” pair

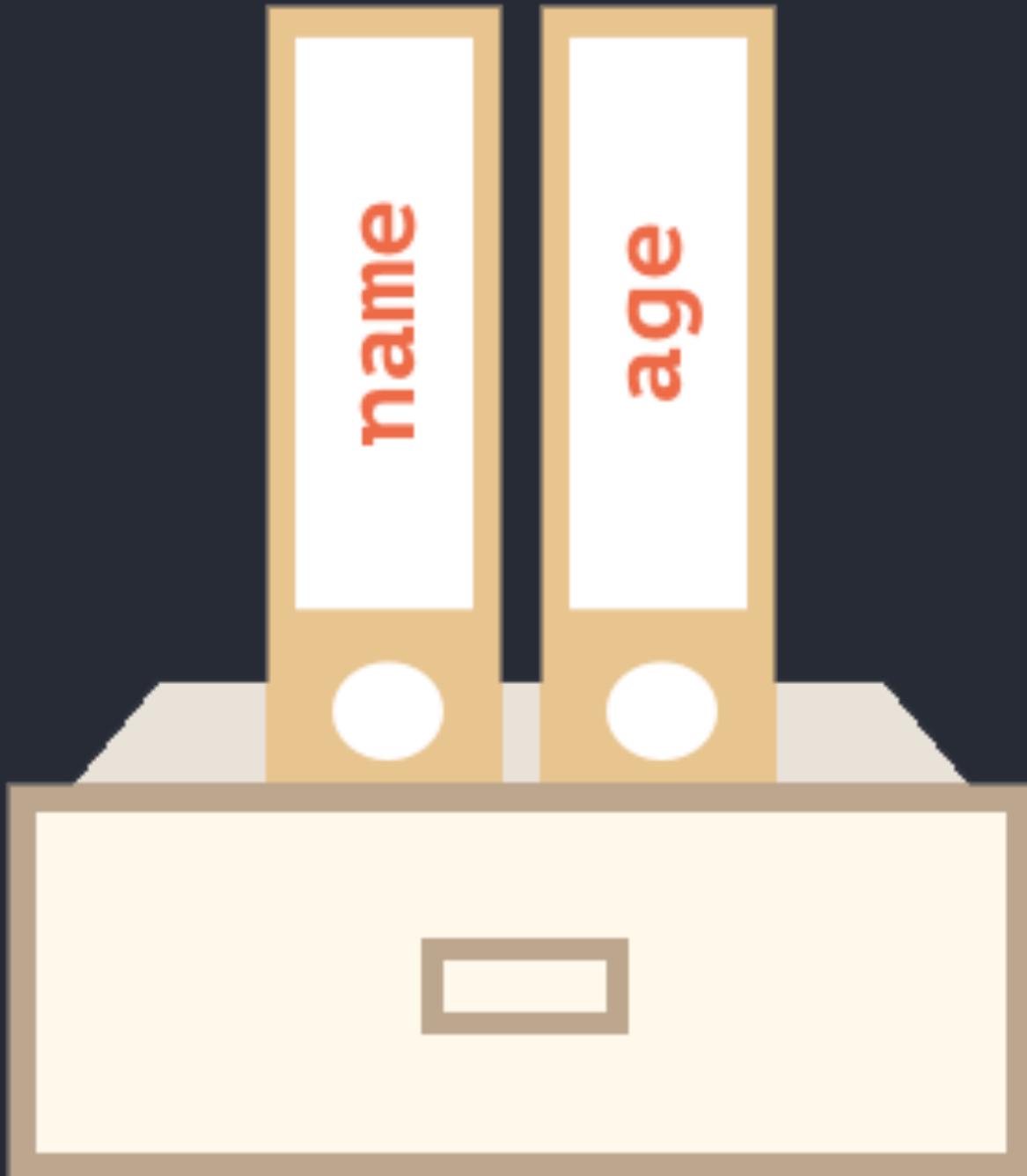
# Objects

A set of named values

```
let user = {  
    name: 'Alicia',  
    age: 6  
};
```

```
console.log(user.name +  
    " is " + user.age +  
    " years old.");
```

user

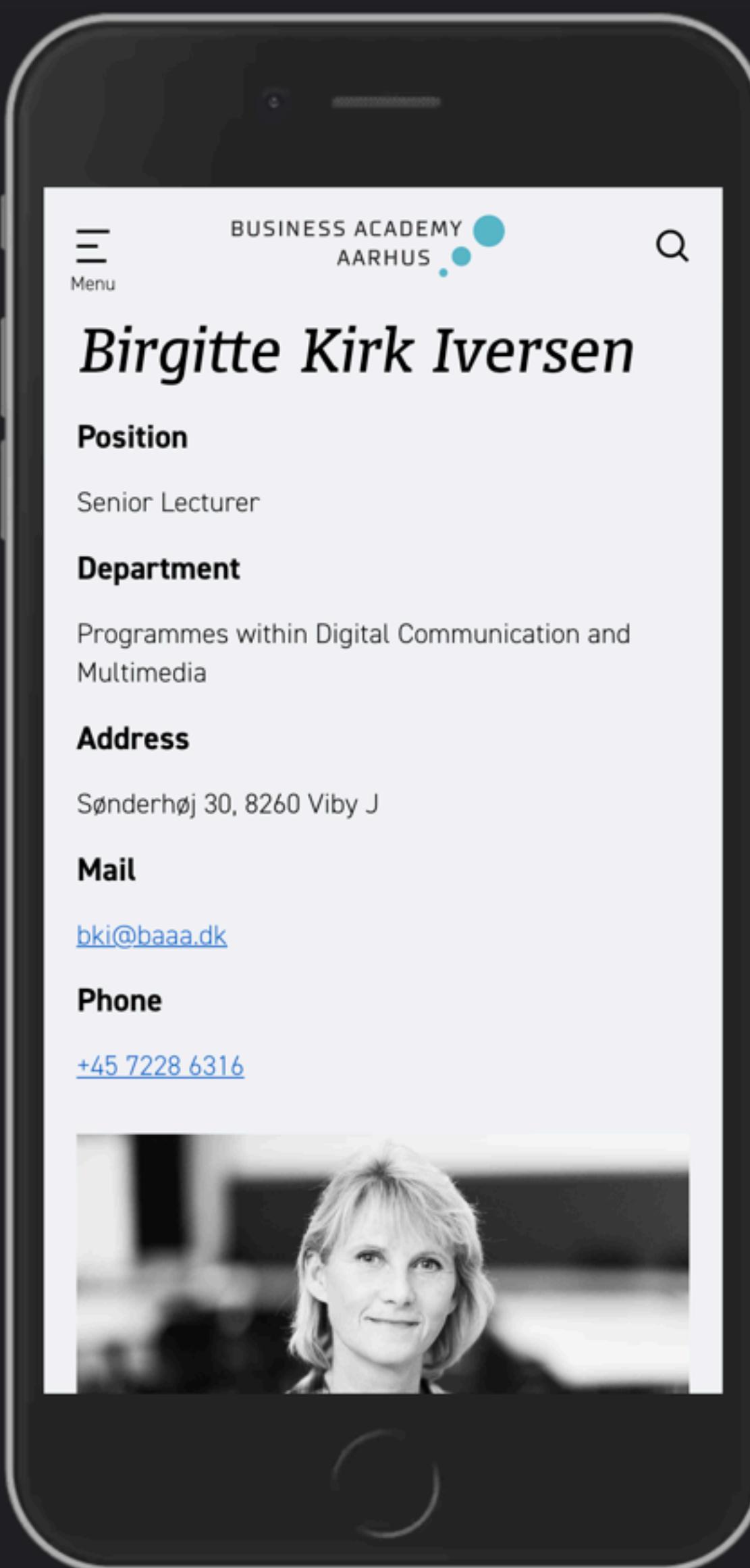


Alicia is 6 years old.

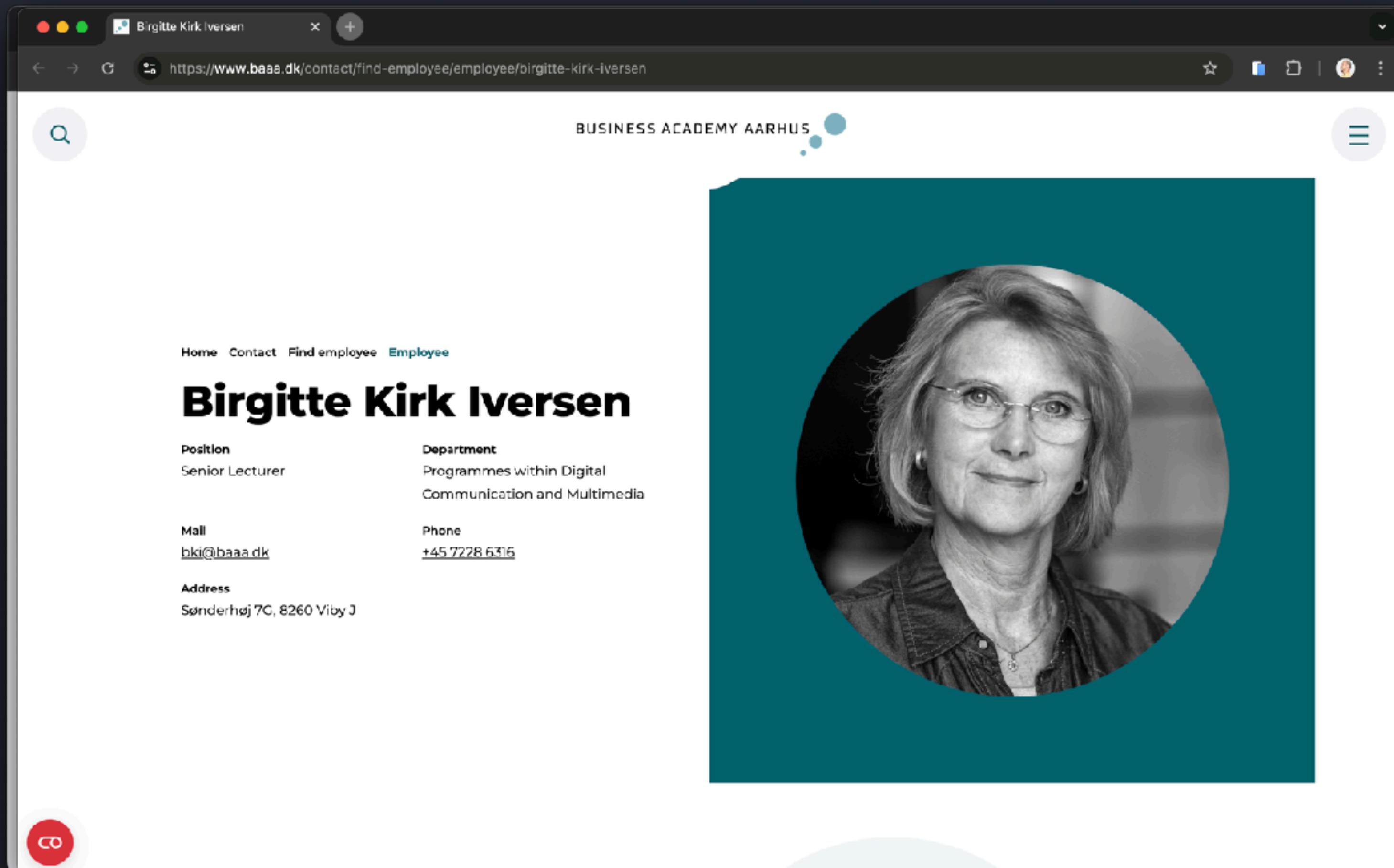
main.js:11

# Objects

# A set of named values



# Objects



<https://www.baaa.dk/contact/find-employee/employee/birgitte-kirk-iversen>

# Objects

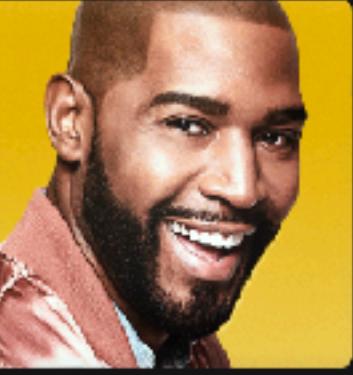
A set of named values

```
key           value
const mrBackend = {
  name: "Kasper Fischer Topp",
  mail: "kato@eaaa.dk",
  phone: "72286328",
  position: "Lecturer",
  favTechnologies: ["PHP", "SQL"]
};
```



NETFLIX

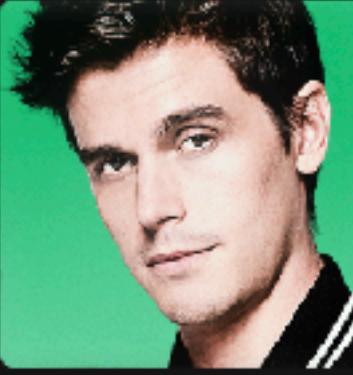
Hvem ser?



Personen der  
rent faktisk  
betaler for  
profilen



Nasser 1



Nasser 2



Nasser 3



Nasser 4 Khader

Administrer profiler

www.netflix.com/browse

NETFLIX Start Serier Film Spil Nyt og populært Min liste Gennemse efter sprog

Too Hot To Handle 3

Mobilspil • Interaktiv historie  
Inkluderet i dit medlemskab

Du er en sexbombe. Crash retræten, og skab drama blandt parrene i denne sæson af det populære dating-spil. Der er saftige overraskelser i vente.

Hent mobilspil Mere info

Actionfilm baseret på bøger

TERNET NINJA 2 THE HUNGER GAMES THE HUNGER GAMES CATCHING FIRE TERNET NINJA JUMANJI: WELCOME TO THE JUNGLE DIVERGENT

Fleste titler til dig

WEDNESDAY KPOP DEMON HUNTERS GINNY & GEORGIA HONEY SQUID GAME DOLLARS DEX

The screenshot shows a dark-themed Netflix interface. On the left, a sidebar for the mobile game 'Too Hot To Handle 3' is displayed, featuring a small thumbnail, the title, genre information ('Mobilspil • Interaktiv historie'), and a note that it's included in the membership. Below this, there's a brief description of the game's premise. At the bottom of the sidebar are two buttons: 'Hent mobilspil' (Get mobile game) and 'Mere info' (More info). The main content area features a large, vibrant image of a man and a woman laughing together, with several small, floating emoji-like icons around them. To the right of the image is a pink heart icon with the number '+15' next to it. Below the main image, there's a banner for 'Actionfilm baseret på bøger' (Action movies based on books), showing thumbnails for 'TERNET NINJA 2', 'THE HUNGER GAMES', 'THE HUNGER GAMES CATCHING FIRE', 'TERNET NINJA', 'JUMANJI: WELCOME TO THE JUNGLE', and 'DIVERGENT'. At the bottom, there's a section titled 'Fleste titler til dig' (Most titles for you) displaying thumbnails for 'WEDNESDAY', 'KPOP DEMON HUNTERS', 'GINNY & GEORGIA', 'HONEY', 'SQUID GAME', and 'DOLLARS'.

Frost II (2019) - IMDb

imdb.com/title/tt4520988/

IMDb Menu All Search IMDb

# Frost II

Original title: Frozen II  
2019 · 7 · 1h 43m

IMDb RATING YOUR RATING POPULARITY  
★ 6.8/10 160K ★ Rate 896 ▲ 102

Cast & crew · User reviews · Trivia · IMDbPro

All topics

Play trailer 0:16

55 VIDEOS

99+ PHOTOS

Animation Adventure Comedy

+ Add to Watchlist

Anna, Elsa, Kristoff, Olaf and Sven leave Arendelle to travel to an ancient, autumn-bound forest of an enchanted land. They set out to find the origin of Elsa's powers in order to save their kingdom.

1.4K User reviews 289 Critic reviews 64 Metascore

Directors Chris Buck · Jennifer Lee

Writers

```
let movie = {  
  title: "Frozen 2",  
  description: "Elsa the Snow Queen has a",  
  trailer: "https://www.youtube.com/embed",  
  length: "1h 43m",  
  year: "2019"  
}
```

# Define yourself as an object

with the following properties

name, age, mail, phone, city, address

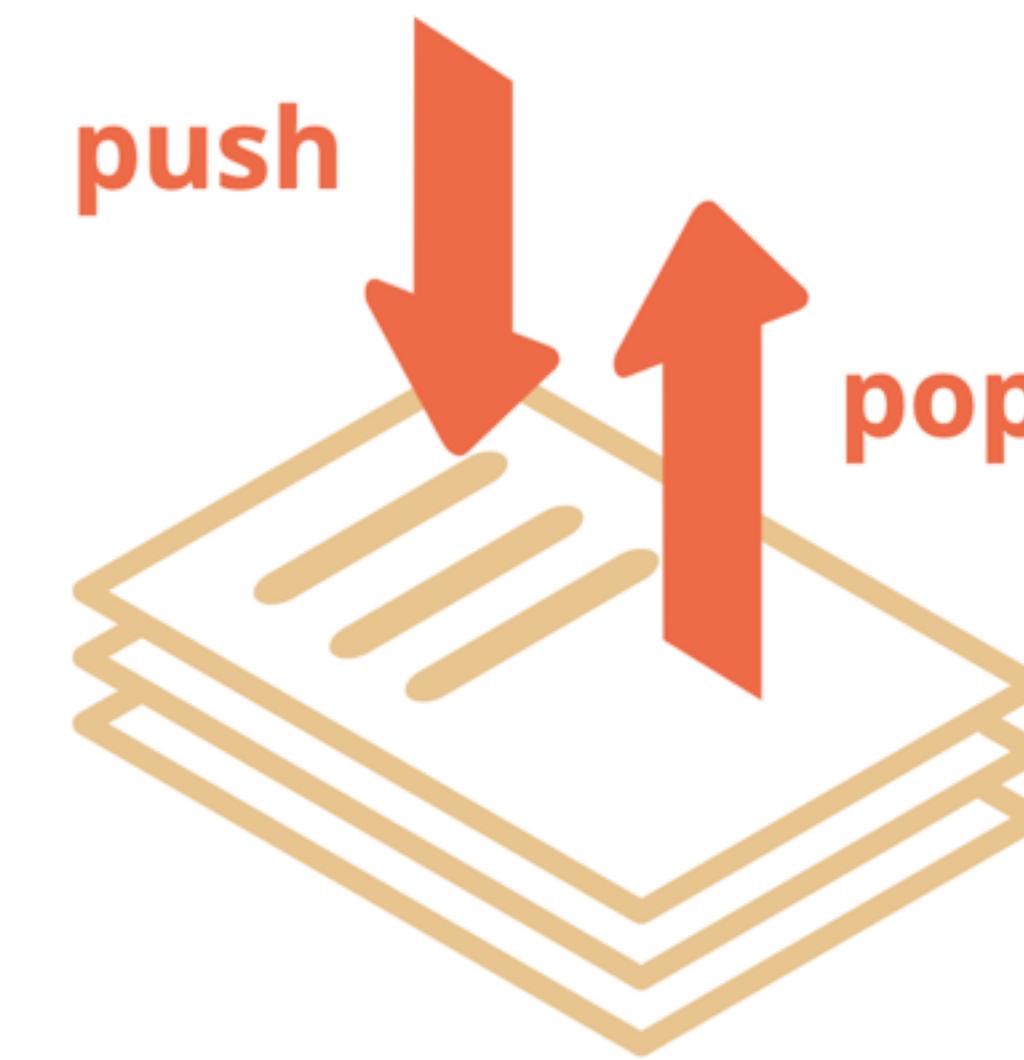
# Arrays

Lists - collection of data

# Arrays

## Collections

Ordered collection of values or  
objects



An array is a way to hold more than one value at a time we have a 1st, a 2nd, a 3rd, a 4th element and so on.

# Arrays

## Collections

Ordered collection of values or objects

```
// Only strings (text)
const movieTitles = ["The Matrix", "Inception"];

// Only numbers
const movieYears = [1999, 2010, 2014];

// Only booleans
const erGode = [true, true, false];

// Blandet indhold (fungerer også!)
const blandetListe = ["The Matrix", 1999, true];

console.log("Film navne:", filmNavne);
console.log("Film år:", filmÅr);
console.log("Er gode:", erGode);
console.log("Blandet:", blandetListe);
```

```
const allMovies = [
  {
    id: 1,
    title: "The Matrix",
    year: 1999, First element
    rating: 8.7,
    genre: ["Action", "Sci-Fi"]
  },
  {
    id: 2,
    title: "Inception",
    year: 2010, Second element
    rating: 8.8,
    genre: ["Action", "Thriller"]
  }
];
```

```
console.log("Complete movie database:", allMovies);
console.log("Number of movies:", allMovies.length);
```

```
let todaysLecturers = [
  {
    name: "Kasper Fischer Topp",
    mail: "kato@eaaa.dk",
    phone: "72286328",
    position: "Lecturer",
    favTechnologies: ["PHP", "SQL"],
    nickname: "Mr. Backend"
  },
  {
    name: "Rasmus Cederdorff",
    mail: "race@eaaa.dk",
    phone: "72286318",
    position: "Lecturer",
    favTechnologies: ["JavaScript"],
    nickname: "Mr. Frontend"
  }
];
```

First element

Second element

# Arrays

Rasmus Cederdorff	
<b>Position:</b> Lecturer	<i>Michael Hvidtfeldt</i>
<b>Department/</b> Multimedia De Digital Conce	
<b>Address:</b> Ringvej Syd 10	Lecturer <i>Birgitte Kirk Iversen</i>
<b>Mail:</b> <a href="mailto:race@baaa.dk">race@baaa.dk</a>	<b>Department/</b> Multimedia Di
<b>Phone:</b> 7228 6318	<b>Position:</b> Senior Lecturer
	<b>Address:</b> Ringvej Syd 10
	<b>Department/programme:</b> Multimedia Design
	<b>Address:</b> Sønderhøj 30, 8260 Viby J
	<b>Mail:</b> <a href="mailto:mhv@baaa.dk">mhv@baaa.dk</a>
	<b>Phone:</b> 7228 6328
	<b>Position:</b> Rasmus Cederdorff
	<b>Address:</b> <a href="mailto:bki@baaa.dk">bki@baaa.dk</a>
	<b>Phone:</b> 7228 6316



```
main.js:21
▼ (3) [{} , {} , {} ] ⓘ
▶ 0: {name: "Birgitte Kirk Iversen", mail: "bki@baaa..."}
▶ 1: {name: "Michael Hvidtfeldt", mail: "mhv@baaa.dk..."}
▶ 2: {name: "Rasmus Cederdorff", mail: "race@baaa.dk..."}
  length: 3
▶ __proto__: Array(0)
main.js:22
▶ {name: "Michael Hvidtfeldt", mail: "mhv@baaa.dk"}
main.js:23
```

```
let teachers = [
  name: "Birgitte Kirk Iversen",
  mail: "bki@baaa.dk"
},
{
  name: "Michael Hvidtfeldt",
  mail: "mhv@baaa.dk"
},
{
  name: "Rasmus Cederdorff",
  mail: "race@baaa.dk"
}
];
```

```
console.log(teachers);
console.log(teachers[1]);
console.log(teachers.length);
```

Teachers

http://127.0.0.1:5501/array-teachers/index.html

Console

main.js:46

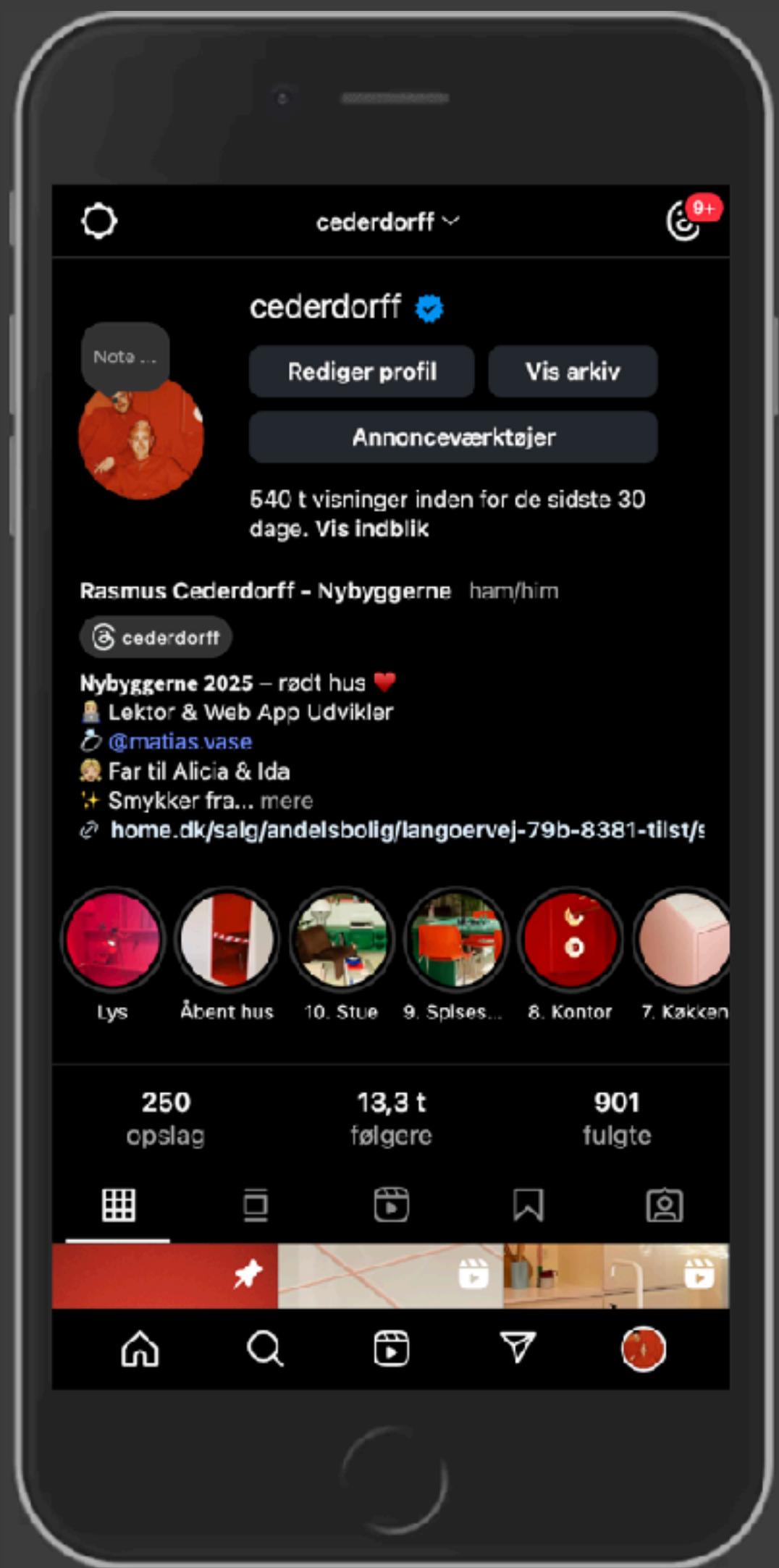
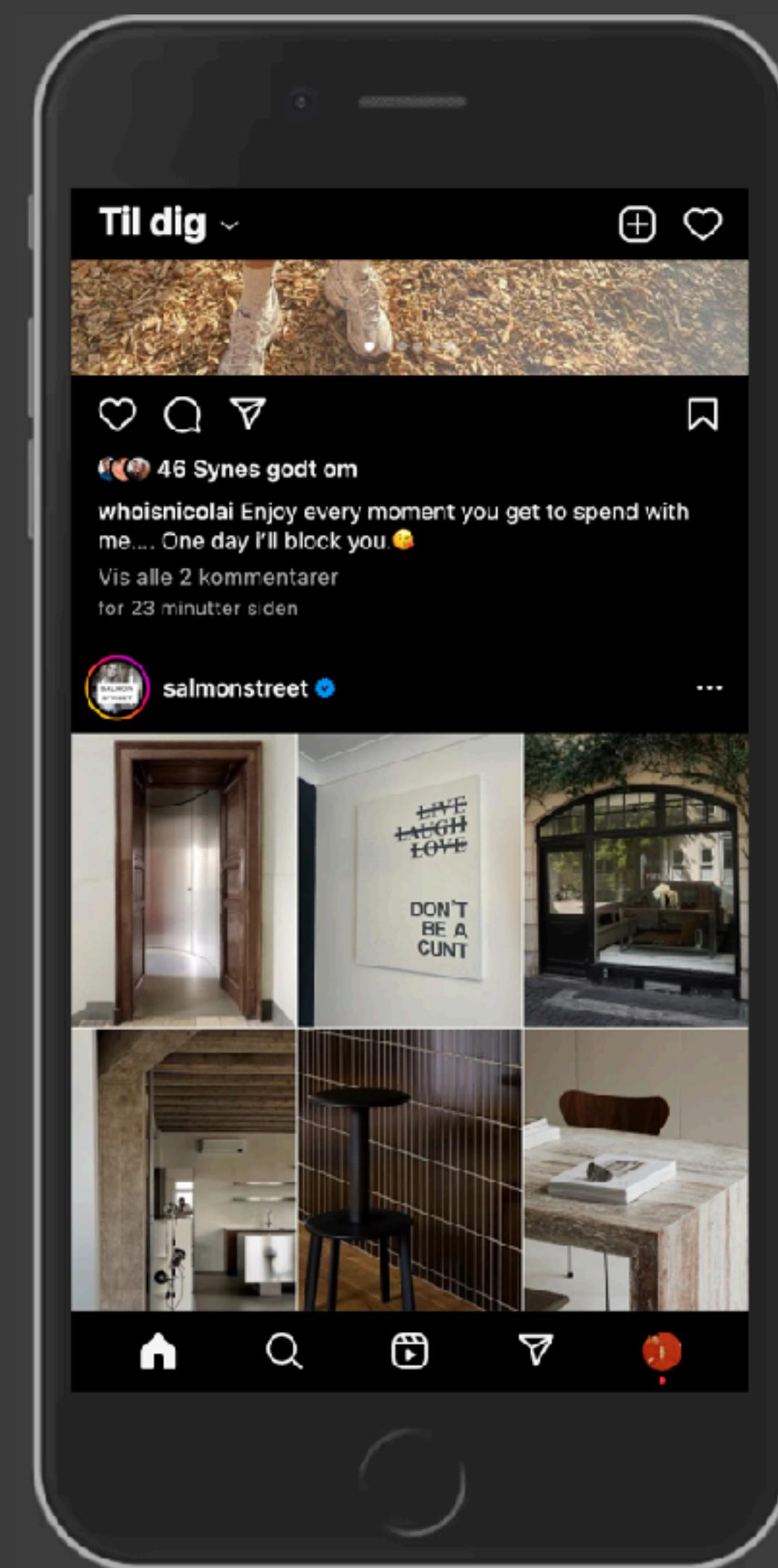
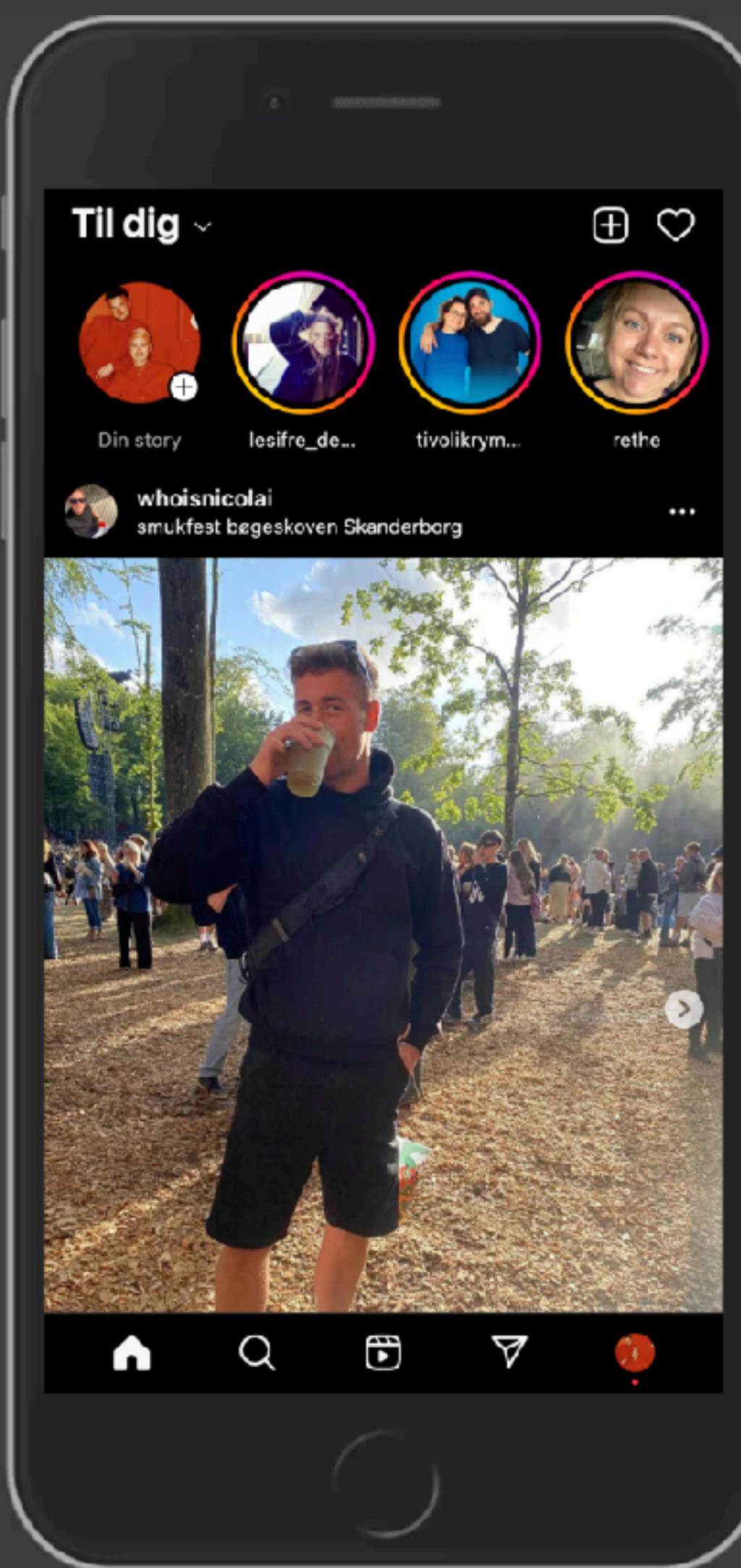
```
▶ (4) [ {...}, {...}, {...}, {...} ] ⓘ
  ▶ 0:
    address: "Sønderhøj 30, 8260 Viby J"
    department: "Multimedia Design"
    img: "https://www.eaaa.dk/media/u4gorzs"
    initials: "bki"
    mail: "bki@baaa.dk"
    name: "Birgitte Kirk Iversen"
    phone: "72286316"
    position: "Senior Lecturer"
    ► [[Prototype]]: Object
  ▶ 1: {name: 'Maria Louise Bendixen', initia...
  ▶ 2: {name: 'Kim Elkjær Marcher-Jepsen', in...
  ▶ 3: {name: 'Rasmus Cederdorff', initials: ...
  length: 4
  ► [[Prototype]]: Array(0)
```

Birgitte Kirk Iversen  
Senior Lecturer  
[bki@baaa.dk](mailto:bki@baaa.dk)

Maria Louise Bendixen  
Senior Lecturer  
[mlbe@baaa.dk](mailto:mlbe@baaa.dk)

Kim Elkjær Marcher-Jepsen  
Lecturer  
[kje@baaa.dk](mailto:kje@baaa.dk)

Rasmus Cederdorff  
Lecturer  
[race@baaa.dk](mailto:race@baaa.dk)



Rasmus Cederdorff - Nybyggerne

https://www.instagram.com/cederdorff/?hl=da

Dimensions: iPhone 6/7/8 ... 414 X 736 10... No throttling

Network

Filter

Preserve log Disable cache No throttling

All Fetch/XHR Doc CSS JS Font Img Media Manifest Socket Wasm Other

Name Headers Payload Preview Response Initiator Timing Cookies

query

query

query

query

query

query

graphql

bz?\_a=1&\_ccg=GOOD...

bulk-route-definitions/

ig\_sso\_users/?hl=da

main

query

query

graphql

graphql/

graphql

cederdorff/?hl=da

graphql

bulk-route-definitions/

bulk-route-definitions/

bz?\_a=1&\_ccg=GOOD...

bulk-route-definitions/

bulk-route-definitions/

bulk-route-definitions/

bulk-route-definitions/

query

bootloader-endpoint/?m...

bz?\_a=1&\_ccg=GOOD...

bz?\_a=1&\_ccg=GOOD...

query

main

sync/?fb\_dtsg\_ag=Ad36i...

49 / 170 requests | 185 kB / 1

6,000 ms 10,000 ms 15,000 ms 20,000 ms 25,000 ms 30,000 ms 35,000 ms 40,000 ms 45,000 ms 50,000 ms 55,000 ms 60,000 ms 65,000 ms 70,000 ms

...

data: {user: {friendship\_status: null, gating: null, is\_memorialized: false, is\_private: false,...},...}

user: {friendship\_status: null, gating: null, is\_memorialized: false, is\_private: false,...}

account\_badges: []

account\_type: 3

address\_street: ""

ai\_agent\_type: null

bio\_links: [{...}, {...}, {...}, {...}, {image\_url: "", is\_pinned: false, link\_type: "external",...}]

biography: "Nybyggerne 2025 – rødt hus ❤️\nLektor & Web App Udvikler\n@matias.vase\nFar til Alicia & Ida\nNybyggerne 2025 – rødt hus ❤️\nLektor & Web App Udvikler\n@matias.vase\nFar til Alicia & Ida\nSmykker fra... mere\nhome.dk/salg/andelsbolig/langoervej-79b-8381-tilst/s...

biography\_with\_entities: {entities: [{hashtag: null, user: {username: "houseofvincent", id: "2016862893"}},...]}

category: "Personal blog"

city\_name: ""

external\_lynx\_url: "https://l.instagram.com/?u=https%3A%2F%2Fhome.dk%2Fsalg%2Fandelsbolig%2Flangoervej-79b-8381-tilst/sag-6400000914/"

external\_url: "https://home.dk/salg/andelsbolig/langoervej-79b-8381-tilst/sag-6400000914/"

fbid\_v2: "17841405578189001"

follower\_count: 13305

following\_count: 901

friendship\_status: null

full\_name: "Rasmus Cederdorff – Nybyggerne"

gating: null

has\_chaining: true

has\_profile\_pic: true

has\_story\_archive: true

hd\_profile\_pic\_url\_info: {...}

hide\_creator\_marketplace\_badge: false

id: "5672009939"

is\_business: false

is\_coppa\_enforced: false

is\_embeds\_disabled: false

is\_memorialized: false

is\_private: false

is\_professional\_account: true

is\_regulated\_c18: false

is\_unpublished: false

is\_verified: true

latest\_besties\_reel\_media: 0

latest\_reel\_media: 0

linked\_fb\_info: {linked\_fb\_page: null, linked\_fb\_user: {name: "Rasmus Cederdorff – Nybyggerne",...}}

live\_broadcast\_id: null

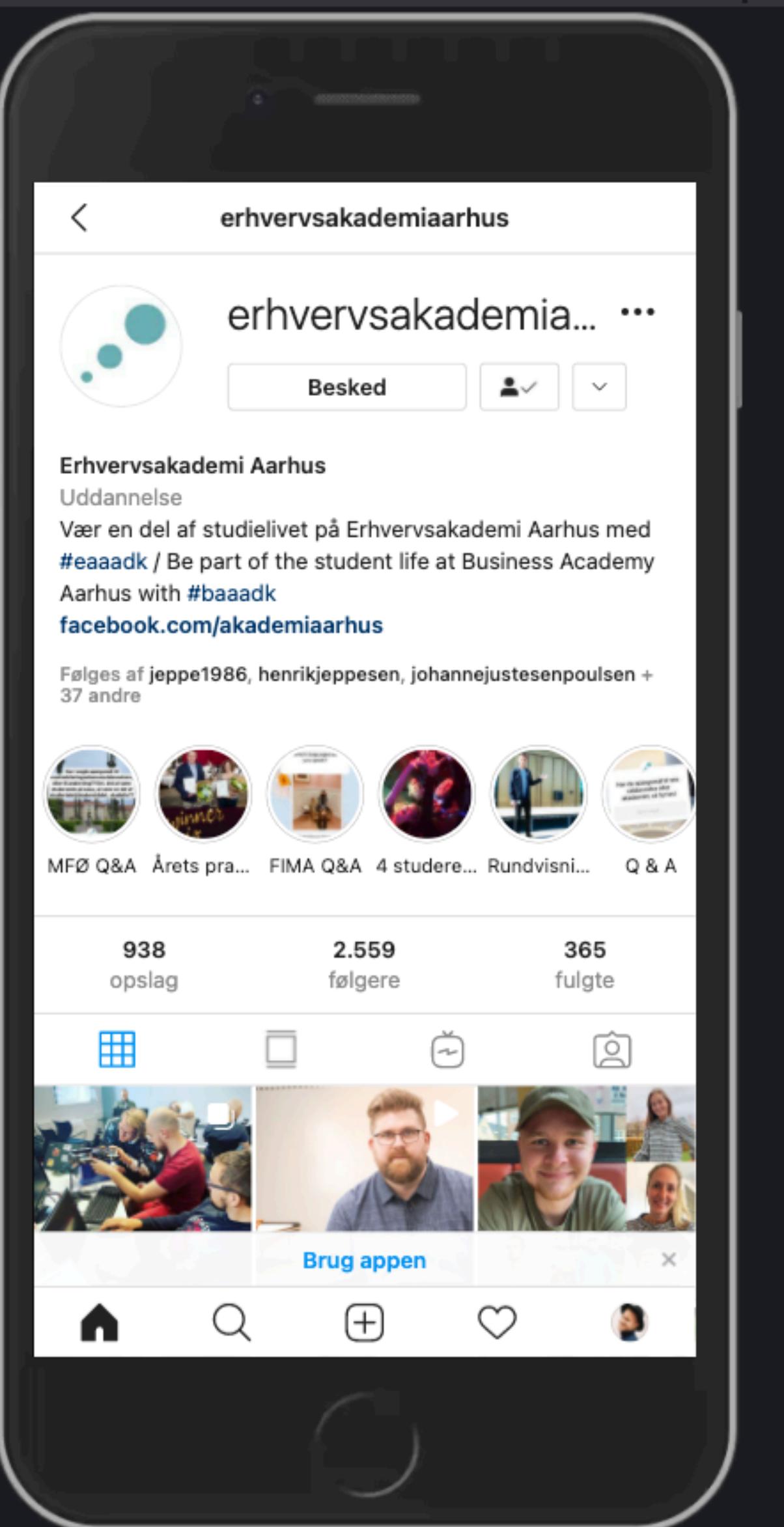
live\_broadcast\_visibility: null

media\_count: 250

mutual\_followers\_count: null

pk: "5672009939"

...



The screenshot shows the Chrome DevTools Network tab. The timeline at the top indicates request times from 5000 ms to 35000 ms. Below the timeline, a list of requests is shown on the left, and a detailed preview of a selected request is on the right.

**Selected Request Preview:**

- Headers:** Headers for the selected request.
- Preview:** Detailed JSON structure of the response body. The response is a nested object with keys: `data`, `status`, `edges`, `node`, `dash_info`, `dimensions`, `display_resources`, `display_url`, `edge_media_preview_comment`, `edge_media_preview_like`, `edge_media_to_caption`, `edge_media_to_sponsor_user`, `edge_media_to_tagged_user`, `fact_check_information`, `fact_check_overall_rating`, and `follow_hashtag_info`. The `node` field contains a `GraphVideo` object with attributes like `__typename`, `id`, `dimensions`, `attribution`, `clips_music_attribution_info`, `coauthor_producers`, `comments_disabled`, `dash_info`, `dimensions`, `display_resources`, `display_url`, `edge_media_preview_comment`, `edge_media_preview_like`, `edge_media_to_caption`, `edge_media_to_sponsor_user`, `edge_media_to_tagged_user`, `fact_check_information`, `fact_check_overall_rating`, and `follow_hashtag_info`.
- Response:** Status code and message.
- Initiator:** The source of the request.
- Timing:** Time taken for the request.
- Cookies:** Cookies associated with the request.

**Request List:**

- reels\_tray/
- ?query\_hash=db...
- ?query\_hash=6ff...
- badge/
- logging\_client\_e...
- bz
- falco
- ?\_a=1
- logging\_client\_e...
- falco
- batch\_fetch\_web/
- ?query\_hash=d4...
- ?query\_hash=8c...
- ?query\_hash=8c...
- logging\_client\_e...
- falco

**Bottom Navigation:** Filter, Fetch/XHR, JS, CSS, Img, Media, Font, Doc, WS, Wasm, Manifest, Other, Has blocked cookies, Blocked Requests.

Rasmus Cederdorff - Nybyggerne

https://www.instagram.com/cederdorff/?hl=da

Dimensions: iPhone 6/7/8 ... 414 X 736 10... No throttling

Network

Fetch/XHR Doc CSS JS Font Img Media Manifest Socket Wasm Other

Name Headers Payload Preview Response Initiator Timing Cookies

query

xdt\_api\_v1\_feed\_user\_timeline\_graphql\_connection: edges: 0: node: code: "DFZnyaxs2X5", pk: "3556048369324615161", id: "3556048369324615161\_603311262", ad\_id: null, accessibility\_caption: null, ad\_id: null, affiliate\_info: null, all\_previous\_submitters: null, audience: null, boost\_unavailable\_identifier: null, boost\_unavailable\_reason: null, boosted\_status: null, can\_reshare: null, can\_see\_insights\_as\_brand: false, can\_viewer\_reshare: true, caption: {has\_translation: true, created\_at: 1738134055, pk: "18021694886382903", ...}, created\_at: 1738134055, has\_translation: true, pk: "18021694886382903", text: "Sidste år hav vi 8 uger ud af vores kalender, pakkede vores kufferter & flyttede ind i en vaskeægte", caption\_is\_edited: false, carousel\_media: null, carousel\_media\_count: null, carousel\_parent\_id: null, clips\_attribution\_info: null, clips\_metadata: {audio\_type: "original\_sounds", achievements\_info: {show\_achievements: false}, music\_info: null}, coauthor\_producers: [{pk: "5672009939", ...}], code: "DFZnyaxs2X5", comment\_count: 113, commenting\_disabled\_for\_viewer: null, comments: null, comments\_disabled: null, crosspost\_metadata: {is\_feedback\_aggregated: null}, display\_uri: null, explore: null, facepile\_top\_likers: [{...}, {...}, {...}], fb\_like\_count: null, feed\_promotion\_control: null, feed\_rec\_promotion\_control: null, follow\_hashtag\_info: null

graphql

bz?\_a=1&\_ccg=GOOD...

bulk-route-definitions/

ig\_sso\_users/?hl=da

main

query

graphql

graphql/

graphql

cederdorff/?hl=da

graphql

bulk-route-definitions/

bulk-route-definitions/

bz?\_a=1&\_ccg=GOOD...

bulk-route-definitions/

bulk-route-definitions/

bulk-route-definitions/

bulk-route-definitions/

query

bootloader-endpoint/?m...

bz?\_a=1&\_ccg=GOOD...

bz?\_a=1&\_ccg=GOOD...

query

main

sync/?fb\_dtsg\_ag=Ad36i...

51 / 172 requests | 185 kB / 1

Course roster: WU-E22a - 1. se

<https://eaaa.instructure.com/courses/15482/users>

WU-E22a > People

60 Student view

Home Announcements Modules Assignments Discussions People BigBlueButton Grades Pages Files Syllabus Outcomes Rubrics Quizzes Collaborations Settings

Everyone Groups + Group set

Search people All roles + People

Name	Login ID	SIS ID	Section	Role	Last Activity	Total Activity
Clara Juul Birk	eaacibi@students.eaaa.dk	WU-E22a - 1.	Student semester	Student	24 Aug at 13:16	01:04:21
Martin Rieper Boesen	eaamrbo@students.eaaa.dk	WU-E22a - 1.	Student semester	Student	24 Aug at 7:54	01:07:06
Dan Okkels Brendstrup	dob@eaaa.dk	WU-E22a - 1.	Teacher semester	Teacher	3 Aug at 8:55	
Rasmus Cederdorff	race@eaaa.dk	WU-E22a - 1.	Teacher semester	Teacher	25 Aug at 9:28	01:19:23
Jeffrey David Serio	jds@eaaa.dk	WU-E22a - 1.	Teacher semester	Teacher	17 Aug at 16:39	
Charlotte Meng Emanuel Dyrholm	eaacmed@students.eaaa.dk	WU-E22a - 1.	Student semester	Student	23 Aug at 16:59	22:24

key value

```
[{"id": "23974", "name": "Clara Juul Birk", "created_at": "2020-08-24T00:46:05+02:00", "email": "dob@eaaa.dk", "short_name": "Clara Juul Birk", "sortable_name": "Birk, Clara Juul", "integration_id": null, "login_id": "dob@eaaa.dk", "name": "Clara Juul Birk", "short_name": "Clara Juul Birk", "sortable_name": "Birk, Clara Juul", "sis_user_id": null}, {"id": "36267", "name": "Martin Rieper Boesen", "created_at": "2021-07-30T00:46:05+02:00", "email": "eaamrbo@students.eaaa.dk", "short_name": "Martin Rieper Boesen", "sortable_name": "Boesen, Martin Rieper", "integration_id": null, "login_id": "eaamrbo@students.eaaa.dk", "name": "Martin Rieper Boesen", "short_name": "Martin Rieper Boesen", "sortable_name": "Boesen, Martin Rieper", "sis_user_id": null}, {"id": "29923", "name": "Dan Okkels Brendstrup", "created_at": "2021-07-30T00:46:05+02:00", "email": "dob@eaaa.dk", "short_name": "Dan Okkels Brendstrup", "sortable_name": "Brendstrup, Dan Okkels", "integration_id": null, "login_id": "dob@eaaa.dk", "name": "Dan Okkels Brendstrup", "short_name": "Dan Okkels Brendstrup", "sortable_name": "Brendstrup, Dan Okkels", "sis_user_id": null}, {"id": "14427", "name": "Rasmus Cederdorff", "created_at": "2021-07-30T00:46:05+02:00", "email": "race@eaaa.dk", "short_name": "Rasmus Cederdorff", "sortable_name": "Cederdorff, Rasmus", "integration_id": null, "login_id": "race@eaaa.dk", "name": "Rasmus Cederdorff", "short_name": "Rasmus Cederdorff", "sortable_name": "Cederdorff, Rasmus", "sis_user_id": null}, {"id": "41", "name": "Jeffrey David Serio", "created_at": "2021-07-30T00:46:05+02:00", "email": "jds@eaaa.dk", "short_name": "Jeffrey David Serio", "sortable_name": "Serio, Jeffrey David", "integration_id": null, "login_id": "jds@eaaa.dk", "name": "Jeffrey David Serio", "short_name": "Jeffrey David Serio", "sortable_name": "Serio, Jeffrey David", "sis_user_id": null}, {"id": "24043", "name": "Charlotte Meng Emanuel Dyrholm", "created_at": "2021-07-30T00:46:05+02:00", "email": "eaacmed@students.eaaa.dk", "short_name": "Charlotte Meng Emanuel Dyrholm", "sortable_name": "Dyrholm, Charlotte Meng Emanuel", "integration_id": null, "login_id": "eaacmed@students.eaaa.dk", "name": "Charlotte Meng Emanuel Dyrholm", "short_name": "Charlotte Meng Emanuel Dyrholm", "sortable_name": "Dyrholm, Charlotte Meng Emanuel", "sis_user_id": null}, {"id": "23978", "name": "Jeppe Frik", "created_at": "2020-08-03T00:46:05+02:00", "email": null, "short_name": "Jeppe Frik", "sortable_name": "Frik, Jeppe", "integration_id": null, "login_id": null, "name": "Jeppe Frik", "short_name": "Jeppe Frik", "sortable_name": "Frik, Jeppe", "sis_user_id": null}, {"id": "23963", "name": "Daniel Tjerrild Gamborg", "created_at": "2020-08-03T00:46:05+02:00", "email": null, "short_name": "Daniel Tjerrild Gamborg", "sortable_name": "Gamborg, Daniel Tjerrild", "integration_id": null, "login_id": null, "name": "Daniel Tjerrild Gamborg", "short_name": "Daniel Tjerrild Gamborg", "sortable_name": "Gamborg, Daniel Tjerrild", "sis_user_id": null}, {"id": "23992", "name": "Casper Hedegaard Hansen", "created_at": "2020-08-03T00:46:05+02:00", "email": null, "short_name": "Casper Hedegaard Hansen", "sortable_name": "Hedegaard Hansen, Casper", "integration_id": null, "login_id": null, "name": "Casper Hedegaard Hansen", "short_name": "Casper Hedegaard Hansen", "sortable_name": "Hedegaard Hansen, Casper", "sis_user_id": null}, {"id": "36266", "name": "Morten Gedsted Hansen", "created_at": "2020-08-03T00:46:05+02:00", "email": null, "short_name": "Morten Gedsted Hansen", "sortable_name": "Hansen, Morten Gedsted", "integration_id": null, "login_id": null, "name": "Morten Gedsted Hansen", "short_name": "Morten Gedsted Hansen", "sortable_name": "Hansen, Morten Gedsted", "sis_user_id": null}, {"id": "23980", "name": "Anders Husted", "created_at": "2020-08-03T00:46:05+02:00", "email": null, "short_name": "Anders Husted", "sortable_name": "Husted, Anders", "integration_id": null, "login_id": null, "name": "Anders Husted", "short_name": "Anders Husted", "sortable_name": "Husted, Anders", "sis_user_id": null}, {"id": "23531", "name": "Søren Bo Jørgensen", "created_at": "2020-08-03T00:46:05+02:00", "email": null, "short_name": "Søren Bo Jørgensen", "sortable_name": "Jørgensen, Søren Bo", "integration_id": null, "login_id": null, "name": "Søren Bo Jørgensen", "short_name": "Søren Bo Jørgensen", "sortable_name": "Jørgensen, Søren Bo", "sis_user_id": null}]]
```

# Objects? Arrays?

The screenshot shows the DR website homepage. At the top, there are navigation links for NYHEDER (News), DRTV, and DR LYD. Below the navigation, there are six thumbnail images of TV shows: DR1: Lovens Hule, DR3: Nationens staerkeste, P1: LSD kælderen, DR LYD: Annas Margrethe, DR3: Du fucker med de forkerte, and A Very British Scandal. A red button labeled "Seneste nyt" (Latest news) is visible. Below this, there are three news snippets: "EU klagter over Kinas hårdt kurs over for Litauen" (EU files a complaint against China's hard currency policy towards Lithuania), "Børn og skoleelever opfordres stedigt til at ugentlige coronatest" (Children and school students are increasingly urged to undergo weekly COVID-19 tests), and "England skræber storstædelen af coronarestriktionerne fra i dag" (England is scrapping most of its COVID-19 restrictions from today). On the left side, there is a large graphic featuring a medical mask, a hand sanitizer bottle, and a COVID-19 test kit. On the right side, there is a small image of a soldier with a rifle. At the bottom, a red banner reads "Regeringen har meldt genåbning - men ikke".

The screenshot shows the website for Københavns Erhvervsakademi. At the top, there are links for "ALLE UDDANNELSER" and "UDDANNELSER UD FRA INTERESSE". Below this, there is a section titled "ALLE ERHVERVSAKADEMI-UDDANNELSER" featuring a grid of 12 portraits of students or professionals, each representing a different vocational program. The programs listed are: AUTOMATIONSTEKNOLOG, BYGGEKOORDINATOR, BYGGETEKNIKER, DATAMATIKER, DESIGNTEKNOLOG, ENTREPRENØRSKAB OG DESIGN, EL-INSTALLATOR, ENERGITEKNOLOG, IT-TEKNOLOG, KORT- OG LANDMÅLING, MULTIMEDIEDESIGNER, and VVS-INSTALLATOR. Each program name is followed by a small arrow pointing to the right.

# Objects with properties in arrays

The screenshot shows a web browser window for the Business Academy Aarhus website ([baaa.dk/programmes/](https://baaa.dk/programmes/)). The page displays various study programs with their descriptions and images.

**Programmes at Business Academy Aarhus**

**Study start in August**

- Multimedia Design**  
AP degree - 2 years  
For those who would like to work with digital communication and interactive design. The programme is the first part of a Bachelor's programme.
- Digital Concept Development**  
Bachelor's top-up degree - 1½ years  
Get additional qualifications to develop concepts for digital platforms - at both the strategic and the practical level.

**Study start in January**

- IT Technology**  
AP degree [Final intake with study start in January 2022]  
Would you like to work with computers, server and network technology? The programme is the first part of a Bachelor's programme.
- Chemical and Biotechnical Technology and Food Technology**  
Bachelor's top-up degree [Final intake with study start in January 2022]  
Be successful in both national and international laboratory environments, and get updated on the
- Web Development**  
Bachelor's top-up degree [Final intake with study start in January 2022]  
Focus on the development of web technologies within several application fields and distribution platforms.

**Programmes that no longer accept new applicants**

- Chemical and Biotechnical Science**  
AP degree [We no longer accept new applicants for this programme]
- Marketing Management**  
AP degree [We no longer accept new applicants for this programme]  
  
**Chat now**

It's all objects &  
arrays!

# Data Types & Data Structures

Objects & Arrays

# Arrays

## Loops

```
for (let teacher of teachers) {  
  console.log(teacher);  
}
```

```
▶ {name: "Birgitte Kirk Iversen", mail: "bki@baaa.dk"}  main.js:20  
▶ {name: "Michael Hvidtfeldt", mail: "mhv@baaa.dk"}  main.js:20  
▶ {name: "Rasmus Cederdorff", mail: "race@baaa.dk"}  main.js:20
```

# Loops

The screenshot shows a dark-themed web browser window. The title bar reads "Loops: while and for". The address bar shows the URL "https://javascript.info/while-for". The main content area has a large heading "Loops: while and for". Below it, a paragraph states "We often need to repeat actions. For example, outputting goods from a list one after another or just running the same code for each number from 1 to 10. Loops are a way to repeat the same code multiple times." A callout box with a blue border and rounded corners contains the following text: "The for...of and for...in loops" (with an info icon), "A small announcement for advanced readers.", "This article covers only basic loops: `while`, `do..while` and `for(...;...;...)`.", "If you came to this article searching for other types of loops, here are the pointers:", a bulleted list ("• See `for...in` to loop over object properties.", "• See `for...of` and `iterables` for looping over arrays and iterable objects."), and "Otherwise, please read on.".

# For of loop

iterate over arrays or other iterable objects

<https://scrimba.com/learn/introductiontojavascript/for-loops-cMMM8U9>

<https://scrimba.com/learn/introductiontojavascript/challenge-for-loops-cPkpJrcv>

# Loops

```
for (const familyMember of familyMembers) {  
    console.log(familyMember);  
}
```

```
for (let index = 0; index < familyMembers.length; index++) {  
    const familyMember = familyMembers[index];  
    console.log(familyMember);  
}
```

[https://www.w3schools.com/js/js\\_loop\\_for.asp](https://www.w3schools.com/js/js_loop_for.asp)  
<https://javascript.info/array#loops>  
<https://javascript.info/while-for>

# JavaScript.info/array#loops

One of the oldest ways to cycle array items is the `for` loop over indexes:

```
1 let arr = ["Apple", "Orange", "Pear"];
2
3 for (let i = 0; i < arr.length; i++) {
4   alert( arr[i] );
5 }
```



But for arrays there is another form of loop, `for..of`:

```
1 let fruits = ["Apple", "Orange", "Plum"];
2
3 // iterates over array elements
4 for (let fruit of fruits) {
5   alert( fruit );
6 }
```



```
const allMovies = [
  {
    id: 1,
    titel: "The Matrix",
    år: 1999,
    rating: 8.7,
    genre: ["Action", "Sci-Fi"],
    instruktører: ["Lana Wachowski", "Lilly Wachowski"]
  },
  {
    id: 2,
    titel: "Inception",
    år: 2010,
    rating: 8.8,
    genre: ["Action", "Thriller", "Sci-Fi"],
    instruktører: ["Christopher Nolan"]
  },
  {
    id: 3,
    titel: "The Dark Knight",
    år: 2008,
    rating: 9.0,
    genre: ["Action", "Crime", "Drama"],
    instruktører: ["Christopher Nolan"]
  }
];

// Loop through all movies
for (const movie of expandedMovieDatabase) {
  console.log(`🎬 ${movie.title} (${movie.year})`);
  console.log(`⭐ Rating: ${movie.rating}`);
  console.log(`🎭 Genre: ${movie.genre[0]}`);
  console.log("----");
}
```

# ARRAYS

## LOOPS

```
for (let teacher of teachers) {  
  console.log(teacher.mail);  
}
```

bki@baaa.dk

main.js:20

mhv@baaa.dk

main.js:20

race@baaa.dk

main.js:20

# LOOPS

... LOOP THROUGH AN ARRAY AND ADD A CONDITION

```
for (let teacher of teachers) {  
  if (teacher.name === "Rasmus Cederdorff") {  
    console.log(teacher);  
  }  
}
```

# Functions

Write reusable code

# Functions

A block of code to perform a specific task.

A way to make reusable code by storing tasks we can use again and again.

Best practice: write reusable code

```
function log(message) {  
  console.log(message);  
}  
  
log("Hi Frontenders!");
```

<https://javascript.info/function-basics>

# Functions

3 different types

```
function log(message) {  
  console.log(message);  
}
```

FUNCTION DECLARATION

```
const log = function (message) {  
  console.log(message);  
};
```

FUNCTION EXPRESSION

```
const log = (message) => {  
  console.log(message);  
};
```

ARROW FUNCTION

# Functions

## Function declaration

```
console.log("Hi Frontenders!");
console.log("Good job!");
console.log("I'm testing something!");
console.log("Hola");
```

```
function log(message) {
  console.log(message);
}

log("Hi Frontenders!");
log("Good job!");
log("I'm testing something!");
log("Hola");
```

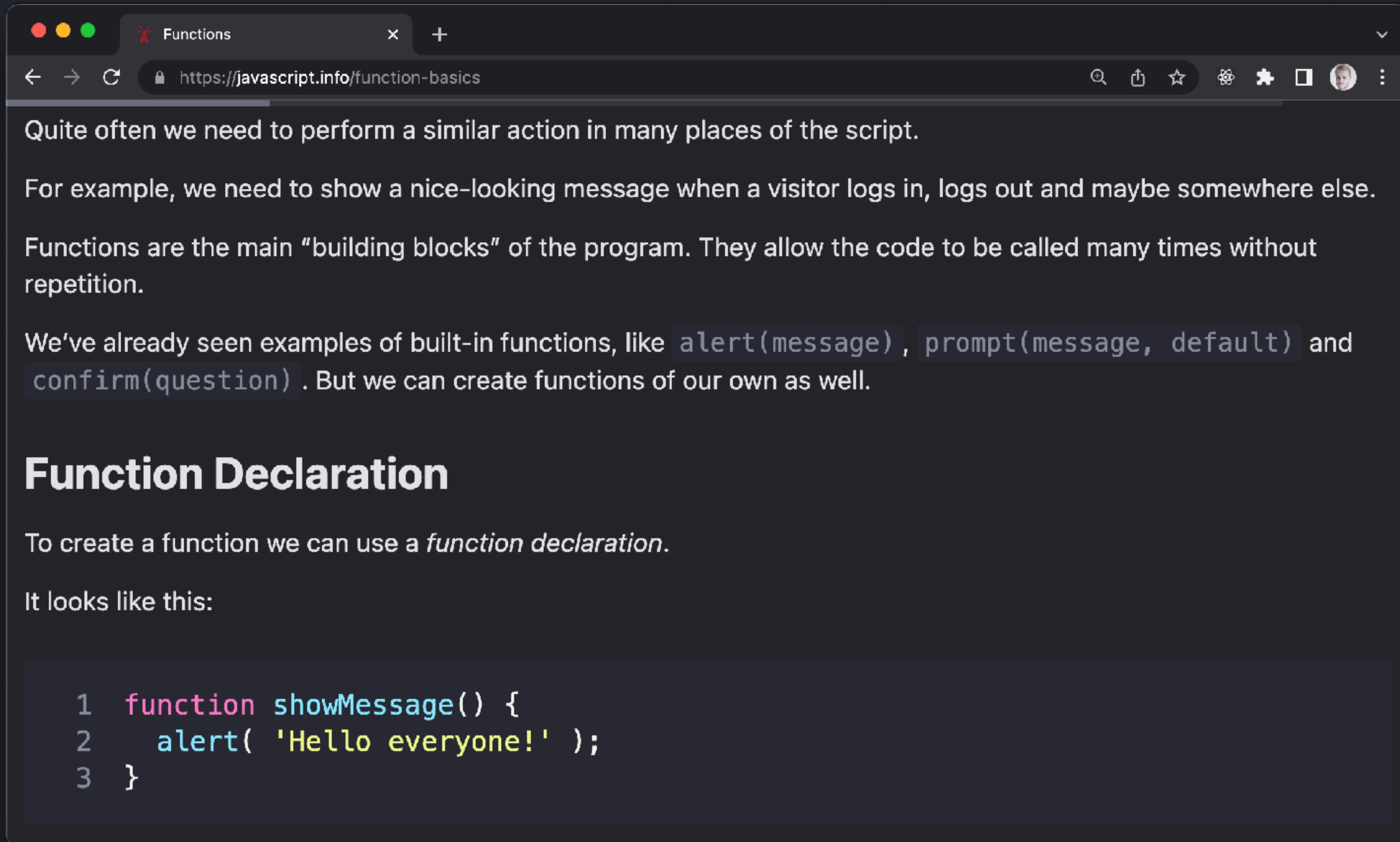
The screenshot shows a web browser window with the title bar "JavaScript Functions". The address bar contains the URL "https://www.w3schools.com/js/js\_functions.asp". The navigation bar includes links for Home, HTML, CSS, JAVASCRIPT (which is highlighted in green), SQL, PYTHON, PHP, and BOOTSTRAP. Below the navigation bar, the main content area has a dark background with white text. The title "JavaScript Function Syntax" is displayed in large, bold letters. The text explains that a JavaScript function is defined with the `function` keyword, followed by a **name**, followed by parentheses `()`. It notes that function names can contain letters, digits, underscores, and dollar signs. It also mentions that parentheses may include parameter names separated by commas, enclosed in parentheses `(parameter1, parameter2, ...)`. The code to be executed is placed inside curly brackets `{ }` . A code block in a light gray box shows the syntax:

```
function name(parameter1, parameter2, parameter3) {  
    // code to be executed  
}
```

Below the code block, three explanatory statements are listed in a light gray box:

- Function **parameters** are listed inside the parentheses `()` in the function definition.
- Function **arguments** are the **values** received by the function when it is invoked.
- Inside the function, the arguments (the parameters) behave as local variables.

# JavaScript.info/Function-Basics

A screenshot of a web browser window titled "Functions". The address bar shows the URL "https://javascript.info/function-basics". The main content area contains text explaining the need for functions and their role as building blocks. It also mentions built-in functions like alert, prompt, and confirm. A code block at the bottom shows a function declaration for "showMessage".

Quite often we need to perform a similar action in many places of the script.

For example, we need to show a nice-looking message when a visitor logs in, logs out and maybe somewhere else.

Functions are the main “building blocks” of the program. They allow the code to be called many times without repetition.

We've already seen examples of built-in functions, like `alert(message)`, `prompt(message, default)` and `confirm(question)`. But we can create functions of our own as well.

## Function Declaration

To create a function we can use a *function declaration*.

It looks like this:

```
1 function showMessage() {  
2     alert( 'Hello everyone!' );  
3 }
```

# Functions

## Function declaration

The name of the function



```
function showMessage() {  
    alert("Hello everyone!");  
}
```



Body of the function  
(code block)

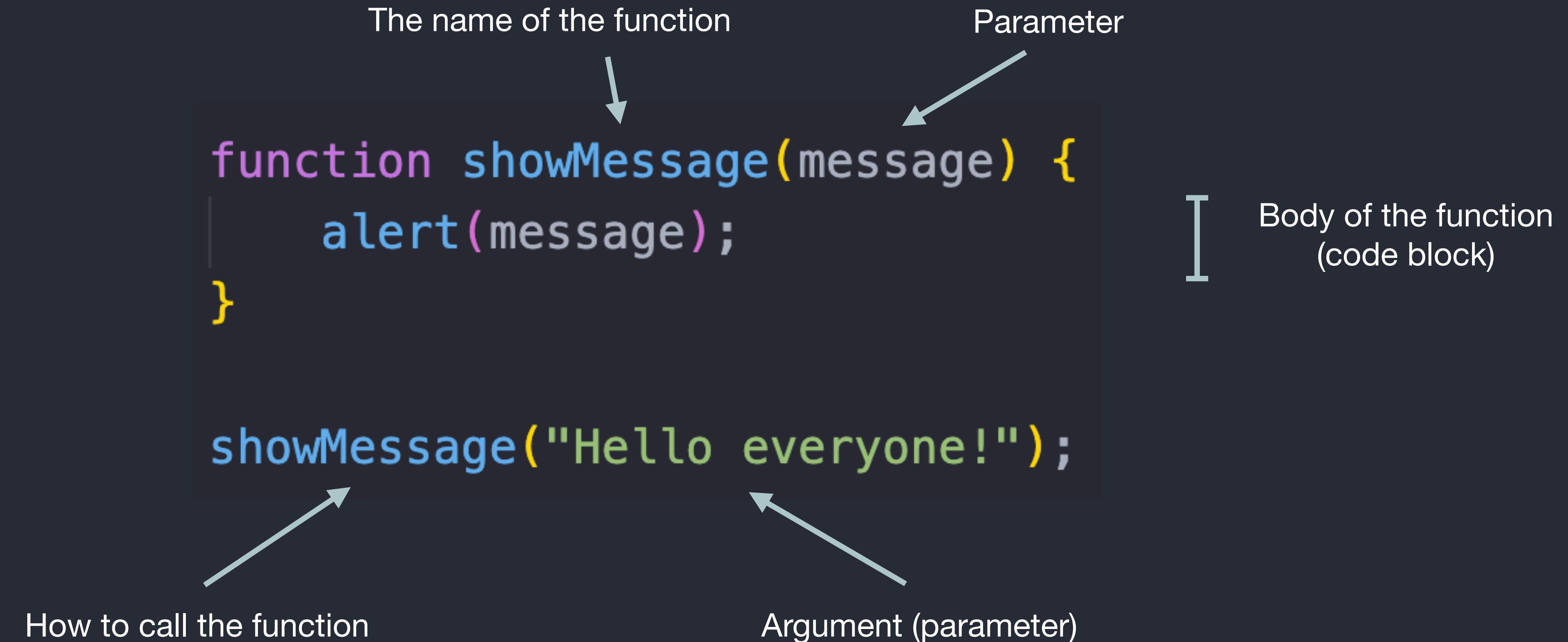
**showMessage();**

How to call the function



# Functions

## Function declaration



# Functions

## Function declaration

Argument passed to the function

```
function showMessage(message) {  
    alert(message);  
}  
  
showMessage("Hello everyone!");  
showMessage("How are you?");  
showMessage("Good, you?");
```

The function can be called as many times as you want  
And with different argument values

*"When a value is passed as a function parameter,  
it's also called an argument.*

*In other words, to put these terms straight:*

- *A parameter is the variable listed inside the parentheses in the function declaration (it's a declaration time term).*
- *An argument is the value that is passed to the function when it is called (it's a call time term)."*

# JavaScript.info/function-basics#parameters

We can pass arbitrary data to functions using parameters.

In the example below, the function has two parameters: `from` and `text`.

```
1 function showMessage(from, text) { // parameters: from, text
2   alert(from + ': ' + text);
3 }
4
5 showMessage('Ann', 'Hello!'); // Ann: Hello! (*)
6 showMessage('Ann', "What's up?"); // Ann: What's up? (**)
```

# Functions

## Arrays & Loops

The name of the function



Parameters



```
function appendTeachers(teachers) {  
  for (let teacher of teachers) {  
    console.log(teacher);  
    document.querySelector("#grid-teachers").innerHTML +=  
      "<article>" +  
      "<img src='" + teacher.img + "'>" +  
      "<h3>" + teacher.name + "</h3>" +  
      teacher.position + "<br>" +  
      "<a href='mailto:" + teacher.mail + "'>" + teacher.mail + "</a>" +  
      "</article>";  
  }  
}  
  
appendTeachers(teachers);
```

How to call the function

Body of the function  
(code block)

TEACHERS



Birgitte Kirk Iversen

Senior Lecturer  
[bki@baaa.dk](mailto:bki@baaa.dk)



Michael Hvidtfeldt

Senior Lecturer  
[mhv@baaa.dk](mailto:mhv@baaa.dk)



Rasmus Cederdorff

Lecturer  
[race@baaa.dk](mailto:race@baaa.dk)

# DOM Manipulation

Change the content of the website with JavaScript

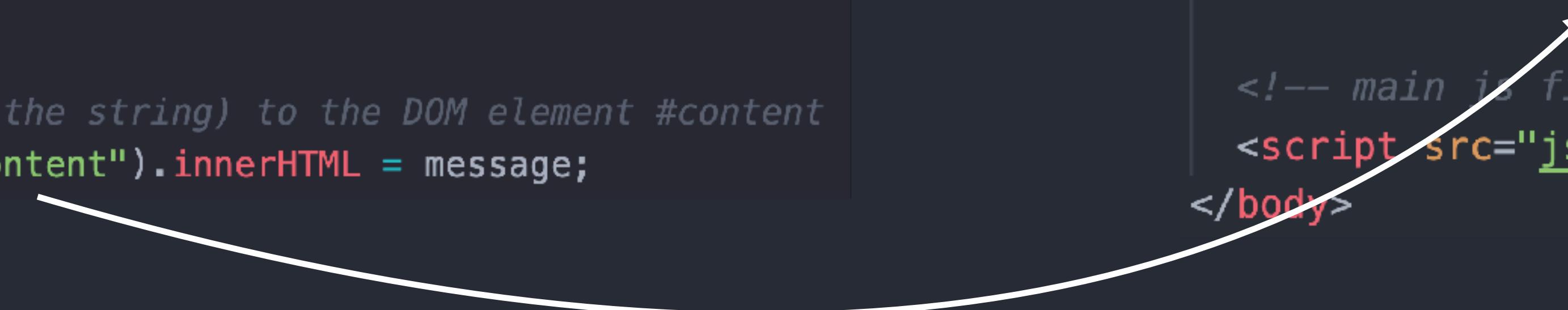
# DOM Manipulation

```
// declaring a variable with a value
let message = "Hi Frontenders!"

//accessing the variable and logging it to the console
console.log(message);

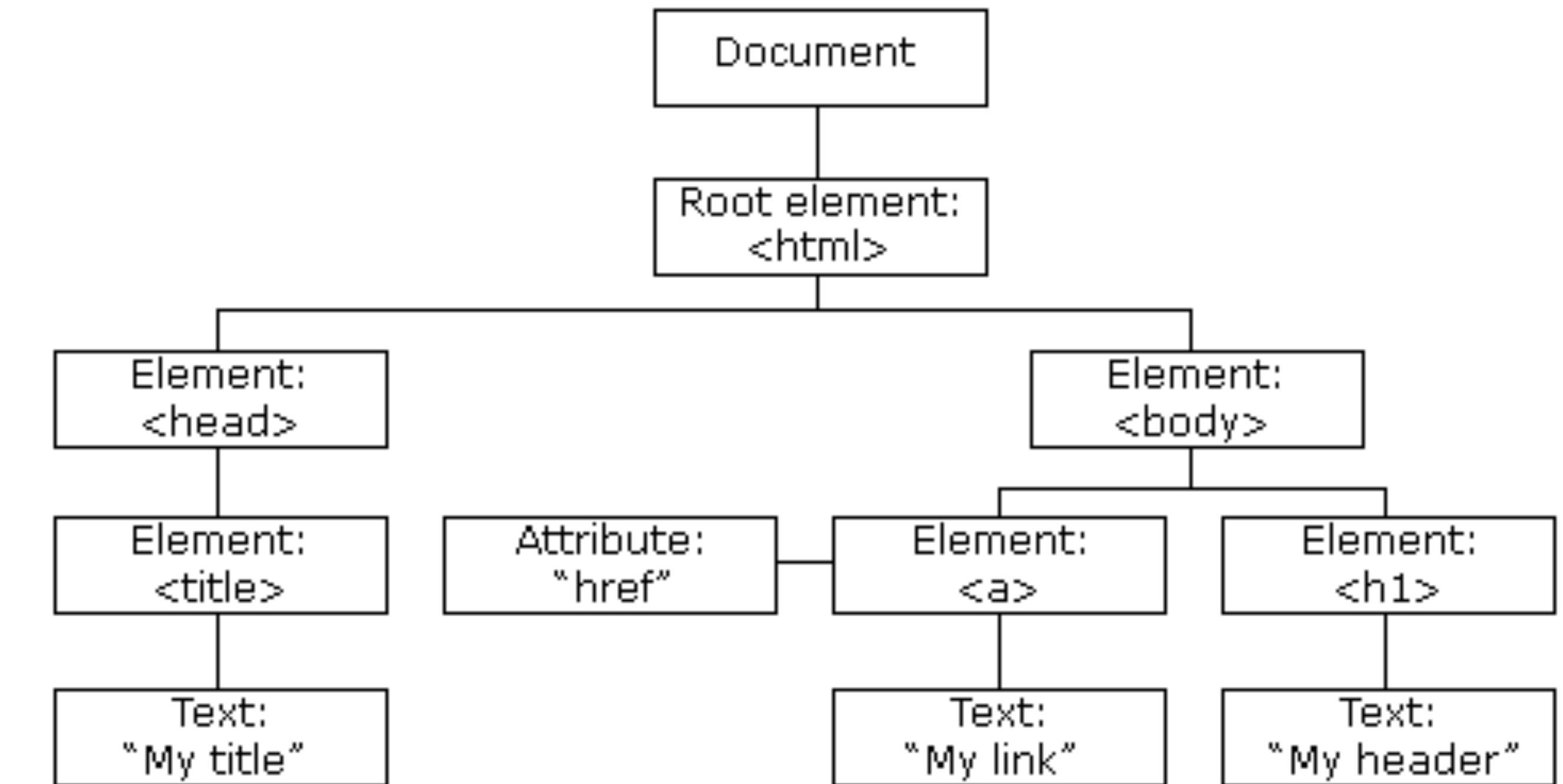
// appending the variable (the string) to the DOM element #content
document.querySelector("#content").innerHTML = message;
```

```
<body>
  <header>
    <h1>PROJECT TEMPLATE</h1>
  </header>
  <section id="content"></section>
  <!-- main is file -->
  <script src="js/main.js"></script>
</body>
```



# JavaScript HTML DOM

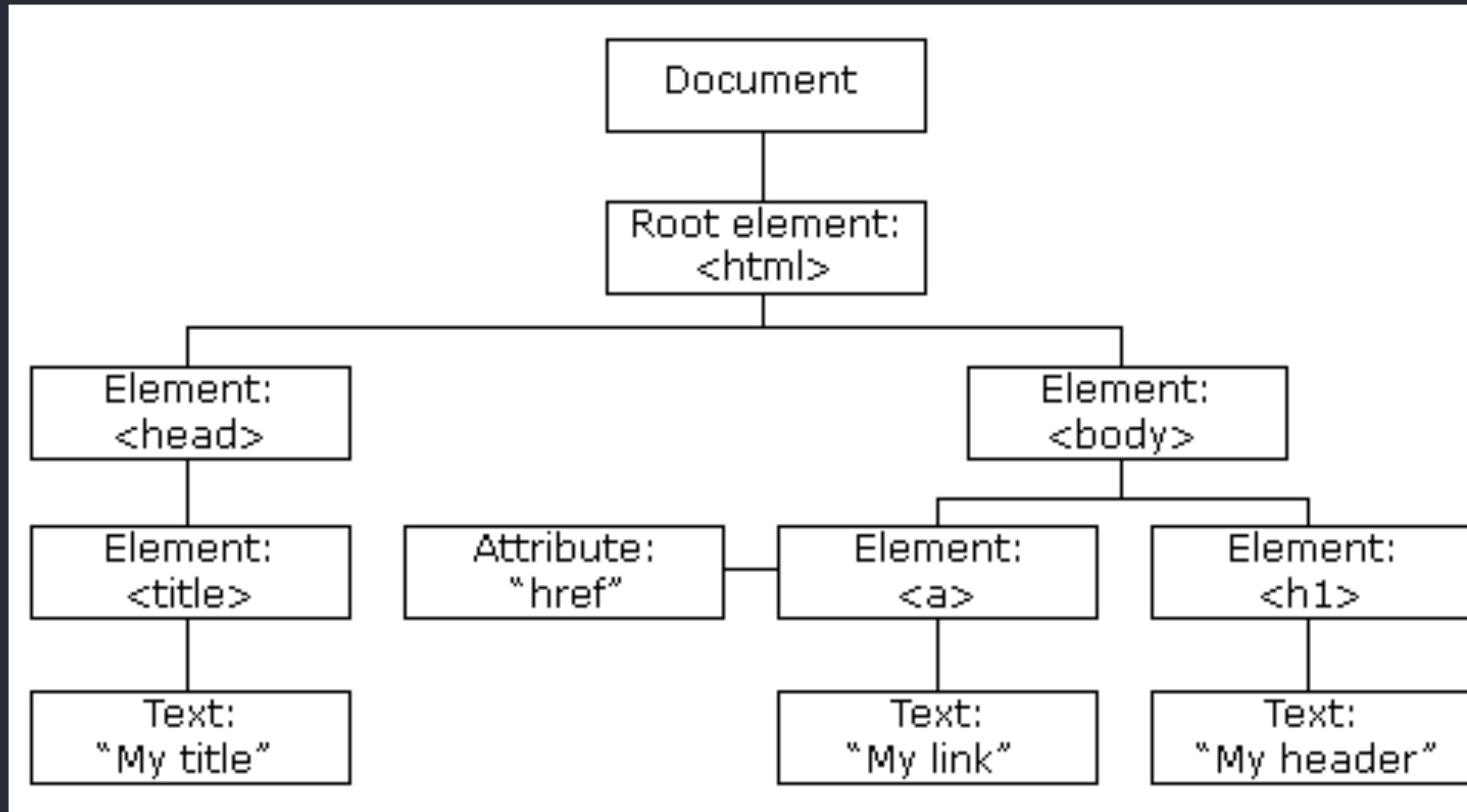
```
index.html ×  
1  <!DOCTYPE html>  
2  <html>  
3  | <head>  
4  | | <title>My title</title>  
5  | </head>  
6  |  
7  <body>  
8  | | <h1>My header</h1>  
9  | | <a href="https://cederdorff.com">My link</a>  
10 | </body>  
11 |  
12 </html>
```



[https://www.w3schools.com/js/js\\_htmldom.asp](https://www.w3schools.com/js/js_htmldom.asp)  
<https://javascript.info/dom-nodes>  
<https://javascript.info/dom-navigation>

# The HTML DOM (Document Object Model)

A model as a tree of Objects



- Object Model for HTML:
  - HTML elements as objects
  - Properties for all HTML elements
  - Methods for all HTML elements
  - Events for all HTML elements

# JavaScript HTML DOM

## Document Object Model

```
index.html ×  
1  <!DOCTYPE html>  
2  <html>  
3  |   <head>  
4  |   |   <title>My title</title>  
5  |   </head>  
6  
7  <body>  
8  |   <h1>My header</h1>  
9  |   <a href="https://cederdorff.com">My link</a>  
10 |</body>  
11 </html>  
12
```

The HTML document as an object

Gives us the power to create dynamic HTML and manipulate with the HTML (the DOM).

JavaScript can:

- ... change all the HTML elements in the page*
- ... change all the HTML attributes in the page*
- ... change all the CSS styles in the page*
- ... remove existing HTML elements and attributes*
- ... add new HTML elements and attributes*
- ... react to all existing HTML events in the page*
- ... create new HTML events in the page*

[https://www.w3schools.com/js/js\\_htmldom.asp](https://www.w3schools.com/js/js_htmldom.asp)

<https://javascript.info/dom-nodes>

<https://javascript.info/dom-navigation>

# Searching the DOM: getElement\* & querySelector\*

```
<section id="elem">
  <article id="elem-content">Element</article>
</section>

<script>
  // get the element
  const element = document.getElementById('elem');
  // make its background red
  element.style.background = 'red';
  // get the elementContent
  const elementContent = document.querySelector('#elem-content');
  // change inner HTML
  elementContent.innerHTML = "<h2>Hi Web Developers!</h2>"
</script>
```

# Searching the DOM: getElementsByTagName\*

```
<section id="elem">
  <article class="elem-content">Element</article>
  <article class="elem-content">Element</article>
  <article class="elem-content">Element</article>
</section>

<script>
  // get all elements matching the selector - returns an array
  const elements = document.getElementsByTagName('elem-content');
  // loop through all elements
  for (const element of elements) {
    element.innerHTML = "<h2>Hi Web Developers!</h2>";
  }
</script>
```

# Searching the DOM: querySelectorAll

```
<section id="elem">
  <article class="elem-content">Element</article>
  <article class="elem-content">Element</article>
  <article class="elem-content">Element</article>
</section>

<script>
  // get all elements matching the selector - returns an array
  const elements = document.querySelectorAll('.elem-content');
  // loop through all elements
  for (const element of elements) {
    element.innerHTML = "<h2>Hi Web Developers!</h2>";
  }
</script>
```

# Insert HTML Element: insertAdjacentHTML

```
const movieList = document.querySelector("#movie-list"); // Find container til film

// Byg HTML struktur dynamisk - template literal med ${} til at indsætte data
const movieHTML = /*html*/
  <article class="movie-card" tabindex="0">
    
    <div class="movie-info">
      <h3>${movie.title} <span class="movie-year">(${movie.year})</span></h3>
      <p class="movie-genre">${movie.genre.join(", ")})</p>
      <p class="movie-rating">★ ${movie.rating}</p>
      <p class="movie-director"><strong>Director:</strong> ${movie.director}</p>
    </div>
  </article>
';

// Tilføj movie card til DOM (HTML) - insertAdjacentHTML sætter HTML ind uden at overskrive
movieList.insertAdjacentHTML("beforeend", movieHTML);
```

# JS HTML DOM

getElement\* or querySelector\*?

# CSS Grid

**CSS Grid is a two-dimensional layout system in CSS that allows you to arrange content in rows and columns with precision and flexibility.**

# CSS Layouts

## Introduction to CSS layout

Normal flow

The display property

Flexbox

Grid

Floats

Positioning

Table layout

Multiple-column layout

# Grid & Flexbox

... two CSS layout models that can be used to create layouts with just a couple of CSS rules.

... responsive and flexible layouts.

# CSS Grids

CSS Grid Teachers

Column 1	Column 2	Column 3
Birgitte Kirk Iversen Senior Lecturer <a href="mailto:bki@mail.dk">bki@mail.dk</a>	Martin Aagaard Nøhr Lecturer <a href="mailto:mnor@mail.dk">mnor@mail.dk</a>	Peter Lind Senior Lecturer <a href="mailto:peli@kua.dk">peli@kua.dk</a>
Column 1	Column 2	Column 3

Header

Column

Column

Column

Footer

Amerikanske chocolate chip cookies

**Cookies**

Klassiske amerikanske cookies - spredt udenpå og bløde indeni... Og hvis der er noget de virkelig kan, derover på den anden side af atlanten, så er det at bage cookies. Om vi kalder dem Chocolate chip cookies eller bare Cookies, ger nok ikke den store forskel. Der er chokoladestykker i - og de smager prægtfuldt. Prøv denne opskrift på den perfekte Chocolate chip cookie, næste gang kiksersluten banker på døren!

**Ingredienser**

**Cookies**

- Blædt smør, 150 g
- Sukker, ca. 115 g
- Brun farrin, ca. 115 g
- Vand/æg, 2 stk.
- Æg, 1
- Hvedemel (ca. 4½ dl), 250 g
- Natur, 1 tsks
- Græn salt, ¼ tsks
- Bagpulver, ½ tsks

**Sådan gør du**

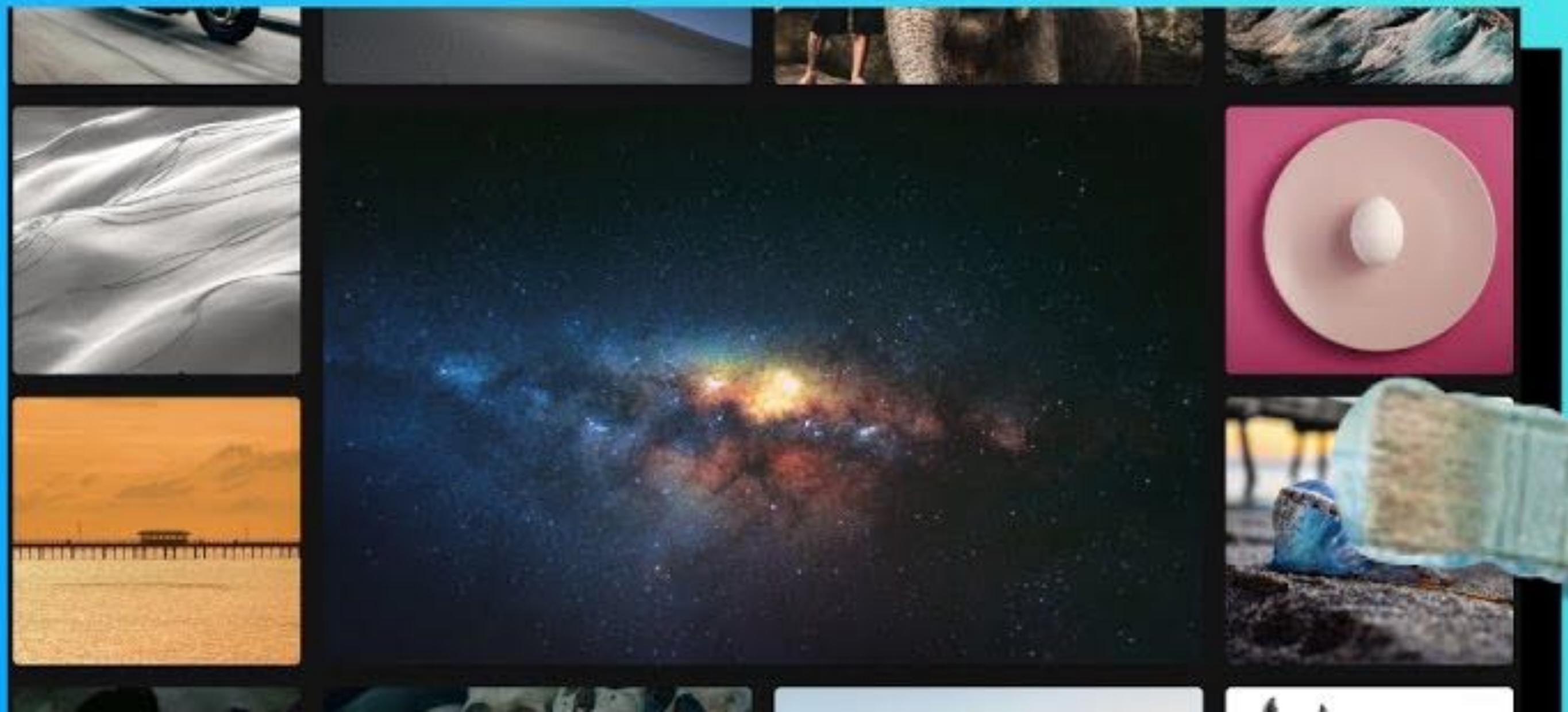
**Cookies**

1. Pak smør, sukker, brun farrin, vanille og æg godt sammen.
2. Bland hvedemel, natur, salt og bagpulver og vær det i smørblandingen sammen med chokolade.
3. Del degen i 20 stykker à ca. 45 g og form dem til kugler.
4. Sat kuglerne på plader med bagpapir - tryk dem let flate. Kagejern fylder en del ud, så seg for, at der er god plads imellem dem.
5. Bag kuglerne til da er pydte, men bløde i midten.
6. Lad kagejern afkøle ca. 6 min. på bagpladen og flyt dem så over på en rist.

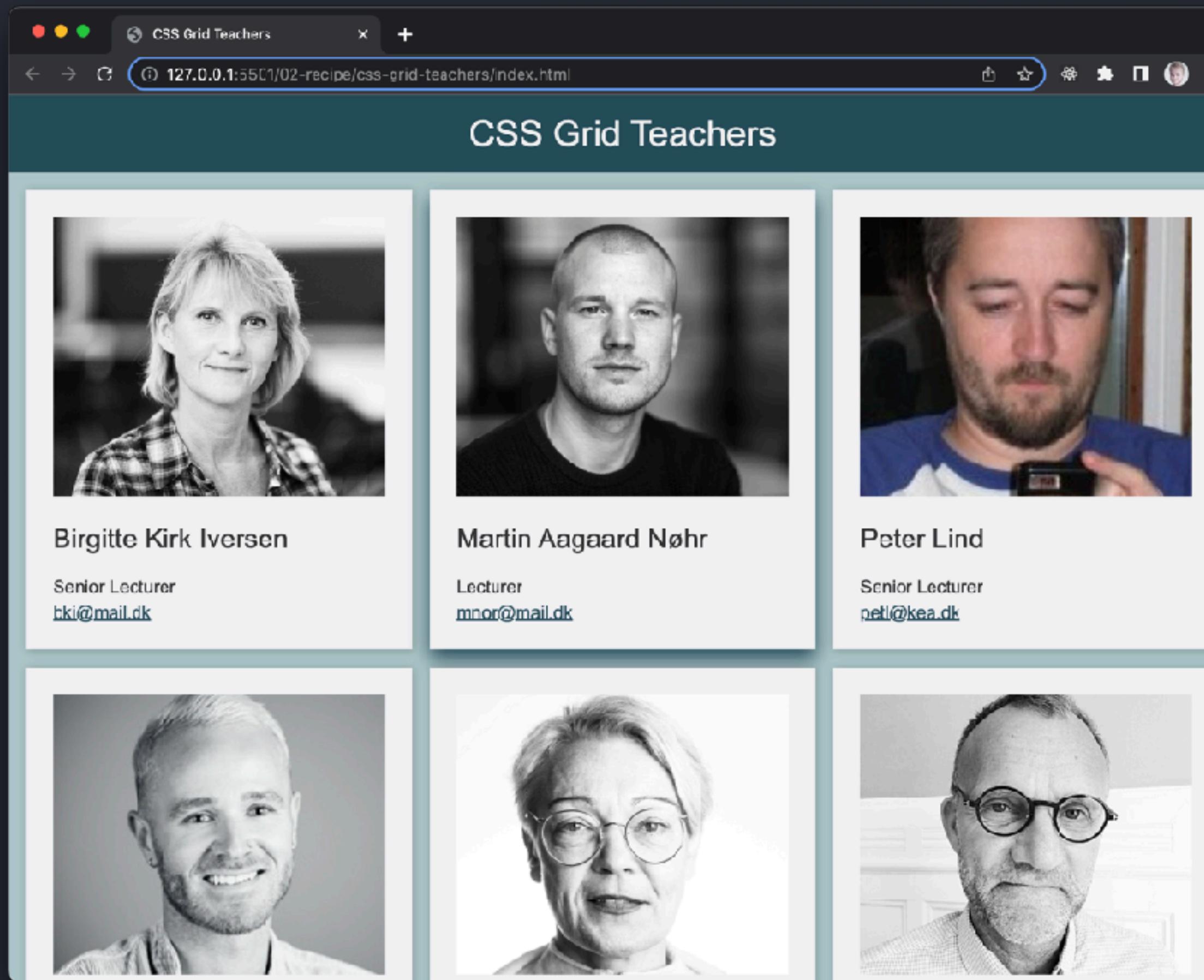
**Bagtid**

# The Joy of

# ART



# CSS Grid



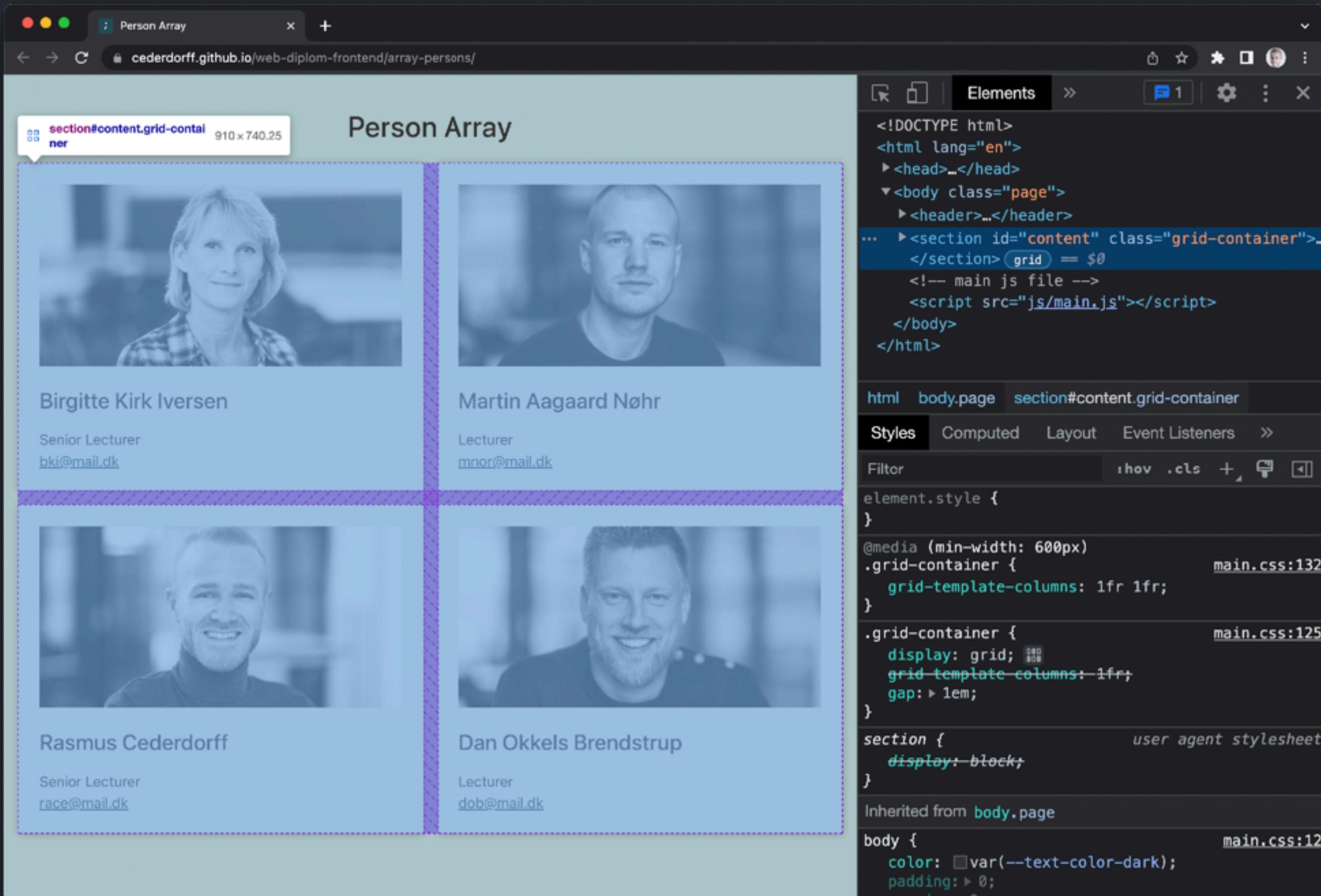
```
<section class="grid-container">...</section>

/* ----- grid container styling ----- */
.grid-container {
  display: grid;
  grid-template-columns: 1fr;
  gap: 1em;
  padding: 1em;
}

@media (min-width: 600px) {
  .grid-container {
    grid-template-columns: 1fr 1fr;
  }
}

@media (min-width: 992px) {
  .grid-container {
    grid-template-columns: 1fr 1fr 1fr;
  }
}
```

# CSS Grid



The screenshot shows a web page titled "Person Array" with four cards arranged in a 2x2 grid. Each card contains a portrait photo and the following information:

- Birgitte Kirk Iversen**: Senior Lecturer, [bki@mail.dk](mailto:bki@mail.dk)
- Martin Aagaard Nøhr**: Lecturer, [mnor@mail.dk](mailto:mnor@mail.dk)
- Rasmus Cederdorff**: Senior Lecturer, [race@mail.dk](mailto:race@mail.dk)
- Dan Okkels Brendstrup**: Lecturer, [dob@mail.dk](mailto:dob@mail.dk)

A developer tools sidebar is visible on the right, showing the DOM structure and CSS styles for the grid container.

```
<section id="content" class="grid-container"></section>

/* ----- grid container styling ----- */
.grid-container {
  display: grid;
  grid-template-columns: 1fr;
  gap: 1em;
}

@media (min-width: 600px) {
  .grid-container {
    grid-template-columns: 1fr 1fr;
  }
}

@media (min-width: 992px) {
  .grid-container {
    grid-template-columns: 1fr 1fr 1fr;
  }
}
```

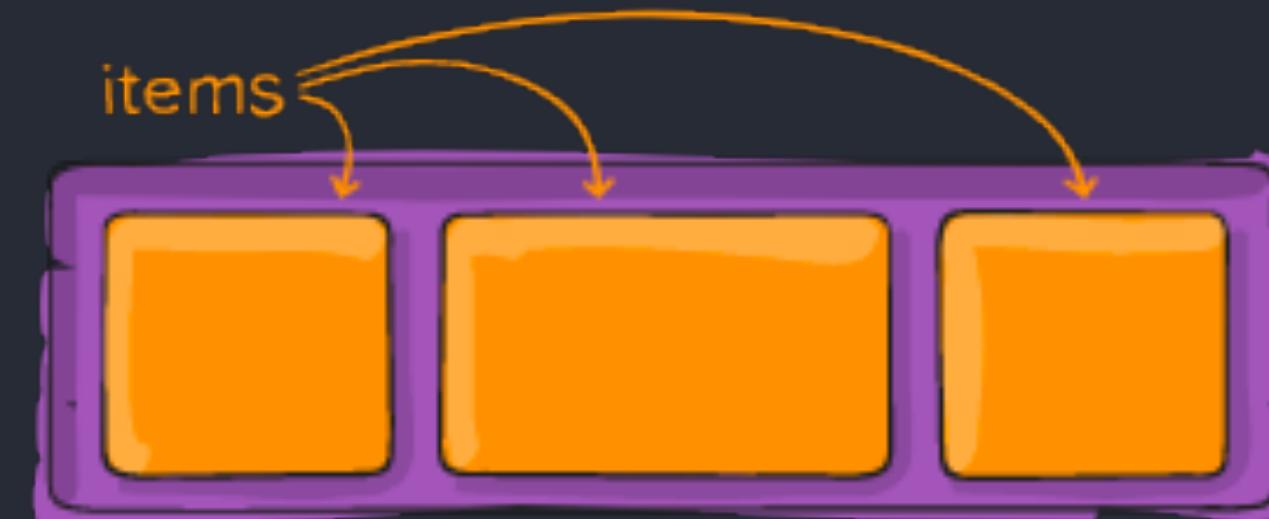
```
<section id="content" class="grid-container"></section>

/* ----- grid container styling ----- */
.grid-container {
  display: grid;
  grid-template-columns: 1fr;
  gap: 1em;
}

@media (min-width: 600px) {
  .grid-container {
    grid-template-columns: 1fr 1fr;
  }
}

@media (min-width: 992px) {
  .grid-container {
    grid-template-columns: 1fr 1fr 1fr;
  }
}
```

# CSS Grid



```
/* ----- grid container styling ----- */
.grid-container {
  display: grid;
  grid-template-columns: 1fr;
  gap: 1em;
  padding: 1em;
}

@media (min-width: 600px) {
  .grid-container {
    grid-template-columns: 1fr 1fr;
  }
}

@media (min-width: 992px) {
  .grid-container {
    grid-template-columns: 1fr 1fr 1fr;
  }
}
```

```
.grid-item {
  box-shadow: 1px 1px 8px □rgba(0, 0, 0, 0.25);
  padding: 1.5em;
  background-color: ■#f1f1f4;
  transition: 0.5s;
  animation: fadeIn 0.5s;
}

.grid-item:hover {
  box-shadow: 0 8px 16px 0 □rgb(38, 76, 89);
}

.grid-item img {
  width: 100%;
  height: 250px;
  object-fit: cover;
}

.grid-item p {
  margin: 0.3em 0;
```

# Grid Container

CSS Grid Layout

w3schools.com/css/css\_grid.asp

HTML CSS JAVASCRIPT

## Display Property

An HTML element becomes a grid container when its `display` property is set to `grid` or `inline-grid`.

### Example

```
.grid-container {  
  display: grid;  
}
```

[Try it Yourself »](#)

### Example

```
.grid-container {  
  display: inline-grid;  
}
```

[Try it Yourself »](#)

All direct children of the grid container automatically become *grid items*.

### Example

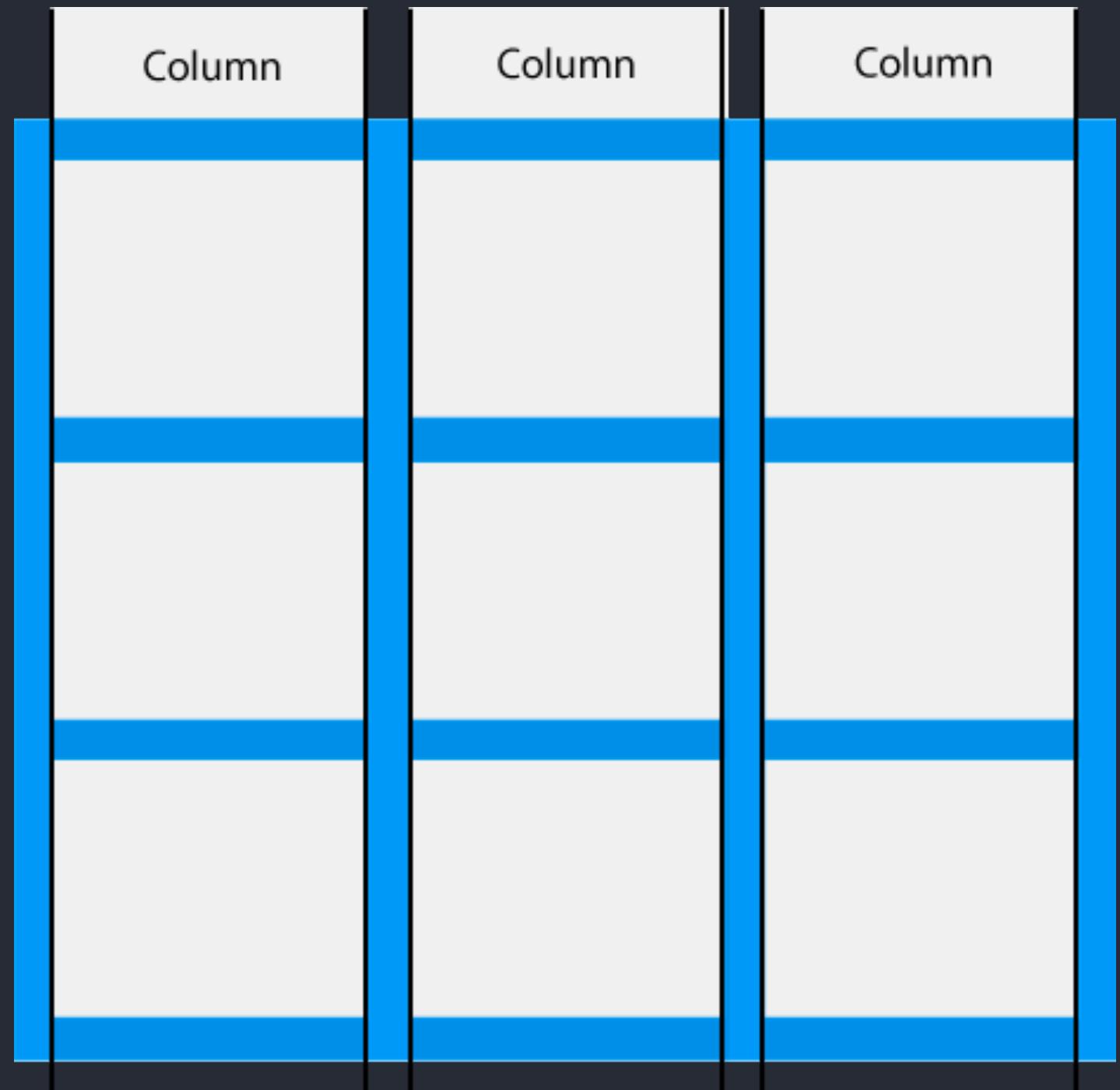
```
<div class="grid-container">  
  <div class="grid-item">1</div>  
  <div class="grid-item">2</div>  
  <div class="grid-item">3</div>  
  <div class="grid-item">4</div>  
  <div class="grid-item">5</div>  
  <div class="grid-item">6</div>  
  <div class="grid-item">7</div>  
  <div class="grid-item">8</div>  
  <div class="grid-item">9</div>  
</div>
```



The image shows a 3x3 grid of numbers from 1 to 9, arranged in three rows and three columns. Each number is contained within a separate blue-bordered box. These boxes represent the grid items. They are arranged in a single row within a larger blue-bordered frame, which represents the grid container.

# Grid Columns

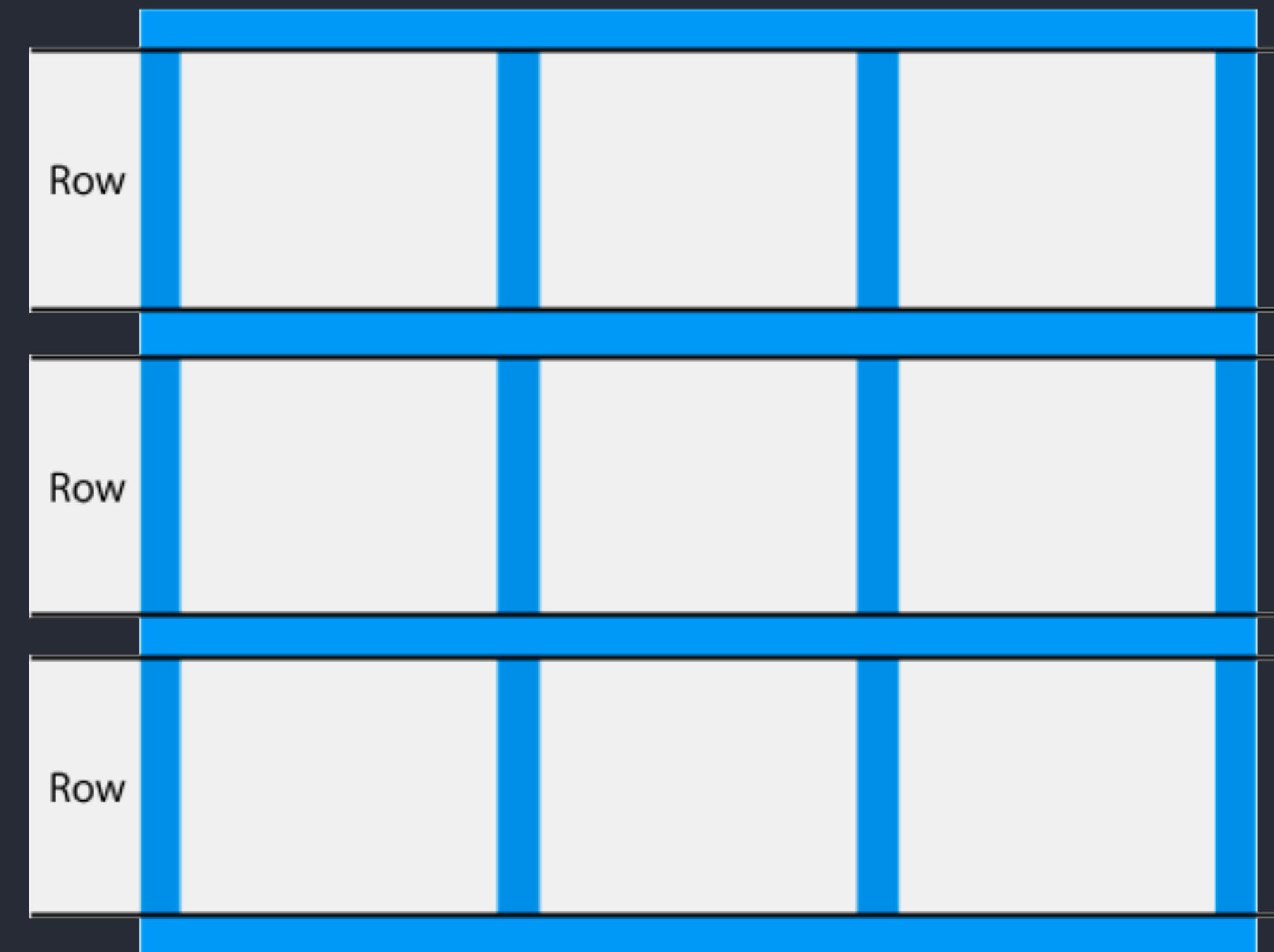
Vertical Lines



```
.grid-container {  
    display: grid;  
    grid-template-columns: auto auto auto auto;  
}
```

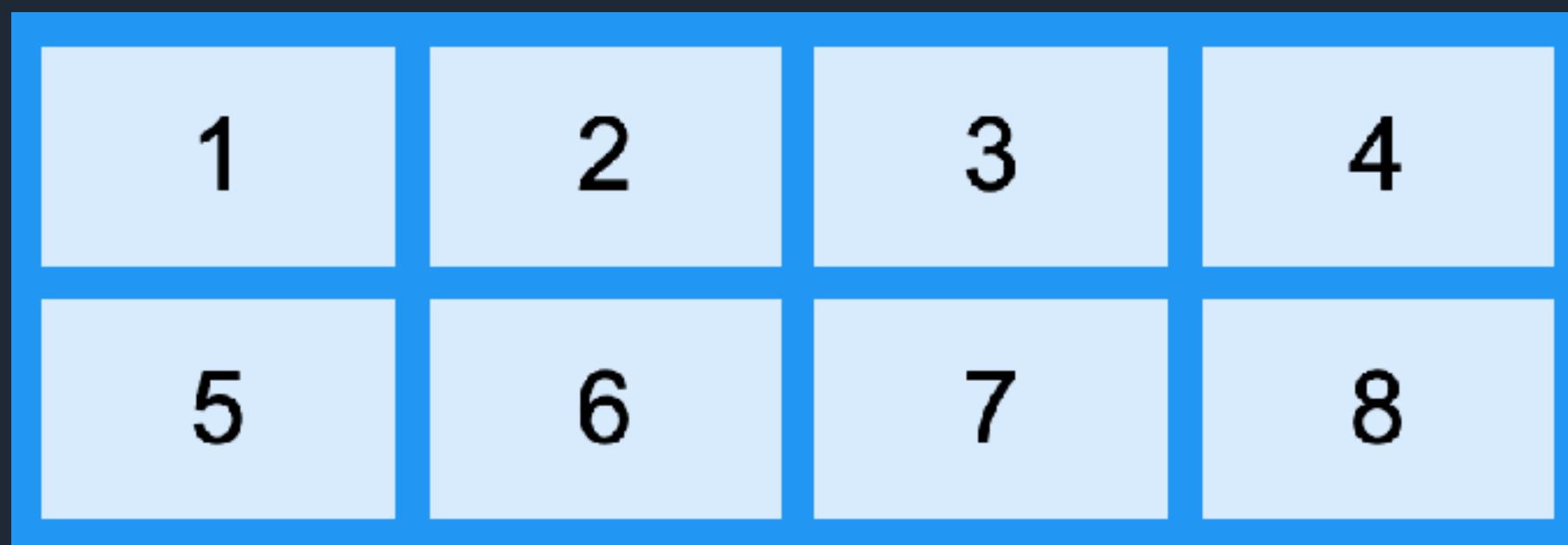
# Grid Rows

Horisontal Lines



```
.grid-container {  
    display: grid;  
    grid-template-rows: 100px 300px;  
}
```

# grid-template-columns



## The grid-template-columns Property

The `grid-template-columns` property defines the number of columns in your grid layout, and it can define the width of each column.

Make a grid with 4 columns:

```
.grid-container {  
    display: grid;  
    grid-template-columns: auto auto auto auto;  
}
```

Set a size for the 4 columns:

```
.grid-container {  
    display: grid;  
    grid-template-columns: 80px 200px auto 40px;  
}
```

[https://www.w3schools.com/css/css\\_grid.asp](https://www.w3schools.com/css/css_grid.asp)

# grid-template-rows

## The grid-template-rows Property

The `grid-template-rows` property defines the height of each row.

1	2	3	4
5	6	7	8

The value is a space-separated-list, where each value defines the height of the respective row:

```
.grid-container {  
    display: grid;  
    grid-template-rows: 80px 200px;  
}
```

[https://www.w3schools.com/css/css\\_grid.asp](https://www.w3schools.com/css/css_grid.asp)

## The justify-content Property

The `justify-content` property is used to align the whole grid inside the container.

**Note:** The grid's total width has to be less than the container's width for the `justify-content` property to have any effect.

### Example

```
.grid-container {  
  display: grid;  
  justify-content: space-around;  
}
```

[Try it Yourself »](#)

### Example

```
.grid-container {  
  display: grid;  
  justify-content: space-around;  
}
```

[Try it Yourself »](#)

## The align-content Property

The `align-content` property is used to *vertically* align the whole grid inside the container.

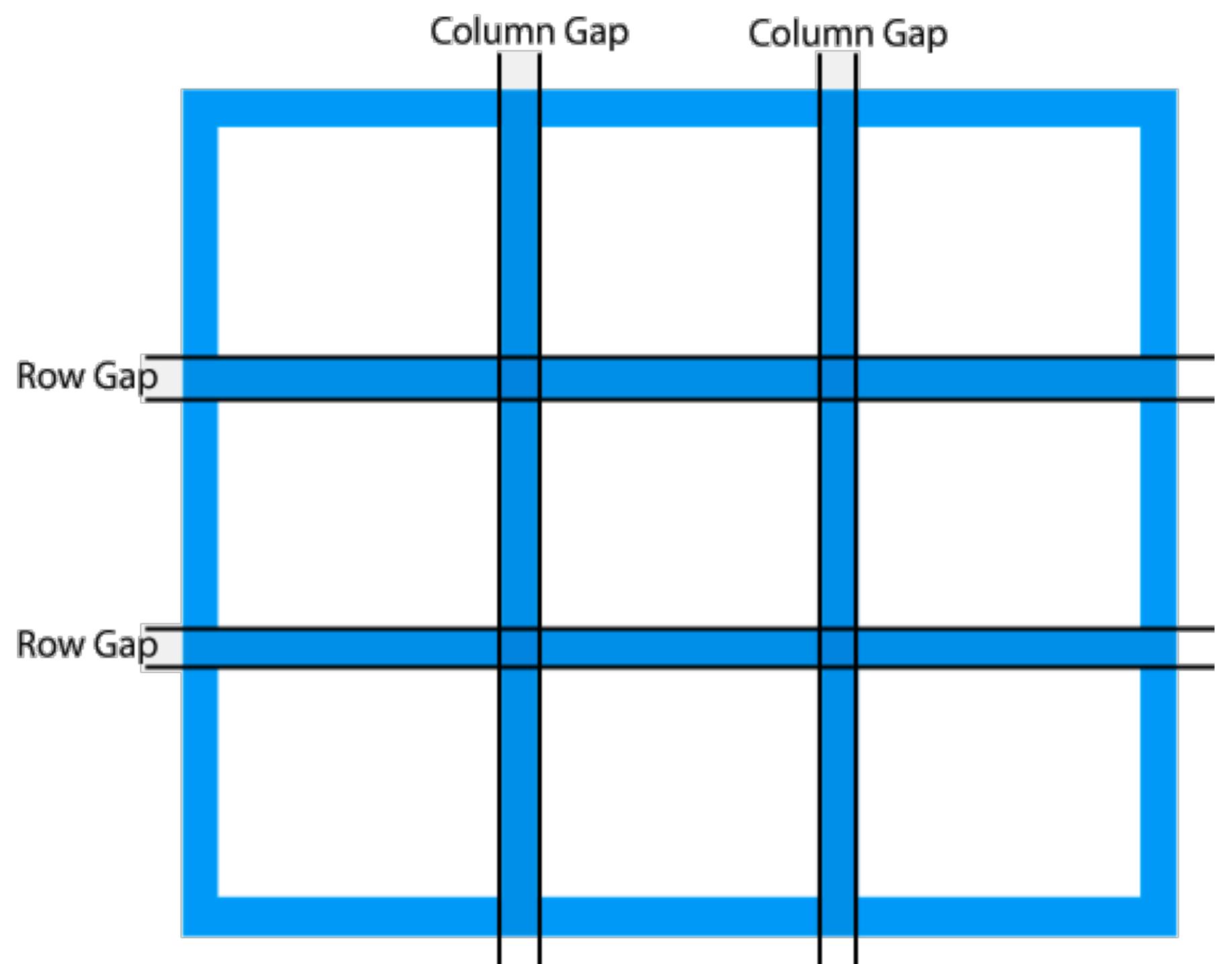
**Note:** The grid's total height has to be less than the container's height for the `align-content` property to have any effect.

### Example

```
.grid-container {  
  display: grid;  
  height: 400px;  
  align-content: center;  
}
```

[Try it Yourself »](#)

# Grid Gap



The `column-gap` property sets the gap between the columns:

```
.grid-container {  
    display: grid;  
    column-gap: 50px;  
}
```

The `row-gap` property sets the gap between the rows:

```
.grid-container {  
    display: grid;  
    row-gap: 50px;  
}
```

The `gap` property is a shorthand property for the `row-gap` and the `column-gap` properties:

```
.grid-container {  
    display: grid;  
    gap: 50px 100px;  
}
```

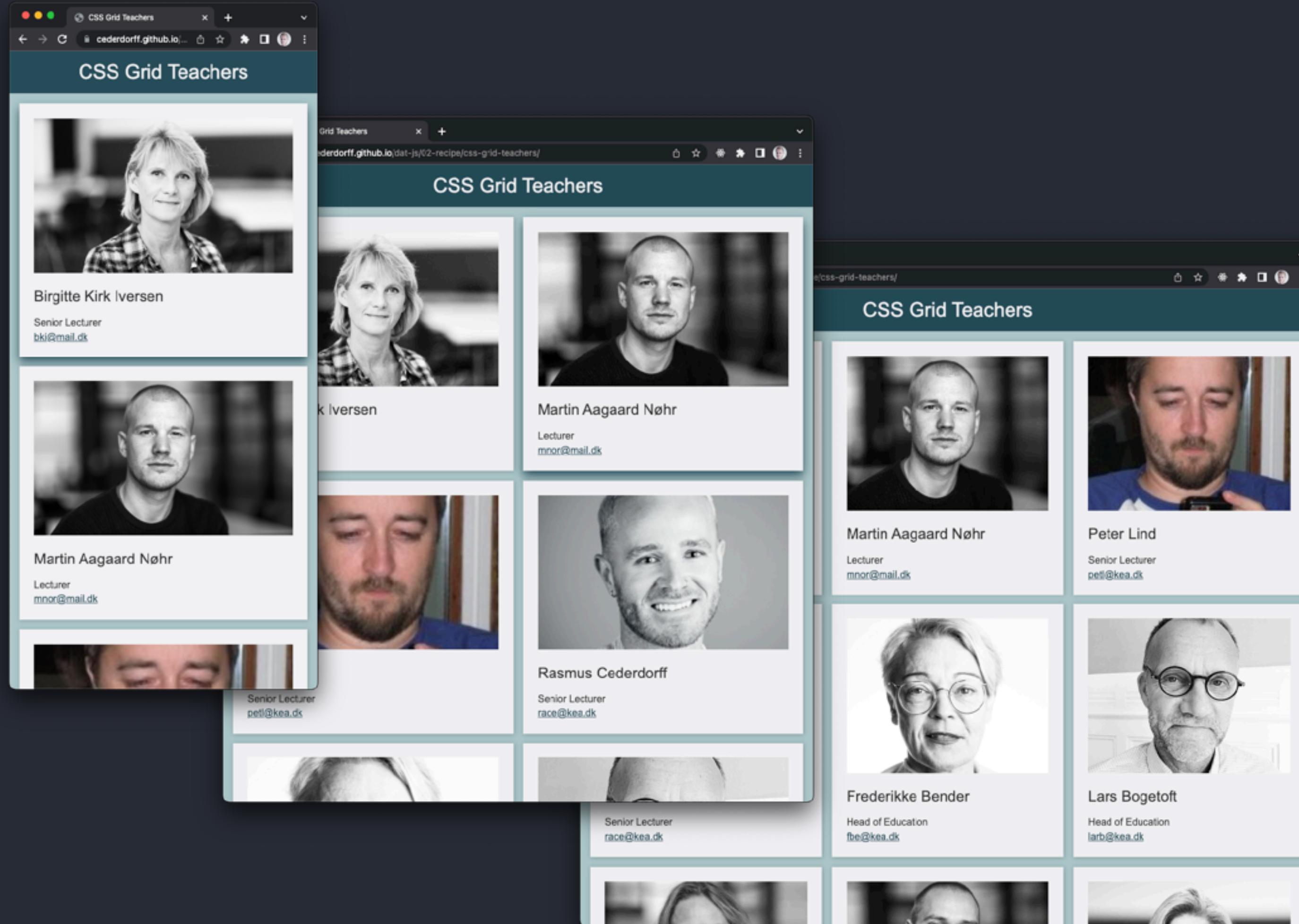
The `gap` property can also be used to set both the row gap and the column gap in one value:

```
.grid-container {  
    display: grid;  
    gap: 50px;  
}
```

100 seconds of **css**

**GIRL ON DIARY OUT**

# CSS Grid & media queries



```
/* ----- grid container styling ----- */
.grid-container {
  display: grid;
  grid-template-columns: 1fr;
  gap: 1em;
  padding: 1em;
}

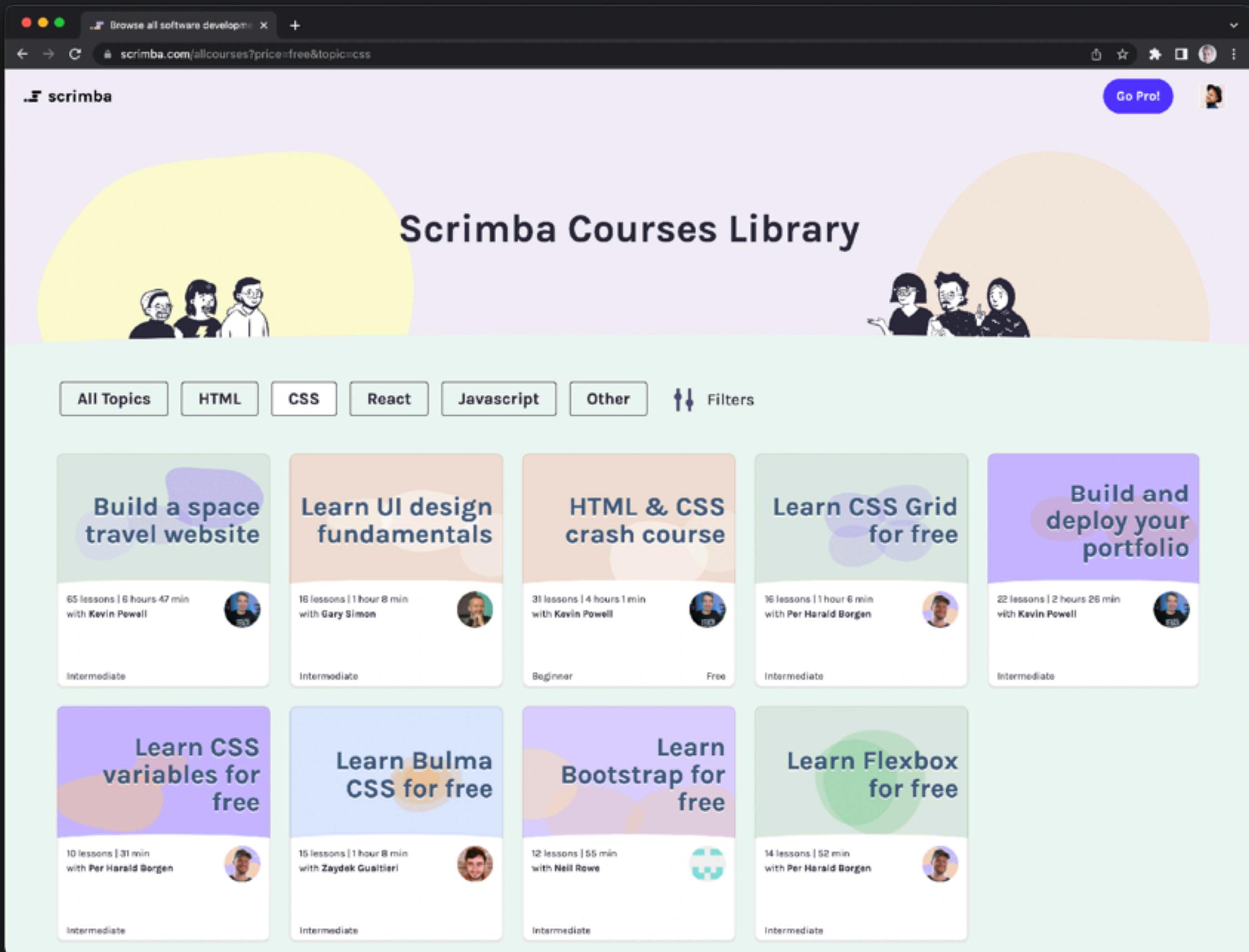
@media (min-width: 600px) {
  .grid-container {
    grid-template-columns: 1fr 1fr;
  }
}

@media (min-width: 992px) {
  .grid-container {
    grid-template-columns: 1fr 1fr 1fr;
  }
}
```

# Scrimba Courses

Learn CSS Grid for free: <https://scrimba.com/learn/cssgrid>

Learn Flexbox for free: <https://scrimba.com/learn/flexbox>



<https://scrimba.com/allcourses>

# CSS Grid Layout Module

The screenshot shows the 'CSS Grid Layout' module page. At the top, there's a navigation bar with links for Tutorials, References, Exercises, and Sign Up. A 'Login' button is also present. Below the navigation, there's a main title 'CSS Grid Layout Module'. Underneath the title, there are two green buttons: '< Previous' and 'Next >'. The main content area displays a wireframe of a web page layout divided into four horizontal sections: Header, Main, Right, and Footer. The 'Main' section is further divided into 'Menu' and 'Content'. A 'Try it Yourself' button is located at the bottom left.

The screenshot shows the 'CSS Grid Container' page. The top navigation bar includes links for HTML, CSS (which is highlighted), JAVASCRIPT, SQL, PYTHON, and JAVA. Below the navigation, the title is 'Grid Container'. It explains that to make an element a grid container, you set the `display` property to `grid` or `inline-grid`. It notes that grid containers consist of grid items placed in columns and rows. A section titled 'The grid-template-columns Property' discusses how this property defines the number of columns and their widths. An example shows a grid with 4 columns. The code provided is:

```
.grid-container {  
  display: grid;  
  grid-template-columns: auto auto auto auto;  
}
```

The screenshot shows the 'CSS Grid Item' page. The top navigation bar includes links for HTML, CSS (highlighted), JAVASCRIPT, SQL, PYTHON, and JAVA. Below the navigation, the title is 'CSS Grid Item'. It features a large diagram of a grid with 5 items labeled 1 through 5. Item 1 is in the top-left, item 2 is in the top-right, item 3 is in the middle-left, item 4 is in the middle-right, and item 5 is in the bottom. A 'Try it Yourself' button is located at the bottom left. A note at the bottom states: 'A grid container contains grid items.'

w CSS Templates

w3schools.com/css/css\_templates.asp

HTML CSS JAVASCRIPT SQL PYTHON PHP BOOTSTRAP HOW TO W3.CSS JAVA JQUERY C++

CSS Multiple Columns  
CSS User Interface  
CSS Variables  
CSS Box Sizing  
CSS Media Queries  
CSS MQ Examples  
CSS Flexbox

**CSS Layout Templates**

We have created some responsive starter templates with CSS.

You are free to modify, save, share, and use them in all your projects.

Header, equal columns and footer:

Try it (using float) »

Try it (using flexbox) »

Try it (using grid) »

Header, unequal columns and footer:

Try it (using float) »

Try it (using flexbox) »

Try it (using grid) »

Topnav, content and footer:

Sidenav and content:

NetSikker inkluderet  
sikrer dig hjælp ved digitale krænkelser  
Se vilkår på telenor.dk

NEW  
We just launched W3Schools videos  
Explore now

COLOR PICKER

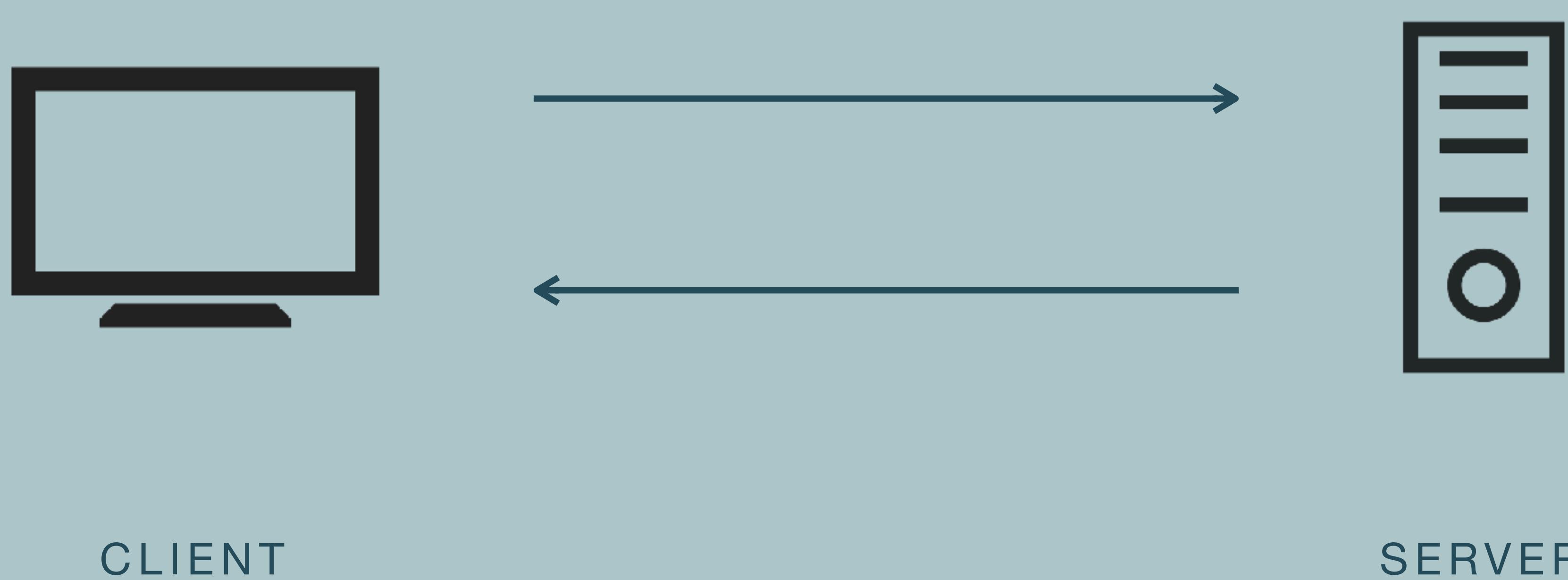
LIKE US

# Fetch

**Fetch is a built-in JavaScript API (functionality) for requesting data from a server.**  
**Fetch = get data from a server.**

# Client-Server Model

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



# fetch(...)

HTTP Requests in  
JavaScript.

A way to get and post data  
from and to data sources.

```
// fetch with callbacks
fetch("https://cederdorff.github.io/web-frontend/callback.js")
  .then(function (response) {
    return response.json();
  })
  .then(function (data) {
    console.log(data);
  });
// or with promises
const response = fetch("https://cederdorff.github.io/web-frontend/promise.js");
const data = response.json();
console.log(data);
```

# fetch(...)

HTTP Requests in  
JavaScript.

A way to get and post data  
from and to data sources.

getCharacters fetches a list of characters  
from a JSON data source, parses the JSON  
to JS and returns the data.

```
async function getCharacters() {  
  const response = await fetch(url);  
  const data = await response.json();  
  console.log(data);  
  return data;  
}
```

# fetch(...)

## Hent og send data med JavaScript

- `fetch()` bruges til at lave HTTP requests – fx hente data fra en API eller sende data til en server.
- Giver adgang til eksterne data-kilder (JSON, REST API, osv.)

Kort fortalt (flow af kode):

`fetch()` → henter data → parser  
JSON → giver dig JavaScript-objekter,  
du kan bruge i din app.

```
// Fetch movies from JSON file - asynkron funktion der returnerer promise
async function getMovies() {
    // Hent data fra URL - await venter på svar før vi fortsætter
    const response = await fetch("https://raw.githubusercontent.com/tjdev/learn-the-dom/master/assets/movies.json");
    // Parser JSON til JS array og gem i global variable
    const data = await response.json();
    // Gør noget med data
    console.log(data);
}
```

# fetch(...)

## HTTP Requests i JavaScript

- `fetch()` returnerer et Promise – et løfte om data, der kommer senere.
- `await` fortæller JavaScript, at den skal vente på et svar, før den går videre i koden.
- Når du bruger `await`, skal funktionen være asynkron – det betyder, du skal skrive `async` foran funktionen.

*Asynkront betyder: JavaScript kan lave andet, mens den venter på data.*

```
// Fetch movies from JSON file - asynkron funktion der venter på et svar fra en URL
async function getMovies() {
    // Hent data fra URL - await venter på svar før vi kan bruge det
    const response = await fetch("https://raw.githubusercontent.com/GeekyAnts/Firebase-Realtime-Databas.../master/movies.json");
    // Parser JSON til JS array og gem i global variable
    const data = await response.json();
    // Gør noget med data
    console.log(data);
}
```

# fetch(...)

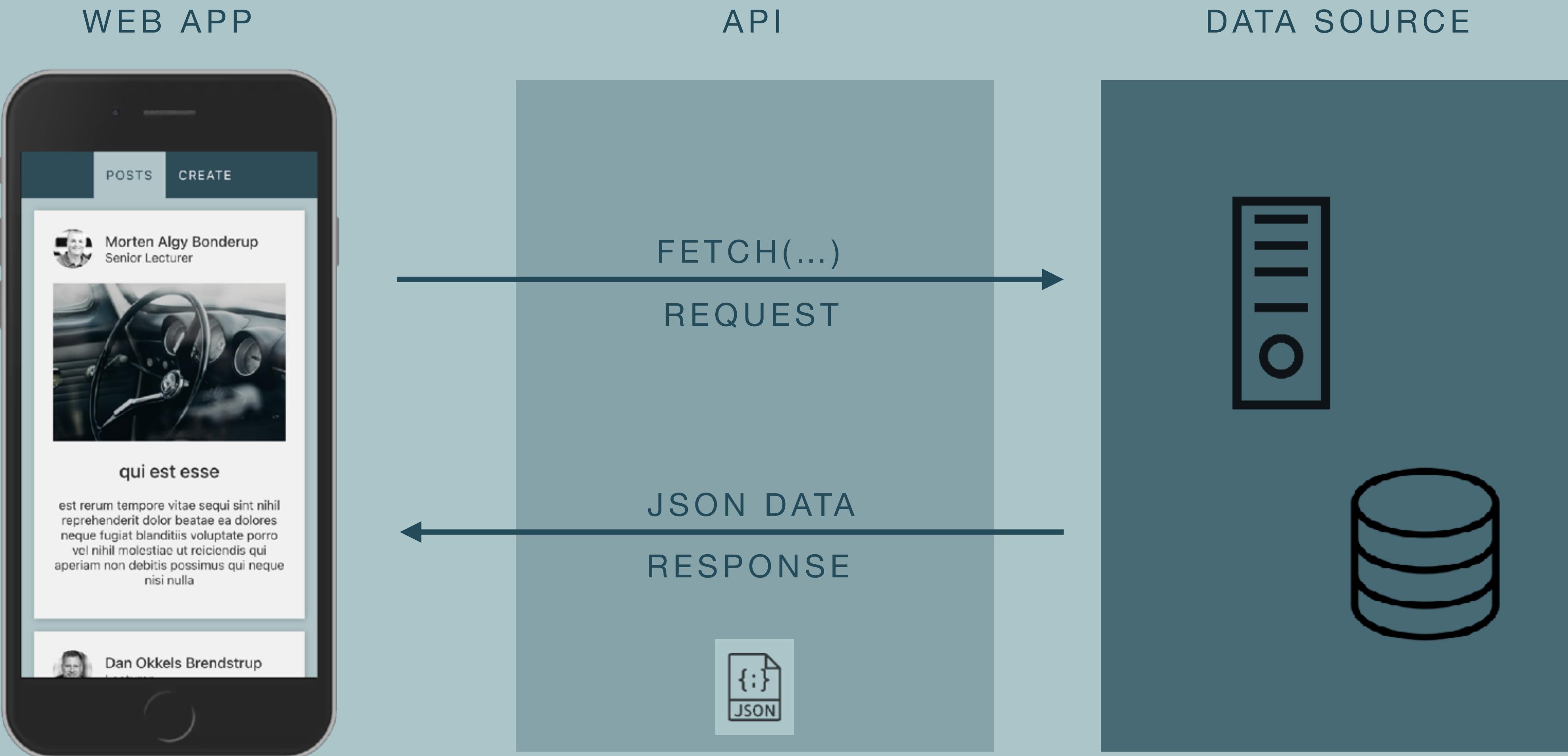
... get & post data from and to a data source  
... can perform network requests to a server

```
1 let promise = fetch(url, [options])
```

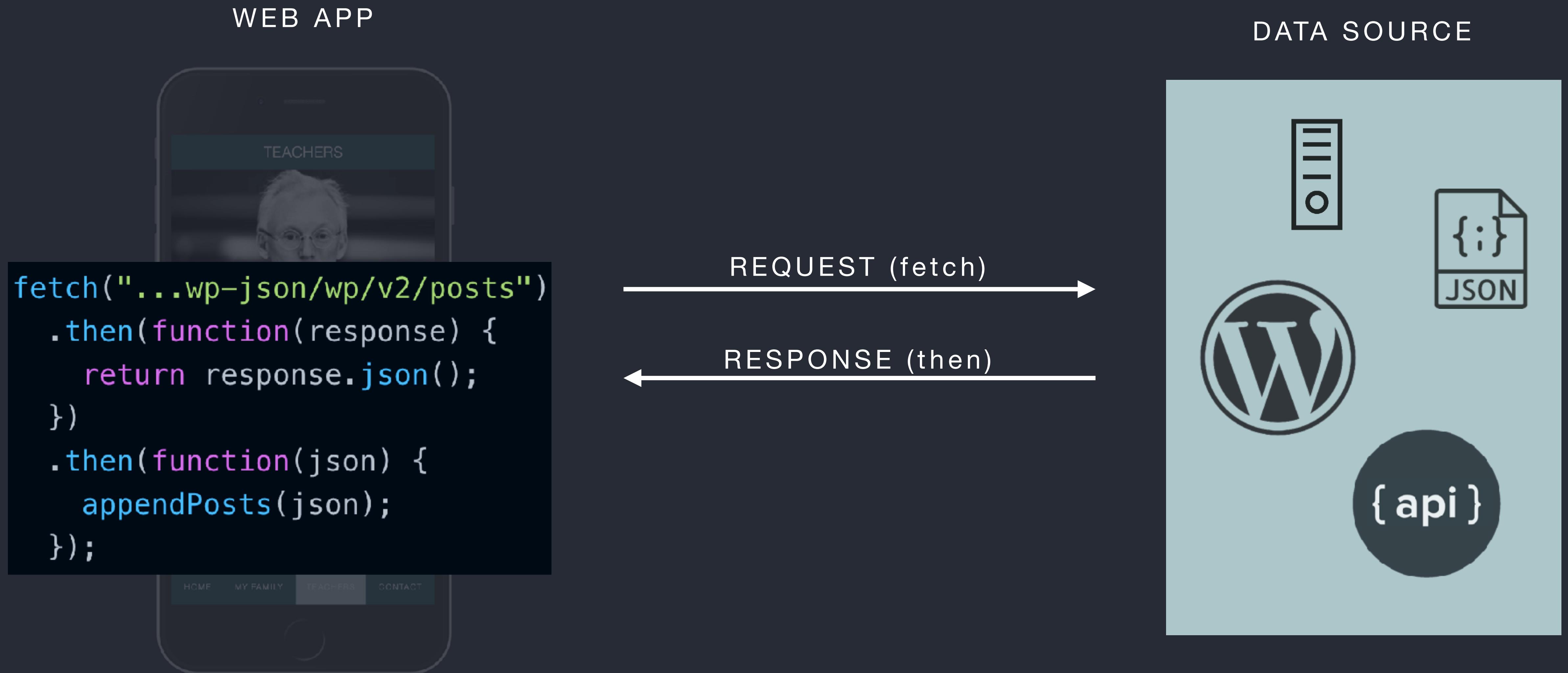
- `url` – the URL to access.
- `options` – optional parameters: method, headers etc.

Without `options`, this is a simple GET request, downloading the contents of the `url`.

# Fetch, fetch, fetch



# Fetch



```

/*
Fetches json data from the file persons.json
*/
fetch('json/persons.json')
  .then(function (response) {
    return response.json();
  })
  .then(function (jsonData) {
    console.log(jsonData);
    appendPersons(jsonData)
  });

/*
Appends json data to the DOM
*/
function appendPersons(persons) {
  let htmlTemplate = "";
  for (let person of persons) {
    htmlTemplate += /*html*/
      `


        <h4>${person.name}</h4>
        <p>${person.age} years old</p>
        <p>Hair color: ${person.hairColor}</p>
        <p>Relation: ${person.relation}</p>
      </article>`;
  }
  document.querySelector("#persons").innerHTML = htmlTemplate;
}


```

```

[
  {
    "name": "Peter Madsen",
    "age": 52,
    "hairColor": "blonde",
    "relation": "dad",
    "img": "img/dad.jpg"
  },
  {
    "name": "Ane Madsen",
    "age": 51,
    "hairColor": "brown",
    "relation": "mom",
    "img": "img/ane.jpg"
  },
  {
    "name": "Rasmus Madsen",
    "age": 28,
    "hairColor": "blonde",
    "relation": "brother",
    "img": "img/IMG_0526_kvadrat.jpg"
  },
  {
    "name": "Mie Madsen",
    "age": 25,
    "hairColor": "brown",
    "relation": "blonde",
    "img": "img/mie.jpg"
  },
  {
    "name": "Mads Madsen",
    "age": 18,
    "hairColor": "dark",
    "relation": "blonde",
    "img": "img/mads.jpg"
  },
  {
    "name": "Jens Madsen",
    "age": 14,
    "hairColor": "blonde",
    "relation": "uncle",
    "img": "img/jenspeter.jpg"
  }
]

```

```

/*
Fetches json data from the file persons.json
*/
fetch('json/persons.json')
  .then(function (response) {
    return response.json();
  })
  .then(function (jsonData) {
    console.log(jsonData);
    appendPersons(jsonData);
  });

/*
Appends json data to the DOM
*/
function appendPersons(persons) {
  let htmlTemplate = "";
  for (let person of persons) {
    htmlTemplate += /*html*/
      <article>
        
        <h4>${person.name}</h4>
        <p>${person.age} years old</p>
        <p>Hair color: ${person.hairColor}</p>
        <p>Relation: ${person.relation}</p>
      </article>
  }
  document.querySelector("#persons").innerHTML = htmlTemplate;
}

```

REQUEST (fetch)

RESPONSE (then)

```

[
  {
    "name": "Peter Madsen",
    "age": 52,
    "hairColor": "blonde",
    "relation": "dad",
    "img": "img/dad.jpg"
  },
  {
    "name": "Ane Madsen",
    "age": 51,
    "hairColor": "brown",
    "relation": "mom",
    "img": "img/ane.jpg"
  },
  {
    "name": "Rasmus Madsen",
    "age": 28,
    "hairColor": "blonde",
    "relation": "brother",
    "img": "img/IMG_0526_kvadrat.jpg"
  },
  {
    "name": "Mie Madsen",
    "age": 25,
    "hairColor": "brown",
    "relation": "blonde",
    "img": "img/mie.jpg"
  },
  {
    "name": "Mads Madsen",
    "age": 18,
    "hairColor": "dark",
    "relation": "blonde",
    "img": "img/mads.jpg"
  },
  {
    "name": "Jens Madsen",
    "age": 14,
    "hairColor": "blonde",
    "relation": "uncle",
    "img": "img/jenspeter.jpg"
  }
]

```

fetch-family-members

```
/*
Fetches json data from the file persons.json
*/
fetch('json/persons.json')
  .then(function (response) {
    return response.json();
  })
  .then(function (jsonData) {
    console.log(jsonData);
    appendPersons(jsonData)
  });

/*
Appends json data to the DOM
*/
function appendPersons(persons) {
  let htmlTemplate = "";
  for (let person of persons) {
    htmlTemplate += /*html*/
      `


        <h4>${person.name}</h4>
        <p>${person.age} years old</p>
        <p>Hair color: ${person.hairColor}</p>
        <p>Relation: ${person.relation}</p>

`;
  }
  document.querySelector("#persons").innerHTML = htmlTemplate;
}
```

```
},
{
  "name": "Rasmus Madsen",
  "age": 28,
  "hairColor": "blonde",
  "relation": "brother",
  "img": "img/IMG_0526_kvadrat.jpg"
},
{
  "name": "Mie Madsen",
  "age": 25,
  "hairColor": "brown",
  "relation": "blonde",
  "img": "img/mie.jpg"
},
{
  "name": "Mads Madsen",
  "age": 18,
  "hairColor": "dark",
  "relation": "blonde",
  "img": "img/mads.jpg"
},
{
```

## request (fetch)

The diagram illustrates the process of making an API request using the `fetch` function in JavaScript. On the left, a screenshot of a code editor shows the `app.js` file. A specific line of code is highlighted: `const response = await fetch("https://raw.githubusercontent.com/cederdorff/web-diplom-frontend/main/_data/persons.json")`. An arrow points from this line to the URL in the browser's address bar on the right. The browser window displays the JSON response, which is an array of objects representing person data. Each object contains properties for name, mail, title, and img.

```
JS app.js    x
fetch-persons-grid > JS app.js > ...
1  let persons = []; // global variable
2
3  async function getPerson() {
4      const response = await fetch(
5          "https://raw.githubusercontent.com/cederdorff/web-diplom-frontend/main/_data/persons.json"
6      ); // fetch request - fetch data from a given url
7      persons = await response.json(); // setting global variable with fetched data
8      displayPersons(persons); // calling displayPersons with persons as parameter
9  }
10
11 function displayPersons(listOfPersons) {
12     let html = ""; // variable to store html
13     //loop through all persons and create an article with content for each
14     for (const person of listOfPersons) {
15         html += /*html*/
16             `

17                 
18                 <h2>${person.name}</h2>
19                 <p>${person.title}</p>
20                 <a href="mailto:${person.mail}">${person.mail}</a>
21             </article>
22         `; // generate and save html for every person in html variable
23     }
24     // set grid container content with person <article> elements
25     // saved in html
26     document.querySelector("#content").innerHTML = html;
27 }
28
29 getPerson(); // execute get persons to start the fun
30


```

Raw Parsed

```
[{"name": "Birgitte Kirk Iversen", "mail": "bkj@mail.dk", "title": "Senior Lecturer", "img": "https://www.eaaa.dk/media/u4gorzsd/birgitte-kirk-iversen2.jpg?width=800&height=450"}, {"name": "Martin Aagaard N\u00f8hr", "mail": "mnor@mail.dk", "title": "Lecturer", "img": "https://www.eaaa.dk/media/oayjq02h/martin-n%C3%B8hr.jpg?width=800&height=450"}, {"name": "Rasmus Cederdorff", "mail": "race@mail.dk", "title": "Senior Lecturer", "img": "https://www.eaaa.dk/media/devlvgj/rasmus-cederdorff.jpg?width=800&height=450"}, {"name": "Dan Okkels Brendstrup", "mail": "dob@mail.dk", "title": "Lecturer", "img": "https://www.eaaa.dk/media/bdojel41/dan-okkels-brendstrup.jpg?width=800&height=450"}, {"name": "Line Skj\u00f8dt", "mail": "lskj@mail.dk", "title": "Senior Lecturer & Internship Coordinator", "img": "https://www.eaaa.dk/media/14qpfeq4/line-skj%C3%B8dt.jpg?width=800&height=450"}, {"name": "Kasper Fischer Topp", "mail": "kato@mail.dk", "title": "Lecturer", "img": "https://www.eaaa.dk/media/lxzcybme/kasper-topp.jpg?width=800&height=450"}, {"name": "Anne Kirketerp", "mail": "anki@mail.dk", "title": "", "img": ""}]
```

fetch-persons-grid

## request (fetch)

```
app.js — web-diplom-frontend
JS app.js  X
fetch-persons-grid > JS app.js > ...
1 let persons = [] // global variable
2
3 async function getPerson() {
4     const response = await fetch(
5         "https://raw.githubusercontent.com/cederdorff/web-diplom-frontend/main/_data/persons.json"
6     ); // fetch request - fetch data from a given url
7     persons = await response.json(); // setting global variable with fetched data
8     displayPersons(persons); // calling displayPersons with persons as parameter
9 }
10
11 function displayPersons(listOfPersons) {
12     let html = ""; // variable to store html
13     //loop through all persons and create an article with content for each
14     for (const person of listOfPersons) {
15         html += /*html*/
16             `

17                 
18                 <h2>${person.name}</h2>
19                 <p>${person.title}</p>
20                 <a href="mailto:${person.mail}">${person.mail}</a>
21             </article>
22         `; // generate and save html for every person in html variable
23     }
24     // set grid container content with person <article> elements
25     // saved in html
26     document.querySelector("#content").innerHTML = html;
27 }
28
29 getPerson(); // execute get persons to start the fun
30


```

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

## response (JSON)

```
https://raw.githubusercontent.com/cederdorff/web-diplom-frontend/main/_data/persons.json
raw.githubusercontent.com/cederdorff/web-diplom-frontend/main/_data/persons.json
[{"name": "Birgitte Kirk Iversen", "mail": "bkj@mail.dk", "title": "Senior Lecturer", "img": "https://www.eaaa.dk/media/u4gorzsd/birgitte-kirk-iversen2.jpg?width=800&height=450"}, {"name": "Martin Aagaard N\u00f8hr", "mail": "mnor@mail.dk", "title": "Lecturer", "img": "https://www.eaaa.dk/media/oayjq02h/martin-n%C3%B8hr.jpg?width=800&height=450"}, {"name": "Rasmus Cederdorff", "mail": "race@mail.dk", "title": "Senior Lecturer", "img": "https://www.eaaa.dk/media/devlvgj/rasmus-cederdorff.jpg?width=800&height=450"}, {"name": "Dan Okkels Brendstrup", "mail": "dob@mail.dk", "title": "Lecturer", "img": "https://www.eaaa.dk/media/bdojel41/dan-okkels-brendstrup.jpg?width=800&height=450"}, {"name": "Line Skj\u00f8dt", "mail": "lskj@mail.dk", "title": "Senior Lecturer & Internship Coordinator", "img": "https://www.eaaa.dk/media/14qpfeq4/line-skj%C3%B8dt.jpg?width=800&height=450"}, {"name": "Kasper Fischer Topp", "mail": "kato@mail.dk", "title": "Lecturer", "img": "https://www.eaaa.dk/media/lxzcybme/kasper-topp.jpg?width=800&height=450"}, {"name": "Anne Kirketerp", "mail": "anki@mail.dk", "title": "", "img": ""}]
```

The diagram illustrates the flow of data from an external JSON file to a JavaScript application. On the left, a code editor shows the `app.js` file with a fetch request to `https://raw.githubusercontent.com/cederdorff/web-diplom-frontend/main/_data/persons.json`. On the right, a browser window displays the parsed JSON data, which is then mapped back to the `app.js` code via arrows.

```
app.js — web-diplom-frontend
JS app.js
fetch-persons-grid > JS app.js > ...
1 let persons = [] // global variable
2
3 async function getPerson() {
4     const response = await fetch(
5         "https://raw.githubusercontent.com/cederdorff/web-diplom-frontend/main/_data/persons.json"
6     ); // fetch request - fetch data from a given url
7     persons = await response.json(); // setting global variable with fetched data
8     displayPersons(persons); // calling displayPersons with persons as parameter
9 }
10
11 function displayPersons(listOfPersons) {
12     let html = ""; // variable to store html
13     //loop through all persons and create an article with content for each
14     for (const person of listOfPersons) {
15         html += /*html*/
16             `

17                 
18                 <h2>${person.name}</h2>
19                 <p>${person.title}</p>
20                 <a href="mailto:${person.mail}">${person.mail}</a>
21             </article>
22         `; // generate and save html for every person in html variable
23     }
24     // set grid container content with person <article> elements
25     // saved in html
26     document.querySelector("#content").innerHTML = html;
27 }
28
29 getPerson(); // execute get persons to start the fun
30


```

Raw Parsed

```
[{"name": "Birgitte Kirk Iversen", "mail": "bki@mail.dk", "title": "Senior Lecturer", "img": "https://www.eaaa.dk/media/u4gorzsd/birgitte-kirk-iversen2.jpg?width=800&height=450"}, {"name": "Martin Aagaard N\u00f8hr", "mail": "mnor@mail.dk", "title": "Lecturer", "img": "https://www.eaaa.dk/media/oayjq02h/martin-n%C3%B8hr.jpg?width=800&height=450"}, {"name": "Rasmus Cederdorff", "mail": "race@mail.dk", "title": "Senior Lecturer", "img": "https://www.eaaa.dk/media/devlvgj/rasmus-cederdorff.jpg?width=800&height=450"}, {"name": "Dan Okkels Brendstrup", "mail": "dob@mail.dk", "title": "Lecturer", "img": "https://www.eaaa.dk/media/bdojel41/dan-okkels-brendstrup.jpg?width=800&height=450"}, {"name": "Line Skj\u00f8dt", "mail": "lskj@mail.dk", "title": "Senior Lecturer & Internship Coordinator", "img": "https://www.eaaa.dk/media/14qpfeq4/line-skj%C3%B8dt.jpg?width=800&height=450"}, {"name": "Kasper Fischer Topp", "mail": "kato@mail.dk", "title": "Lecturer", "img": "https://www.eaaa.dk/media/lxzcybme/kasper-topp.jpg?width=800&height=450"}, {"name": "Anne Kirketerp", "mail": "anki@mail.dk", "title": "", "img": ""}]
```

fetch-persons-grid

index.html — web-diplom-frontend

app.js

```
fetch-persons-grid > JS app.js > ...
1 let persons = [] // global variable
2
3 async function getPerson() {
4     const response = await fetch(
5         "https://raw.githubusercontent.com/cederdorff/web-diplom-frontend/main/_data/persons.json"
6     ); // fetch request - fetch data from a given url
7     persons = await response.json(); // setting global variable with fetched data
8     displayPersons(persons); // calling displayPersons with persons as parameter
9 }
10
11 function displayPersons(listOfPersons) {
12     let html = ""; // variable to store html
13     //loop through all persons and create an article with content for each
14     for (const person of listOfPersons) {
15         html += /*html*/ `
16             <article>
17                 
18                 <h2>${person.name}</h2>
19                 <p>${person.title}</p>
20                 <a href="mailto:${person.mail}">${person.mail}</a>
21             </article>
22         `; // generate and save html for every person in html variable
23     }
24     // set grid container content with person <article> elements
25     // saved in html
26     document.querySelector("#content").innerHTML = html;
27 }
28
29 getPerson(); // execute get persons to start the fun
30
```

index.html

```
fetch-persons-grid > index.html > HTML > body > main > section#content.grid-container
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5     <meta charset="UTF-8">
6     <meta http-equiv="X-UA-Compatible" content="IE=edge">
7     <meta name="viewport" content="width=device-width, initial-scale=1.0">
8     <link rel="stylesheet" href="app.css">
9     <title>Fetch Persons</title>
10
11 <body>
12     <header>
13         <h1>Fetch Persons</h1>
14     </header>
15     <main>
16         <section id="content" class="grid-container"></section>
17     </main>
18     <script src="app.js"></script>
19
20 </body>
21
22 </html>
```

DOM Manipulation

Ln 17, Col 9 Spaces: 4 UTF-8 LF HTML Go Live Prettier

fetch-persons-grid

# Fetch

... get & post data from and to a data source

```
// Simple javascript 😒  
  
//Synchronous fetch using async/await.  
  
// Usual way  
✓ const jsonData = fetch('URL')  
    .then(response => response.json())  
    .then(json => console.log(json));  
  
// Using await  
✓ const jsonData = await fetch('URL').then(res => res.json())  
  
// Shorter syntax 😊  
✓ const jsonData = await (await fetch('URL')).json();
```

<https://www.instagram.com/p/B0nxQjXj9Zi/>

# Async JS

JavaScript reads and runs the script from top to bottom.

JavaScript functions are executed in the sequence they are called. Not in the sequence they are defined.

... by default JavaScript is synchronous.

# With callbacks, we can make JS Asynchronous

```
setTimeout(() => {
  console.log("Hey, I'm async!");
}, 3000);

btn.addEventListener('click', () => {
  alert("Hey, you clicked me!");
});
```

```
// fetch with callbacks
fetch("https://cederdorff.github.io/web-frontend/canvas")
  .then(function (response) {
    return response.json();
  })
  .then(function (data) {
    console.log(data);
  });
});
```

# Fetch is Asynchronous

And it's about making HTTP requests in JavaScript.  
... and a way to get & post data from and to a data source.

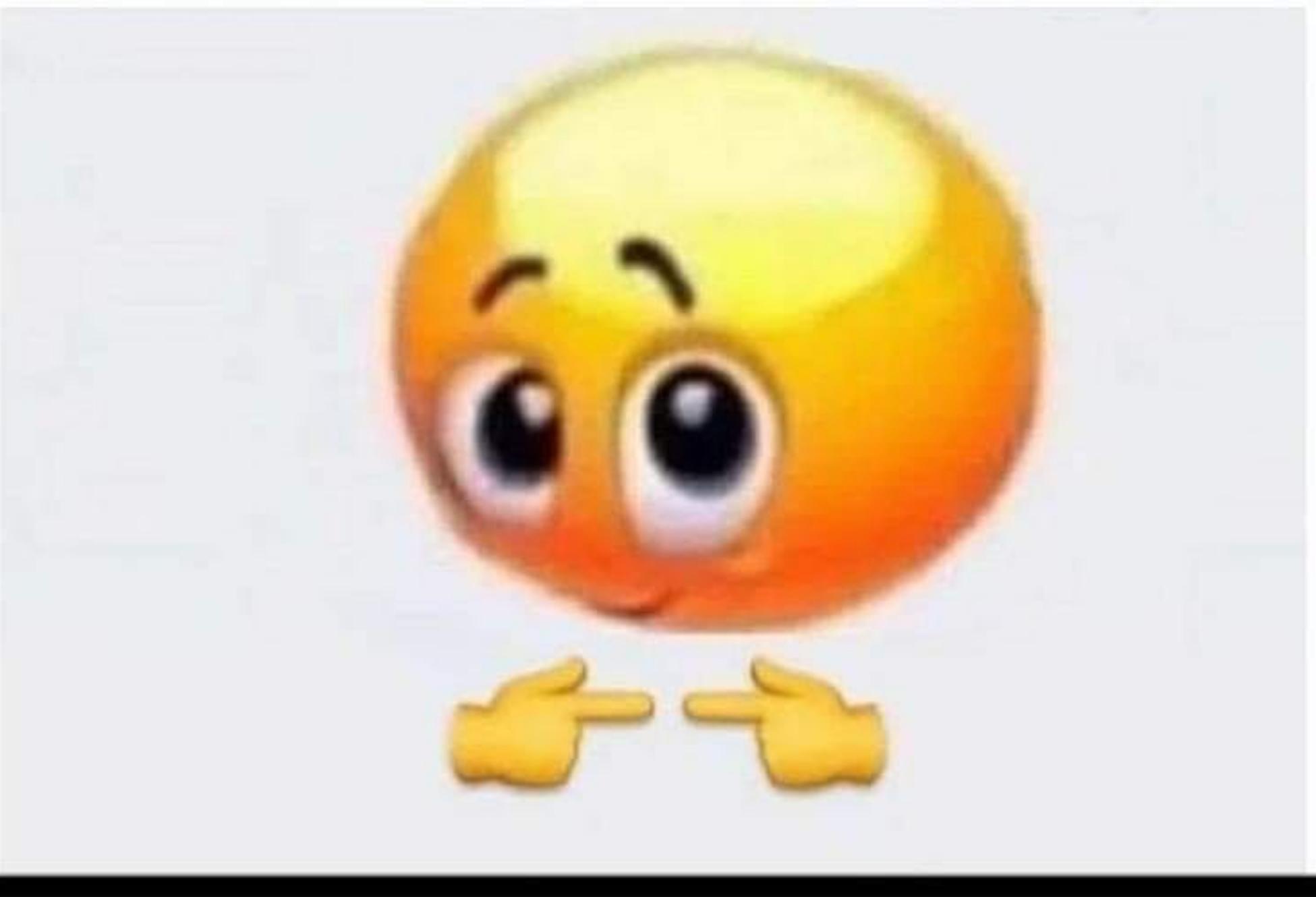
```
// fetch with callbacks
fetch("https://cederdorff.github.io/web-frontend/canvas-users/data.json")
  .then(function (response) {
    return response.json();
  })
  .then(function (data) {
    console.log(data);
  });

```

# Fetch: callback (then) vs async/await

```
// fetch with callbacks
fetch("https://cederdorff.github.io/web-frontend/canvas-users/data.json")
  .then(function (response) {
    return response.json();
  })
  .then(function (data) {
    console.log(data);
  });
// 
// 
// or with async/await
const response = await fetch("https://cederdorff.github.io/web-frontend/canvas-users/data.json");
const data = await response.json();
console.log(data);
```

will you be async to my await?



# async & await

- Use await to tell JS to wait for a fetch call to finish and to wait for JSON to parse.
- When using await you must tell JS that inside of the function goes some asynchronous code by wrapping it in an async function.

```
async function getPosts() {  
  const url = "https://raw.githubusercontent.com/.../main/posts.json";  
  const response = await fetch(url);  
  const data = await response.json();  
  setPosts(data);  
}  
  
getPosts();
```

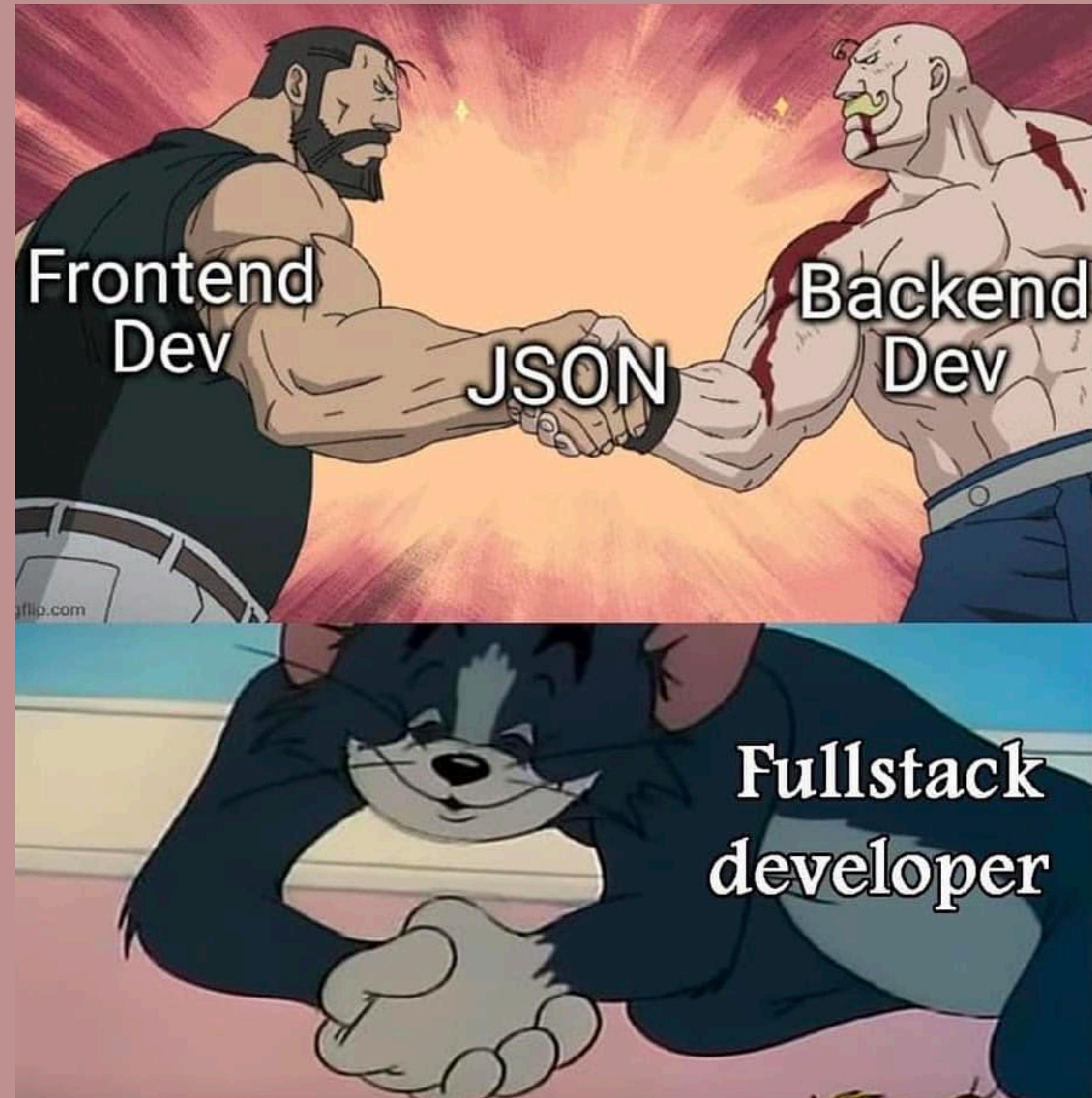
# JSON

JavaScript Object Notation

# JSON

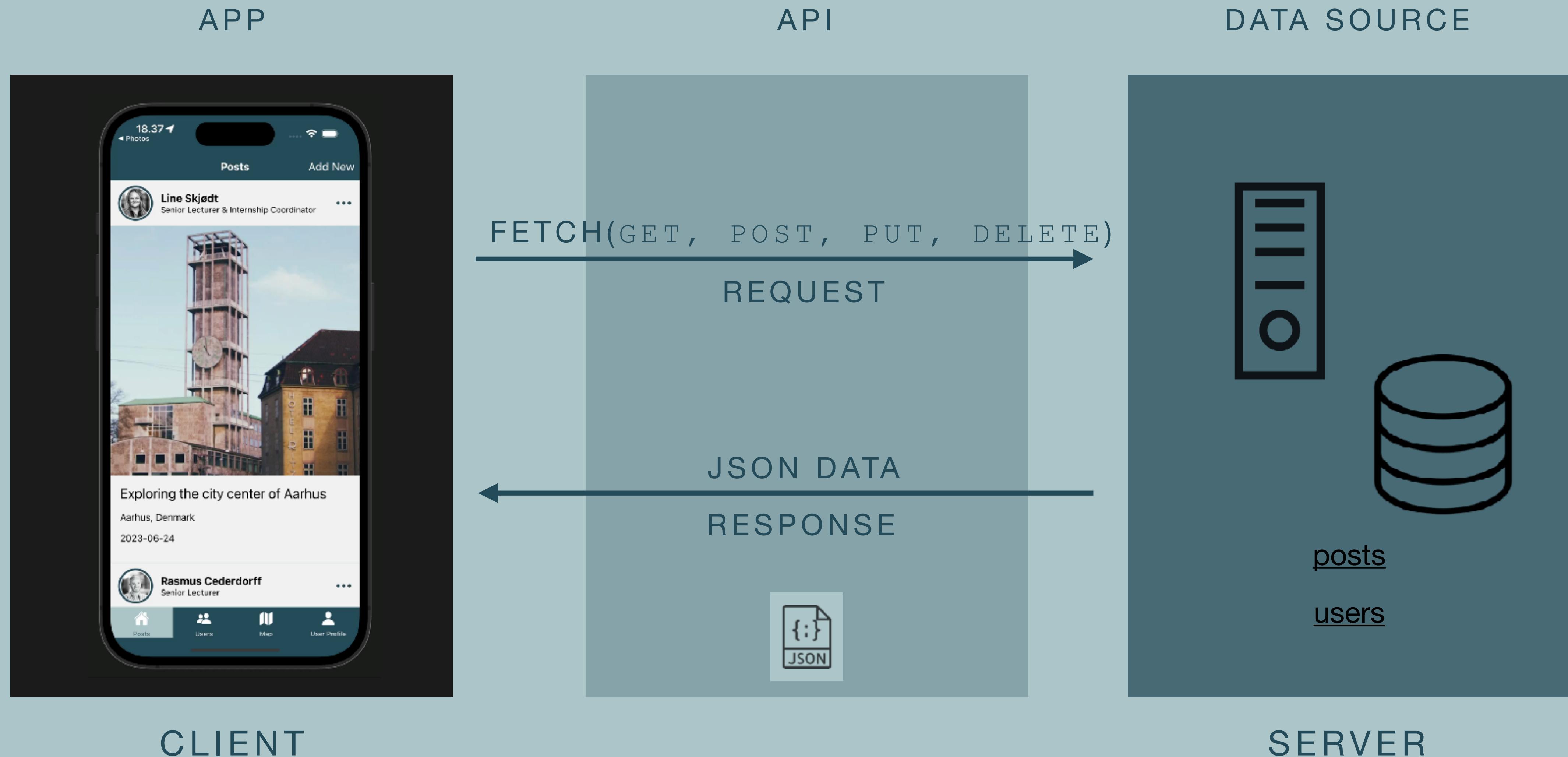
## JavaScript Object Notation

... a syntax for storing & exchanging data  
over the web



<https://www.instagram.com/p/CVqbCzgsZUF/>

# JSON (& API) is the glue



# JSON

... a syntax for storing and exchanging data over the web

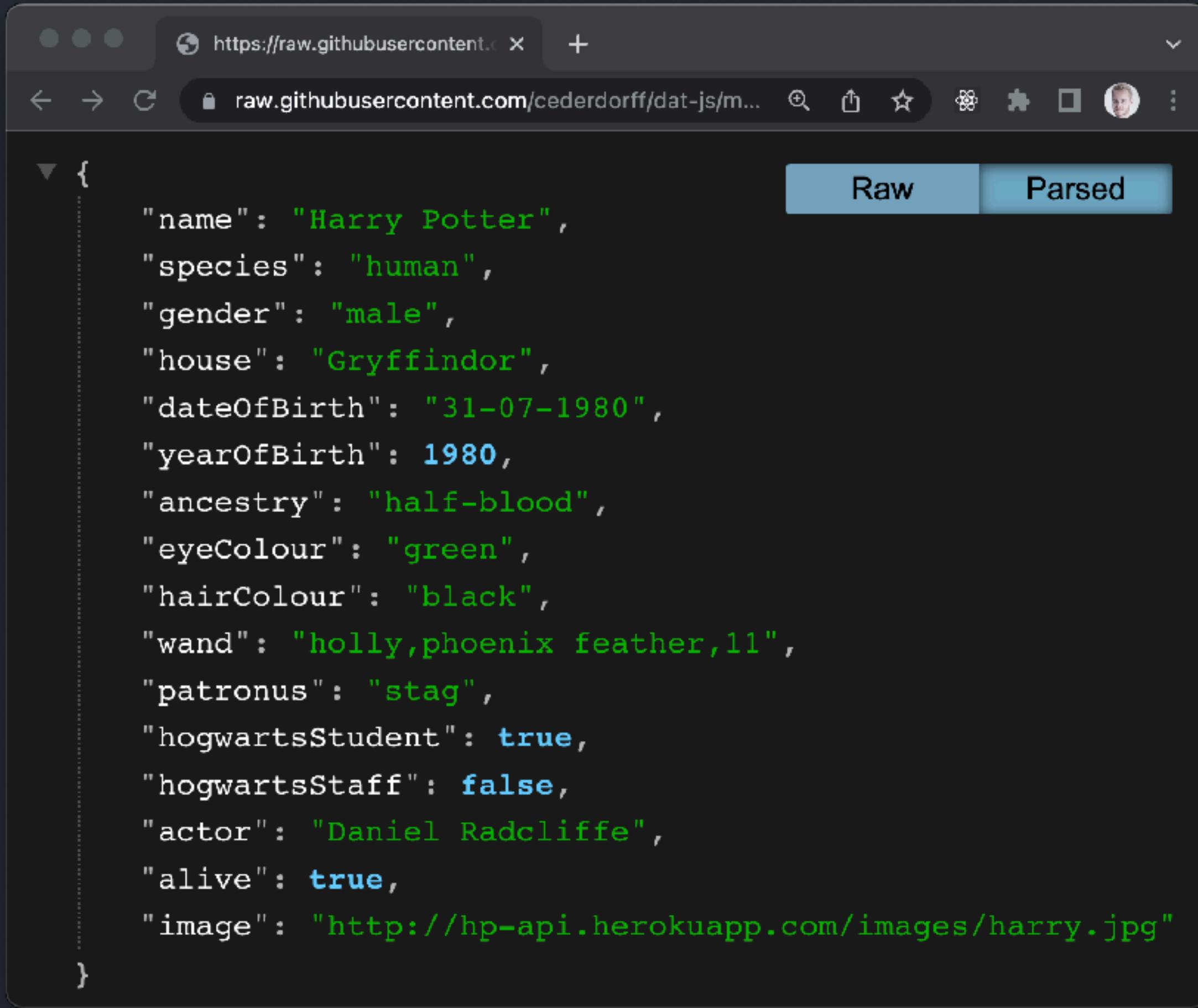
```
{  
  "name": "Alicia",  
  "age": 6  
}
```

JSON OBJECT

```
[{  
  "name": "Alicia",  
  "age": 6  
, {  
  "name": "Peter",  
  "age": 22  
}]
```

LIST OF JSON OBJECTS

# JSON

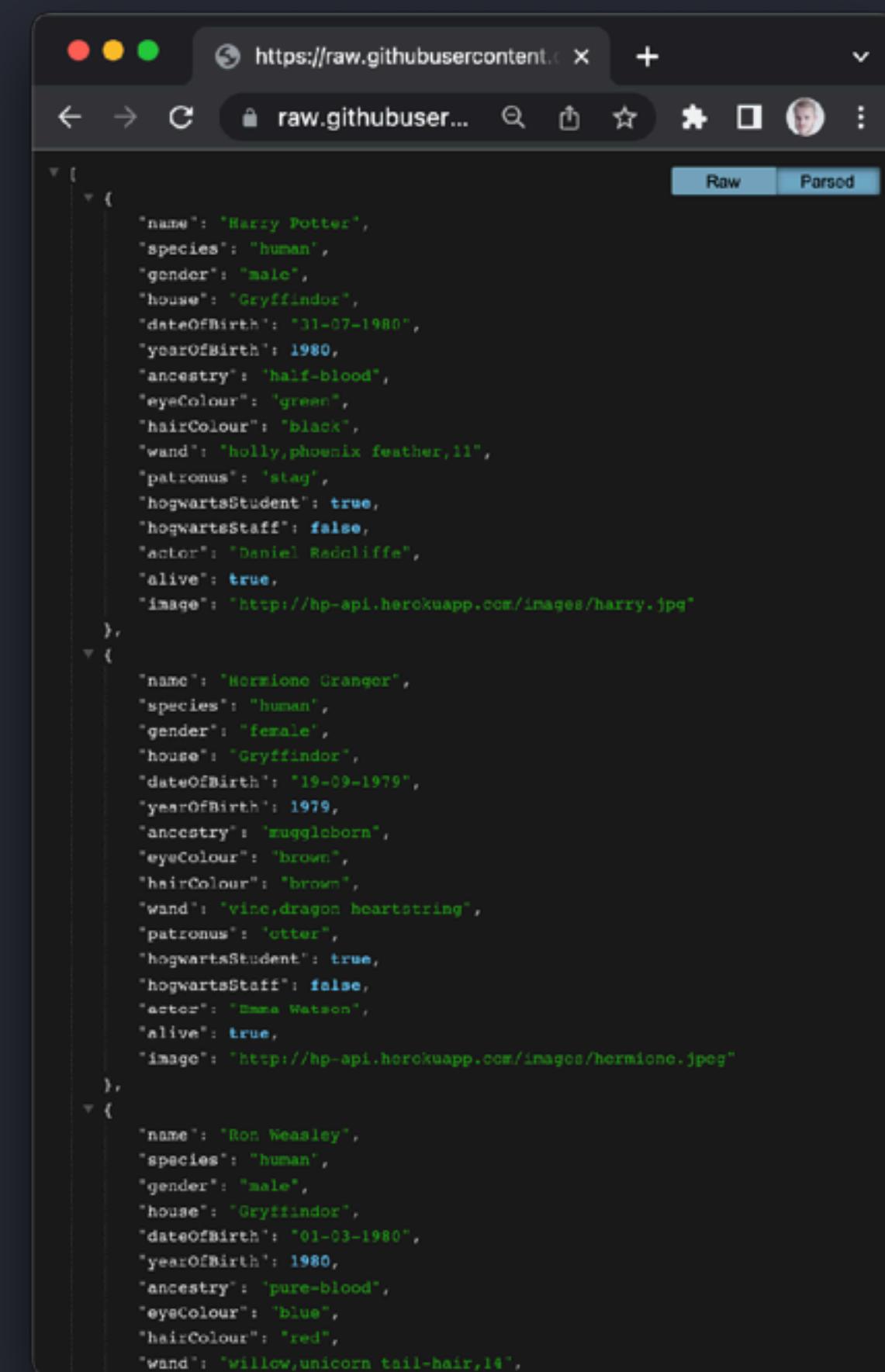


A screenshot of a web browser displaying a single JSON object. The URL is <https://raw.githubusercontent.com/cederdorff/dat-js/master/data/harry.json>. The JSON structure is as follows:

```
{  
  "name": "Harry Potter",  
  "species": "human",  
  "gender": "male",  
  "house": "Gryffindor",  
  "dateOfBirth": "31-07-1980",  
  "yearOfBirth": 1980,  
  "ancestry": "half-blood",  
  "eyeColour": "green",  
  "hairColour": "black",  
  "wand": "holly,phoenix feather,11",  
  "patronus": "stag",  
  "hogwartsStudent": true,  
  "hogwartsStaff": false,  
  "actor": "Daniel Radcliffe",  
  "alive": true,  
  "image": "http://hp-api.herokuapp.com/images/harry.jpg"  
}
```

The "Raw" tab is selected at the bottom.

JSON OBJECT



A screenshot of a web browser displaying a list of three JSON objects. The URL is <https://raw.githubusercontent.com/cederdorff/dat-js/master/data/characters.json>. The JSON structure is as follows:

```
[  
  {  
    "name": "Harry Potter",  
    "species": "human",  
    "gender": "male",  
    "house": "Gryffindor",  
    "dateOfBirth": "31-07-1980",  
    "yearOfBirth": 1980,  
    "ancestry": "half-blood",  
    "eyeColour": "green",  
    "hairColour": "black",  
    "wand": "holly,phoenix feather,11",  
    "patronus": "stag",  
    "hogwartsStudent": true,  
    "hogwartsStaff": false,  
    "actor": "Daniel Radcliffe",  
    "alive": true,  
    "image": "http://hp-api.herokuapp.com/images/harry.jpg"  
  },  
  {  
    "name": "Hermione Granger",  
    "species": "human",  
    "gender": "female",  
    "house": "Gryffindor",  
    "dateOfBirth": "19-09-1979",  
    "yearOfBirth": 1979,  
    "ancestry": "muggleborn",  
    "eyeColour": "brown",  
    "hairColour": "brown",  
    "wand": "vine,dragon heartstring",  
    "patronus": "otter",  
    "hogwartsStudent": true,  
    "hogwartsStaff": false,  
    "actor": "Emma Watson",  
    "alive": true,  
    "image": "http://hp-api.herokuapp.com/images/hermione.jpg"  
  },  
  {  
    "name": "Ron Weasley",  
    "species": "human",  
    "gender": "male",  
    "house": "Gryffindor",  
    "dateOfBirth": "01-03-1980",  
    "yearOfBirth": 1980,  
    "ancestry": "pure-blood",  
    "eyeColour": "blue",  
    "hairColour": "red",  
    "wand": "willow,unicorn tail-hair,14",  
  }  
]
```

The "Parsed" tab is selected at the bottom.

LIST OF JSON OBJECTS

# JSON Object

The diagram illustrates the connection between a JSON object and a web application displaying Harry Potter characters.

**Left Side: Web Application**

A screenshot of a web browser window titled "Harry Potter Characters". The page displays three cards:

- Harry Potter**: Shows a portrait of Harry Potter and is labeled "Gryffindor".
- Hermione Granger**: Shows a portrait of Hermione Granger and is labeled "Gryffindor".
- Ron Weasley**: Shows a portrait of Ron Weasley and is labeled "Gryffindor".

**Right Side: JSON Object**

A screenshot of a web browser window showing the JSON representation of Harry Potter's character. The JSON is displayed in a collapsible tree view with "Raw" and "Parsed" tabs. The "Parsed" tab shows the following data:

```
{
  "name": "Harry Potter",
  "species": "human",
  "gender": "male",
  "house": "Gryffindor",
  "dateOfBirth": "31-07-1980",
  "yearOfBirth": 1980,
  "ancestry": "half-blood",
  "eyeColour": "green",
  "hairColour": "black",
  "wand": "holly,phoenix feather,11",
  "patronus": "stag",
  "hogwartsStudent": true,
  "hogwartsStaff": false,
  "actor": "Daniel Radcliffe",
  "alive": true,
  "image": "http://hp-api.herokuapp.com/images/harry.jpg"
}
```

Red arrows point from the character cards in the application to the corresponding JSON properties in the JSON object, illustrating how the application maps the JSON data to its user interface.

<https://raw.githubusercontent.com/cederdorff/dat-js/main/data/harry.json>

# JSON Object

The diagram illustrates the relationship between a JSON object and its usage in a JavaScript application.

**JavaScript Code (app.js):**

```
28
29  function showCharacter(character) {
30      document.querySelector("#characters").insertAdjacentHTML(
31          "beforeend",
32          /*html*/
33          `

34              
35              <h2>${character.name}</h2>
36              <p>${character.house}</p>
37          </article>
38      `);
39  };
40 }


```

**JSON Object (Raw Data):**

```
{
    "name": "Harry Potter",
    "species": "human",
    "gender": "male",
    "house": "Gryffindor",
    "dateOfBirth": "31-07-1980",
    "yearOfBirth": 1980,
    "ancestry": "half-blood",
    "eyeColour": "green",
    "hairColour": "black",
    "wand": "holly,phoenix feather,11",
    "patronus": "stag",
    "hogwartsStudent": true,
    "hogwartsStaff": false,
    "actor": "Daniel Radcliffe",
    "alive": true,
    "image": "http://hp-api.herokuapp.com/images/harry.jpg"
}
```

A large red arrow points from the JSON object on the right to the `character` variable in the `showCharacter` function on the left, indicating that the JSON object is being passed as a parameter to the function and used to dynamically generate HTML.

<https://raw.githubusercontent.com/cederdorff/dat-js/main/data/harry.json>

# JAVASCRIPT OBJECT NOTATION

- Collection of key-value pair: “key” : “value”
  - List of values, collections or objects
  - Lightweight data-interchange format
  - Syntax / text format for storing and exchanging data over the web
  - Human and machine readable **text**: small, fast and simple
  - Language independent
  - Can be parsed directly to JavaScript Object
  - JavaScript Objects can be converted directly to JSON
  - The glue between programs (interface between frontend and backend)

```
{  
    "id": "1",  
    "firstname": "Kasper",  
    "lastname": "Topp",  
    "age": "34",  
    "haircolor": "Dark Blonde",  
    "countryName": "Denmark",  
    "gender": "Male",  
    "lookingFor": "Female"  
},  
{  
    "id": "2",  
    "firstname": "Nicklas",  
    "lastname": "Andersen",  
    "age": "22",  
    "haircolor": "Brown",  
    "countryName": "Denmark",  
    "gender": "Male",  
    "lookingFor": "Female"  
},  
{  
    "id": "3",  
    "firstname": " Sarah",  
    "lastname": "Dybvad ",  
    "age": "34",  
    "haircolor": "Blonde",  
    "countryName": "Denmark",  
    "gender": "Female",  
    "lookingFor": "Male"  
},  
{  
    "id": "4",  
    "firstname": "Alex"
```

# JSON METHODS

```
const user = {  
    name: "John",  
    age: 30,  
    gender: "male",  
    lookingFor: "female"  
};  
  
// === JSON.stringify === //  
const jsonUser = JSON.stringify(user);  
console.log(jsonUser); // {"name": "John", "age": 30, "gender": "male", "lookingFor": "female"}  
  
// === JSON.parse === //  
const jsonString = '{"name": "John", "age": 30, "gender": "male", "lookingFor": "female"}';  
const userObject = JSON.parse(jsonString);  
console.log(userObject); // logging userObject
```

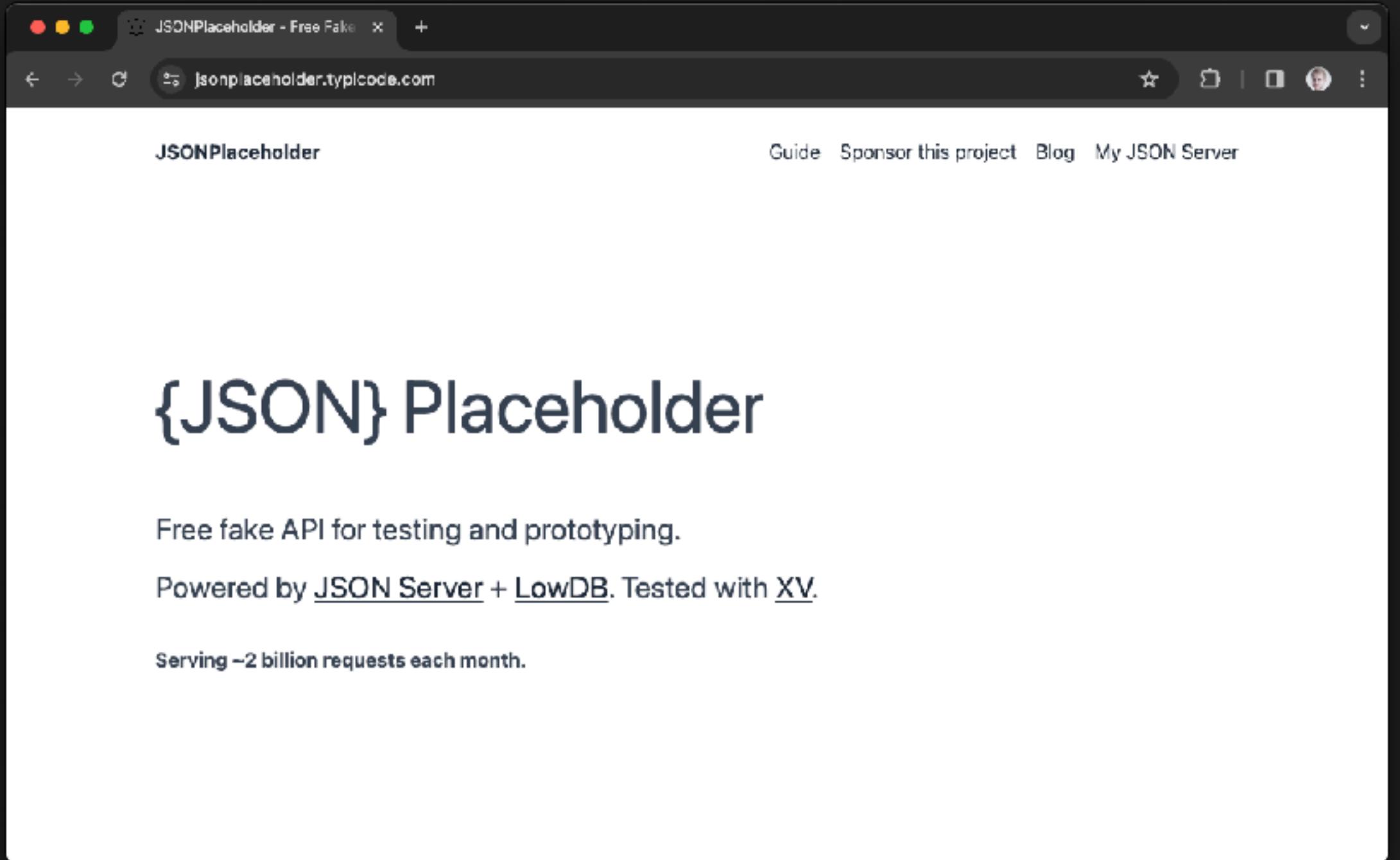
# CRUD App

 <p><b>Rasmus Cederdorff</b></p> <p>Senior Lecturer <a href="mailto:race@dev.dk">race@dev.dk</a></p> <p><b>UPDATE</b> <b>DELETE</b></p>	 <p><b>Lars Bogetoft</b></p> <p>Head of Education <a href="mailto:larb@eaaa.dk">larb@eaaa.dk</a></p> <p><b>UPDATE</b> <b>DELETE</b></p>	 <p><b>Edith Terte</b></p> <p>Lecturer <a href="mailto:edan@kea.dk">edan@kea.dk</a></p> <p><b>UPDATE</b> <b>DELETE</b></p>	 <p><b>Frederikke Bender</b></p> <p>Head of Education <a href="mailto:fbe@kea.dk">fbe@kea.dk</a></p> <p><b>UPDATE</b> <b>DELETE</b></p>	 <p><b>Murat Kilic</b></p> <p>Senior Lecturer <a href="mailto:mki@eaaa.dk">mki@eaaa.dk</a></p> <p><b>UPDATE</b> <b>DELETE</b></p>	 <p><b>Anne Andersen</b></p> <p>Head of Education <a href="mailto:anki@mail.dk">anki@mail.dk</a></p> <p><b>UPDATE</b> <b>DELETE</b></p>
--	---	--	---	--	--

The Network tab of the browser developer tools shows a list of requests. The most recent request is a GET to '/users', which returned a JSON response containing the following data:

```
[{"id": 2, "name": "Rasmus Cederdorff", "mail": "race@dev.dk", "title": "Senior Lecturer", "image": "https://share.cederdorff.com/images/race.jpg"}, {"id": 3, "name": "Lars Bogetoft", "mail": "larb@eaaa.dk", "title": "Head of Education", "image": "https://kea.dk/slir/w200-c1x1/images/user-profile/chefer/larb.jpg"}, {"id": 4, "name": "Edith Terte", "mail": "edan@kea.dk", "title": "Lecturer", "image": "https://kea.dk/slir/w180-c1x1/images/user-profile/synced/EDAN.jpg"}, {"id": 5, "name": "Frederikke Bender", "mail": "fbe@kea.dk", "title": "Head of Education", "image": "https://kea.dk/slir/w200-c1x1/images/user-profile/chefer/fbe.jpg"}, {"id": 6, "name": "Murat Kilic", "mail": "mki@eaaa.dk", "title": "Senior Lecturer", "image": "https://www.eaaa.dk/media/llyavasj/murat-kilic.jpg?width=800&height=800"}, {"id": 7, "name": "Anne Andersen", "mail": "anki@mail.dk", "title": "Head of Education", "image": "https://www.eaaa.dk/media/5kuh1xeo/anne-kirketerp.jpg?width=800&height=800"}]
```

# Test endpoints



<https://jsonplaceholder.typicode.com/>

- Install [JSON Formatter](#)
- Test the following endpoints:
  - <https://jsonplaceholder.typicode.com/posts/>
  - <https://jsonplaceholder.typicode.com/posts/5>
  - <https://jsonplaceholder.typicode.com/users/>
  - <https://jsonplaceholder.typicode.com/users/1>
  - <https://jsonplaceholder.typicode.com/todos>
  - <https://jsonplaceholder.typicode.com/todos/5>

```
{  
  "name": "Rasmus Cederdorff",  
  "birthday": "1990-03-12",  
  "title": "Experienced JavaScript Developer",  
  "experienceYears": 10,  
  "education": {  
    "degree": "Master of Science in IT - Web Communication",  
    "specialization": "Web Architecture"  
  },  
  "skills": ["JavaScript", "React", "Node.js", "UI/UX Design"],  
  "currentPosition": "Senior Lecturer at EAAA",  
  "teachingSubjects": ["Web Development", "JavaScript", "React", "BaaS & Node.js"],  
  "lovesJavaScript": true  
}
```

# Define yourself with JSON

- <https://race.notion.site/Define-yourself-with-JSON-391944b6b1a041bfa341a0d46f5c0e4a?pvs=4>

# Array Methods

Array methods are built-in JavaScript functions that make it easier to work with lists of data.

# Array Methods

Array methods are built-in JavaScript functions that make it easier to work with lists of data.

They can be used to:

- Loop through all items  
(`.forEach`, `.map`)
- Filter items based on conditions  
(`.filter`)
- Search or find specific items  
(`.find`, `.some`, `.every`)
- Sort or reorder data  
(`.sort`, `.reverse`)

# Array Methods to know

.push (...)

.map (...)

.filter (...)

.find (...)

.sort (...)

# Computer science student



## Senior developer, 10+ years experience



<https://www.instagram.com/p/BxWAgatgSmn/>

# Array methods

<https://javascript.info/array-methods#filter>

- Chapter
- Data types
- Lesson navigation
- Add/remove items
- Iterate: forEach
- Searching in array**
- Transform an array
- Array.isArray
- Most methods support "thisArg"
- Summary
- Tasks (13)
- Comments
- Share
- <
- [Edit on GitHub](#)
- Ads

**filter**

The `find` method looks for a single (first) element that makes the function return `true`. If there may be many, we can use `arr.filter(fn)`.

The syntax is similar to `find`, but `filter` returns an array of all matching elements:

```
1 let results = arr.filter(function(item, index, array) {  
2   // if true item is pushed to results and the iteration continues  
3   // returns empty array if nothing found  
4});
```

For instance:

```
1 let users = [  
2   {id: 1, name: "John"},  
3   {id: 2, name: "Pete"},  
4   {id: 3, name: "Mary"}  
5];  
6  
7 // returns array of the first two users  
8 let someUsers = users.filter(item => item.id < 3);  
9  
10 alert(someUsers.length); // 2
```

## Transform an array

Let's move on to methods that transform and reorder an array.

### map

The `arr.map` method is one of the most useful and often used. It calls the function for each element of the array and returns the array of results.

■ ■ ■ ■	.map( ■ → ● )	→	● ● ● ●
■ ■ ● ■	.filter( ■ )	→	■ ■ ■
● ● ■ ■	.find( ■ )	→	■
● ● ● ■	.findIndexof( ■ )	→	3
■ ■ ■ ■	.fill(1, ● )	→	■ ● ● ●
● ■ ■ ●	.some( ■ )	→	true
■ ■ ■ ●	.every( ■ )	→	false

<https://javascript.info/array-methods>

<https://medium.com/@mandeepkaur1/a-list-of-javascript-array-methods-145d09dd19a0>

# Arrays

## .filter()

```
let users = [  
  { age: 35, name: "John" },  
  { age: 40, name: "Pete" },  
  { age: 44, name: "Mary" }  
];
```

// returns array of with users older than 39

```
let someUsers = users.filter(item => item.age > 39);
```

```
console.log(someUsers);
```

```
▼ Array(2) ⓘ  
  ► 0: {age: 40, name: "Pete"}  
  ► 1: {age: 44, name: "Mary"}  
  length: 2
```

# Arrays

## .filter() to search

```
const persons = [
  {
    name: "Birgitte Kirk Iversen",
    mail: "bk@gmail.dk",
    title: "Senior Lecturer",
    img: "https://www.eaaa.dk/media/u4gorzsd/birgitte-kirk-iversen2.jpg?width=800&height=450"
  },
  {
    name: "Martin Aagaard Nørh",
    mail: "mnor@mail.dk",
    title: "Lecturer",
    img: "https://www.eaaa.dk/media/oayjq02h/martin-n%C3%88hr.jpg?width=800&height=450"
  },
  {
    name: "Rasmus Cederorff",
    mail: "raco@mail.dk",
    title: "Senior Lecturer",
    img: "https://www.eaaa.dk/media/devlvgj/rasmus-cederorff.jpg?width=800&height=450"
  },
  {
    name: "Dan Okkels Brendstrup",
    mail: "dob@gmail.dk",
    title: "Lecturer",
    img: "https://www.eaaa.dk/media/bdoje141/dan-okkels-brendstrup.jpg?width=800&height=450"
  },
  {
    name: "Anne Kirketerp",
    mail: "anki@mail.dk",
    title: "Head of Department",
    img: "https://www.eaaa.dk/media/5buhi1xeo/anne-kirketerp.jpg?width=800&height=450"
  }
];
```

```
function search(event) {
  const searchValue = event.target.value.toLowerCase(); // input text to lower case
  const results = persons.filter(function (person) {
    return person.name.toLowerCase().includes(searchValue);
  });
  displayPersons(results); // call displayPersons with the filtered results
}
```

```
// with arrow function
function search(event) {
  const searchValue = event.target.value.toLowerCase(); // input text to lower case
  const results = persons.filter(person => person.name.toLowerCase().includes(searchValue));
  displayPersons(results); // call displayPersons with the filtered results
}
```

Filter condition:  
person.name  
must include  
searchValue

# Arrays

## .sort()

### JavaScript Demo: Array.sort()

```
1 const months = ['March', 'Jan', 'Feb', 'Dec'];
2 months.sort();
3 console.log(months);
4 // expected output: Array ["Dec", "Feb", "Jan", "March"]
5
6 const array1 = [1, 30, 4, 21, 100000];
7 array1.sort();
8 console.log(array1);
9 // expected output: Array [1, 100000, 21, 30, 4]
10
```

*“The `sort()` method sorts the elements of an array in place and returns the reference to the same array, now sorted.”*

# Arrays w/ objects

## .sort()

```
const persons = [
  {
    name: "Birgitte Kirk Iversen",
    mail: "bki@mail.dk",
    title: "Senior Lecturer",
    img: "https://www.eaaa.dk/media/u4gorzsd/birgitte-kirk-iversen2.jpg?width=800&height=450"
  },
  {
    name: "Martin Aagaard Nøhr",
    mail: "mnor@mail.dk",
    title: "Lecturer",
    img: "https://www.eaaa.dk/media/oayjq02h/martin-n%C3%B8hr.jpg?width=800&height=450"
  },
  {
    name: "Rasmus Cederdorff",
    mail: "race@mail.dk",
    title: "Senior Lecturer",
    img: "https://www.eaaa.dk/media/devlvgj/rasmus-cederdorff.jpg?width=800&height=450"
  },
  {
    name: "Dan Okkels Brendstrup",
    mail: "dob@mail.dk",
    title: "Lecturer",
    img: "https://www.eaaa.dk/media/bdojel41/dan-okkels-brendstrup.jpg?width=800&height=450"
  },
  {
    name: "Anne Kirketerp",
    mail: "anki@mail.dk",
    title: "Head of Department",
    img: "https://www.baaa.dk/media/5buh1xeo/anne-kirketerp.jpg?width=800&height=450"
  }
];

// Use localeCompare for strings
persons.sort(function (person1, person2) {
  person1.name.localeCompare(person2.name);
});

// with arrow function
persons.sort((person1, person2) => person1.name.localeCompare(person2.name));
```

# Code Every Day



Arnold Franciscus > main.js > main.js

```
</script>
</head>
<body>
<main>
<section id="hero--page">
  <ul class="left--items">
    <li class="right--items"><a href="#section--content" class="pink--button scroll">Mijn portfolio</a></li>
    <li class="right--items"><a href="#work--section" class="pink--button scroll">Mijn werk</a></li>
  </ul>
</nav>
<section id="intro--section">
  <article id="intro--section--text">
    
    Hey, ik ben Arnold!
    <h1>Arnold Franciscus</h1>
    <p class="section--text">Ik ben een full-stack developer en student applicatiewerking</p>
    <p><a href="#work--section" class="pink--button scroll">Mijn werk</a></p>
    
  </article>
  <section id="skills--section">
    <h2 class="section--header">Mijn Skills.</h2>
    <ul>
      <li>HTML</li>
      <li>CSS</li>
      <li>JavaScript</li>
      <li>React</li>
      <li>Node.js</li>
      <li>MongoDB</li>
      <li>MySQL</li>
      <li>Git</li>
      <li>GitHub</li>
      <li>Figma</li>
      <li>Adobe XD</li>
      <li>Photoshop</li>
    </ul>
  </section>
</main>
</body>
</html>
```

index.php x

Project Arnold Franciscus

File Edit View Navigator Code Help

MacBook Pro