

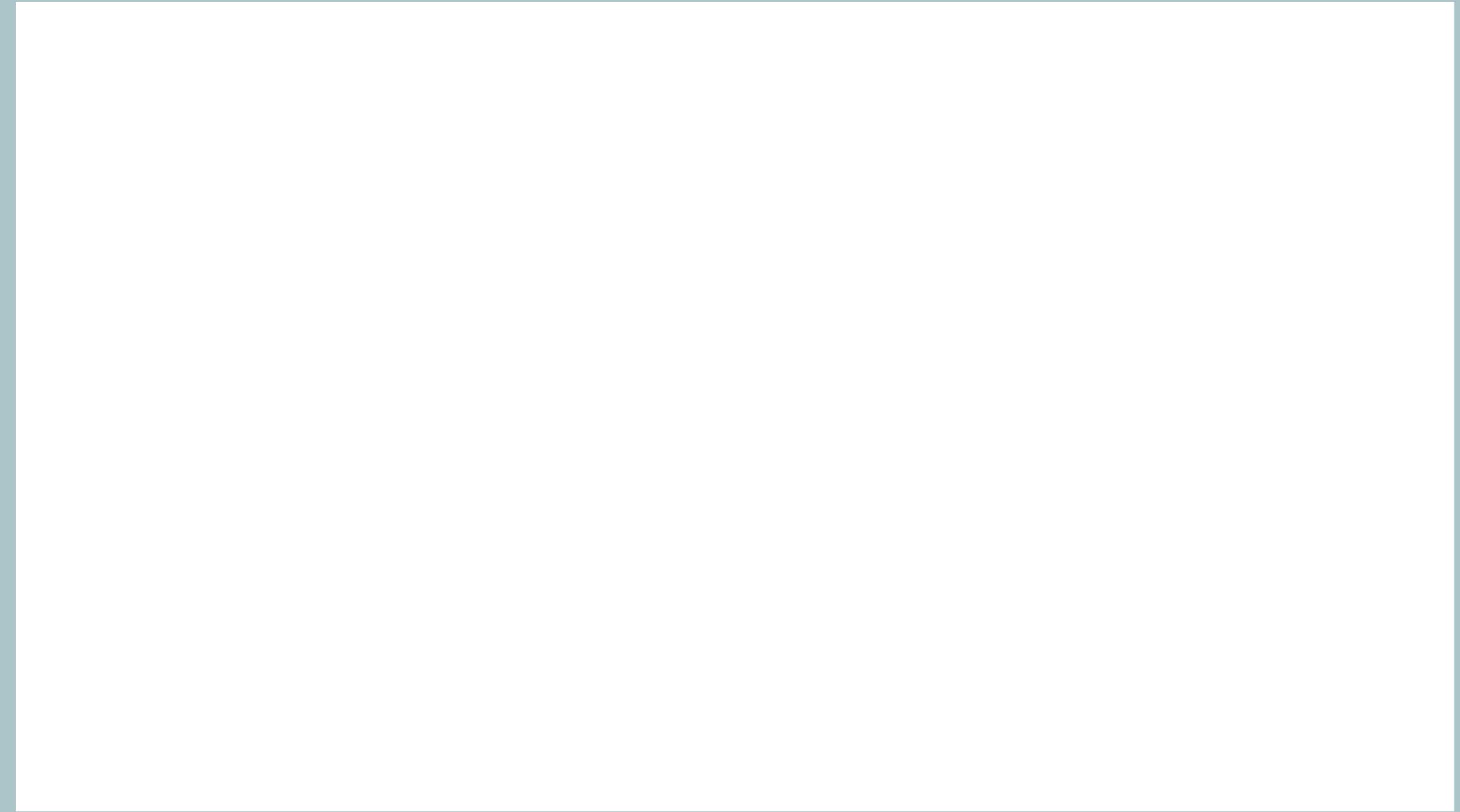
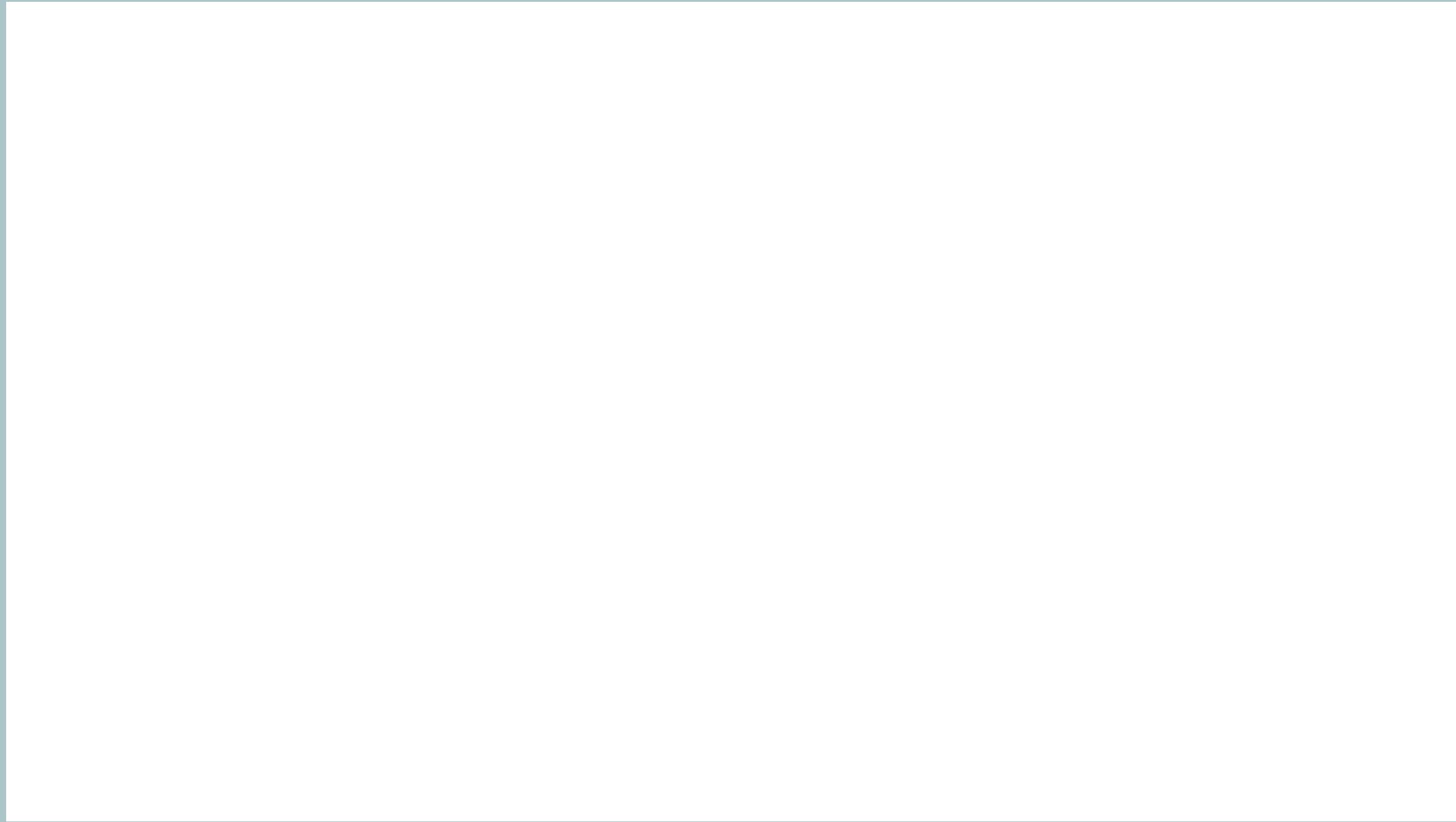
Frontend introduction

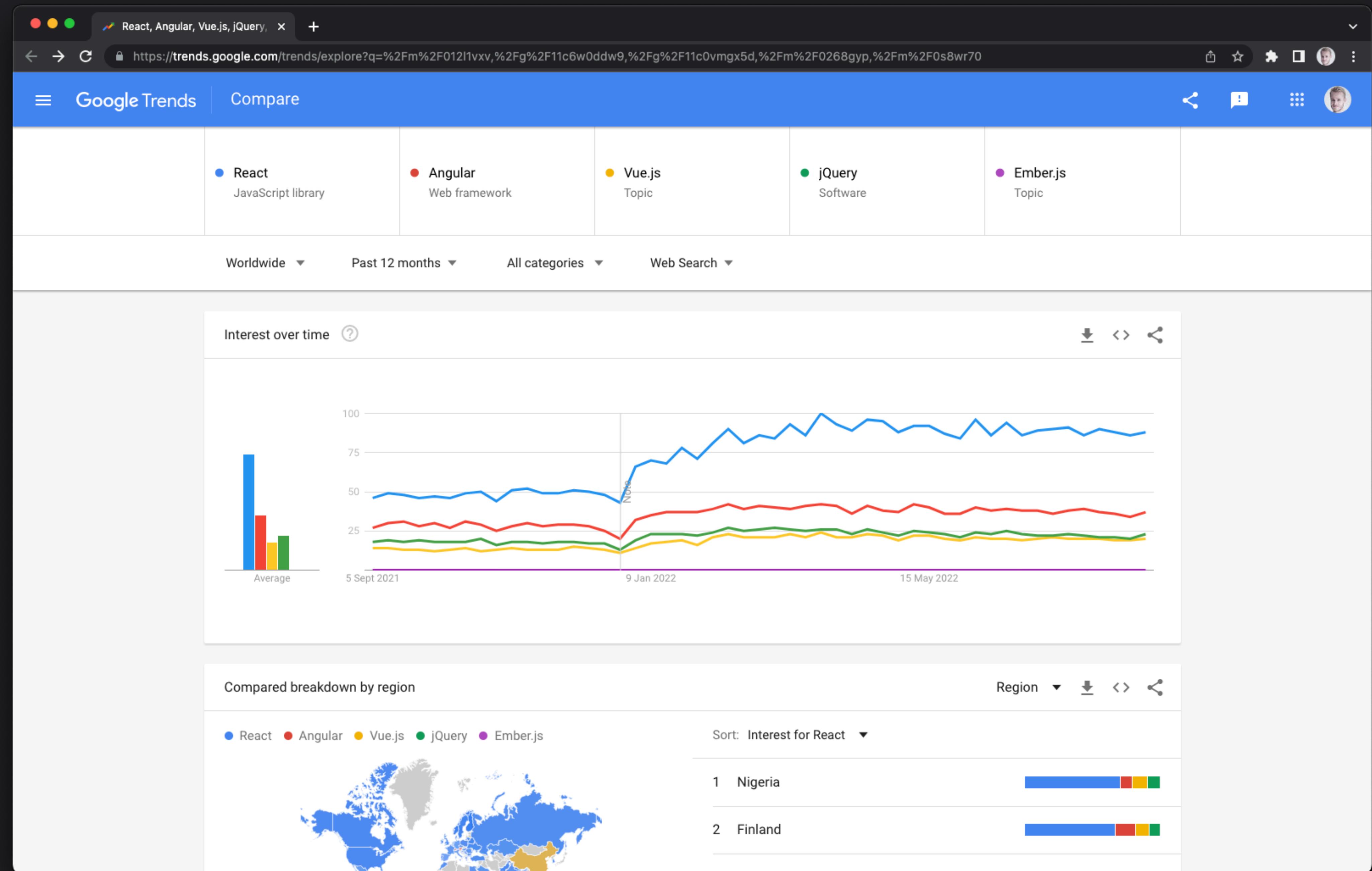
Web Development

ERHVERVSAKADEMI AARHUS

BUSINESS ACADEMY AARHUS

REMINDER





Agenda

- Frontend Dev Stack & Tools
- Canvas Users Case
 - CSS Layout & JS Concepts
- Introduction to React
 - Setup React Dev Environment
 - React Core Concepts
- Canvas Users Case w/ React

Lectures (RACE) ...

Date	Name	Notes
02/09/2022	<u>1. Frontend: Introduction</u>	Course Intro CSS Layouts From JS to React Pre React Intro
08/09/2022	<u>2. Frontend: Thinking in React</u>	Introduction to React React Core Concepts
19/09/2022	<u>3. Frontend: React & Single Page Apps</u>	React: Read + Create
26/09/2022	<u>4. Frontend: React & CRUD</u>	React: Update + Delete

Frontend stack

HTML



JS



CSS



STRUCTURE
CONTENT

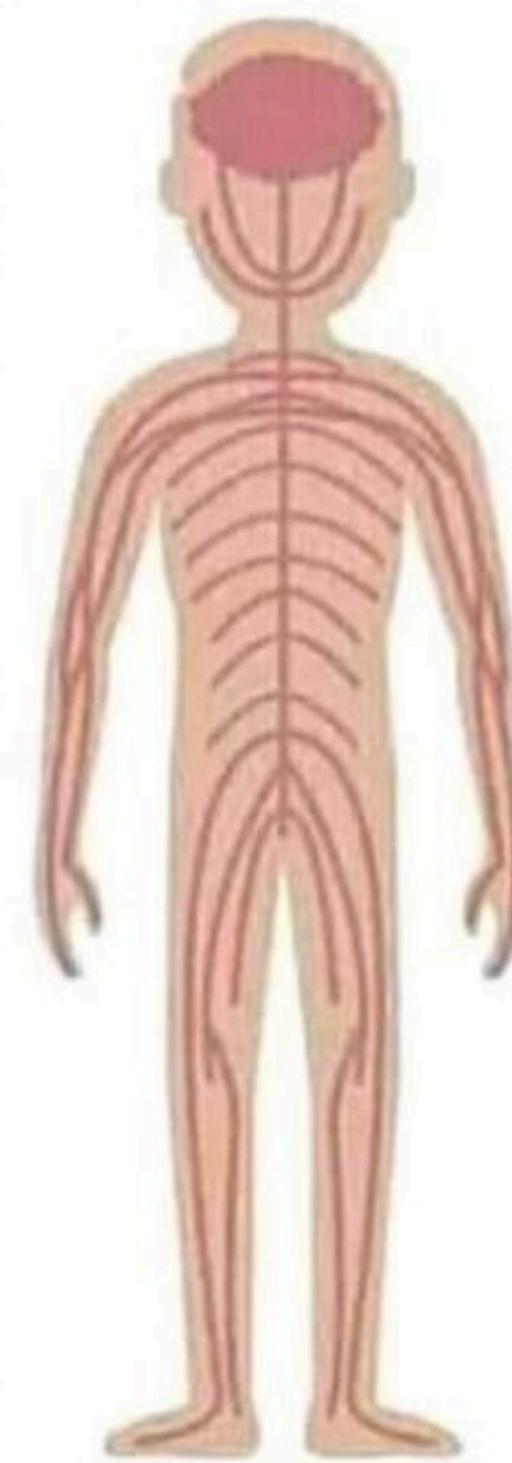
FUNCTIONALITY
BEHAVIOR

LAYOUT/STYLING
PRESENTATION

HTML



JS



CSS



JavaScript, HTML & CSS

You can literally build anything with it!

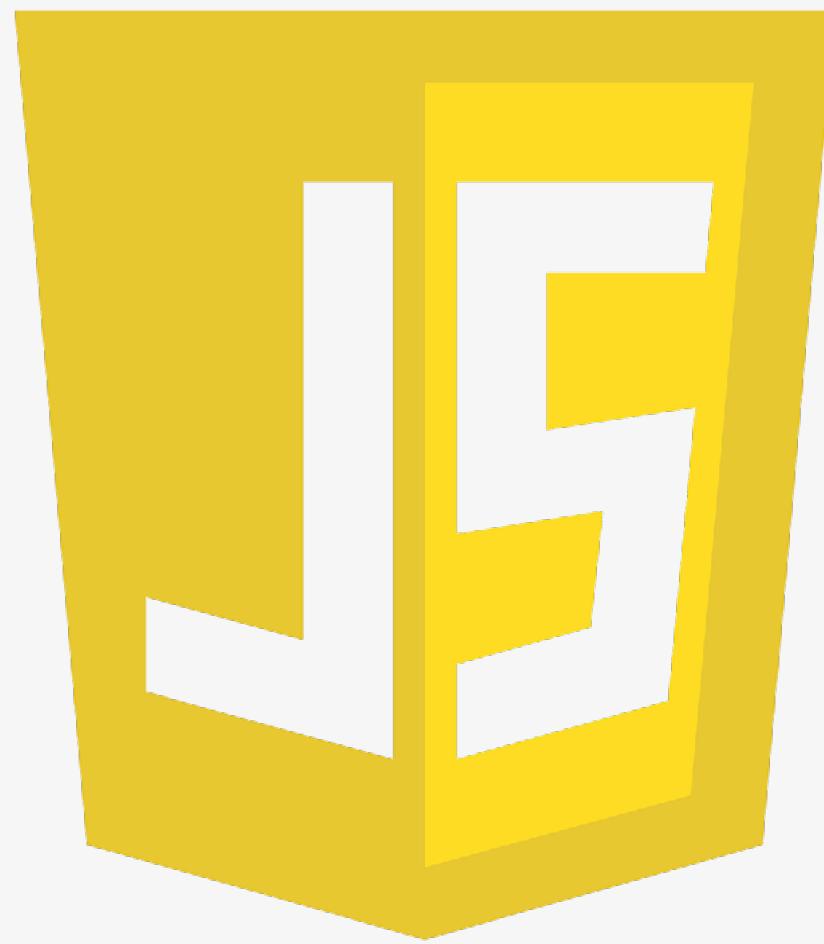
Separation of concerns

HTML



STRUCTURE
CONTENT

JS



FUNCTIONALITY
BEHAVIOR

CSS



LAYOUT/STYLING
PRESENTATION

Frontend Project Structure

- Project structure:
 - HTML
 - CSS
 - JavaScript
- Keep a good structure & be consistent
- Separation of concerns
- Naming & conventions

The image shows a screenshot of a dark-themed code editor, likely Visual Studio Code, displaying two separate project structures side-by-side.

Left Project (CSS-GRID):

- EXPLORER view: Shows files `app.js`, `index.html`, and `style.css`.
- Editor view: Shows the content of `index.html` with the following code:

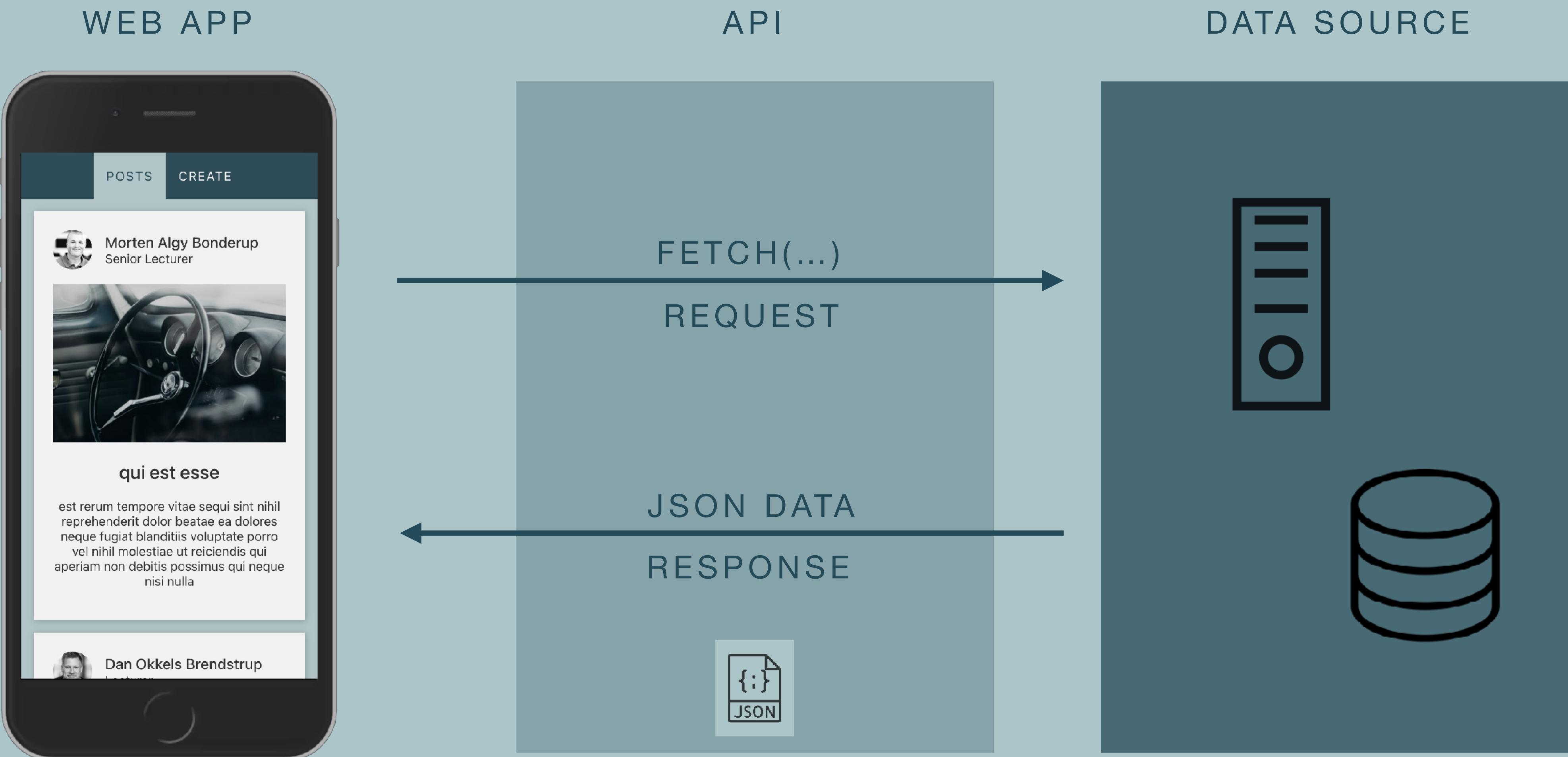
```
<!DOCTYPE html>
<html lang="en">
<head>
```

Right Project (PROJECT-TEMPLATE):

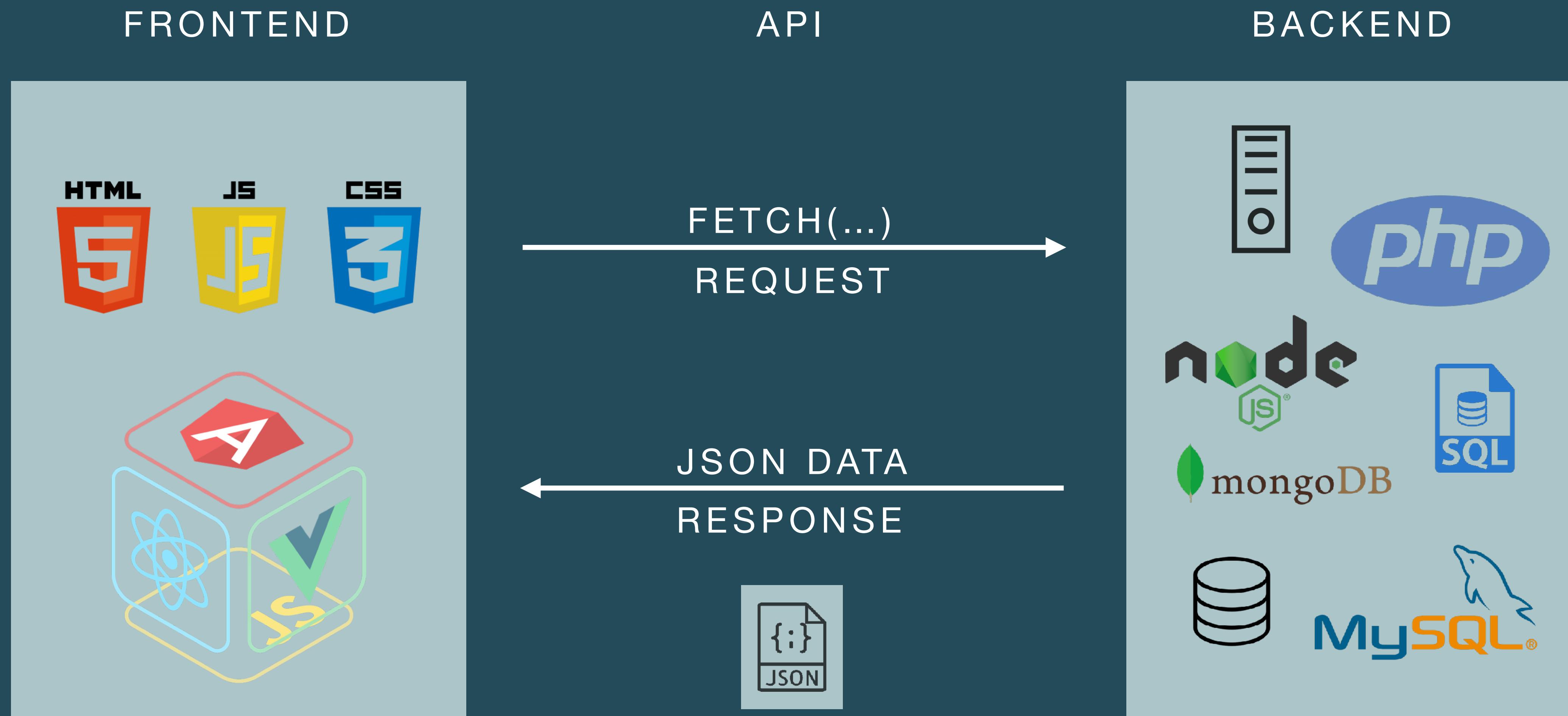
- EXPLORER view: Shows a folder structure with subfolders `css`, `img`, and `js`, and files `favicon.png`, `main.css`, and `main.js`.
- Editor view: Shows the content of `index.html` with the following code:

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>PROJECT TEMPLATE</title>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Get started with your new project template!">
<meta name="author" content="Rasmus Cederdorff" />
<link rel="stylesheet" href="css/main.css" type="text/css" />
<link rel="shortcut icon" type="image/png" href="img/favicon.png" />
</head>
<body>
<header>
<h1>PROJECT TEMPLATE</h1>
</header>
<button onclick="showAlert()">Hit me</button>
<section id="content"></section>
<!-- main js file -->
<script src="js/main.js"></script>
```

Web Development



Web Development



Tools

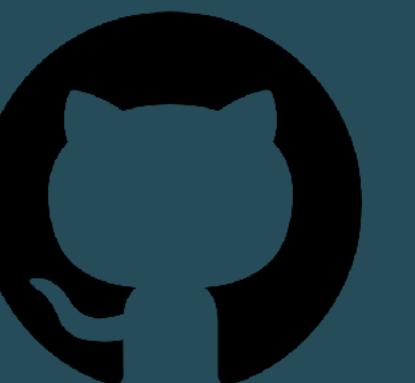


Visual Studio Code



Extensions: ESLint & Live Server

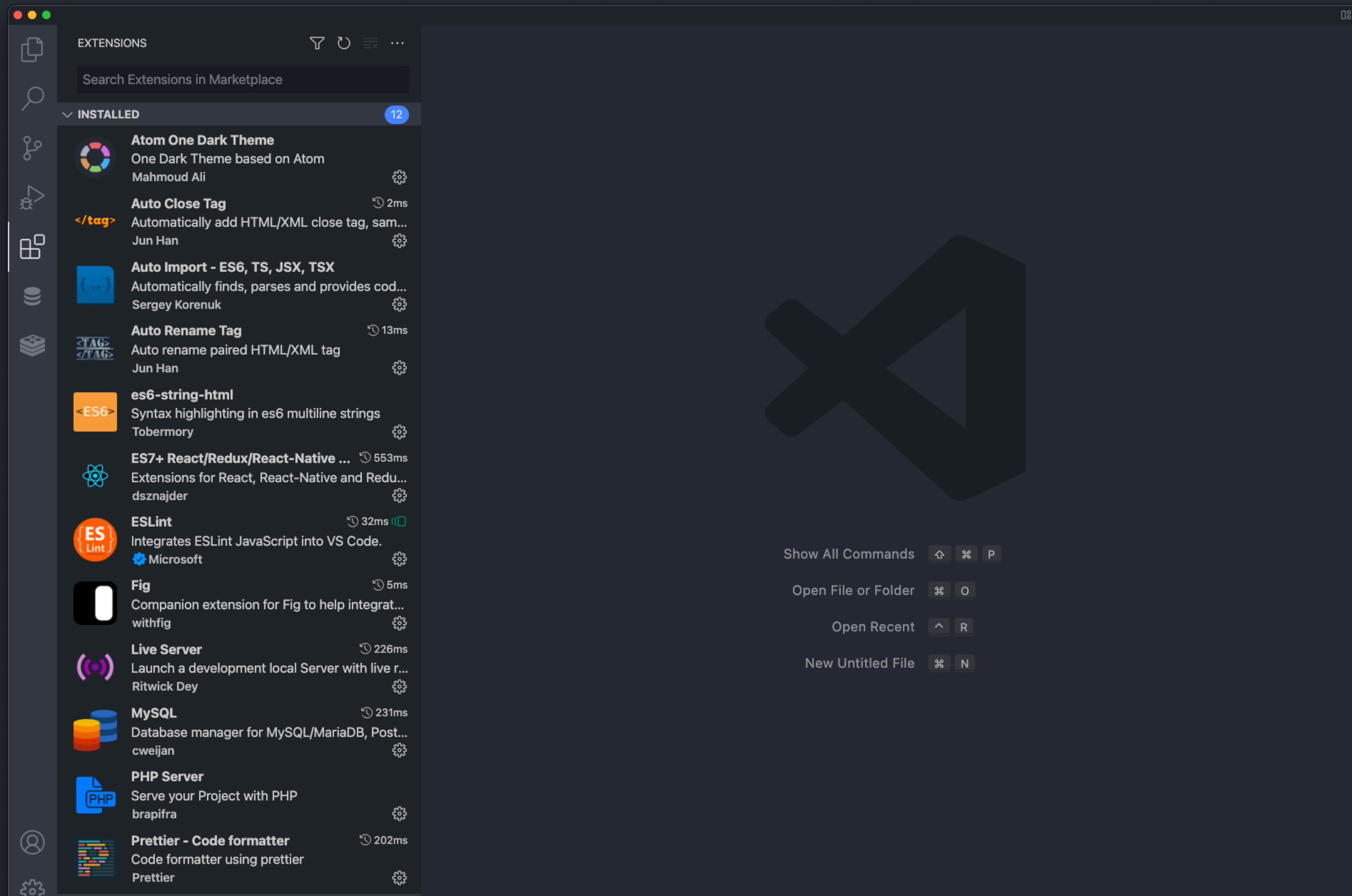
GitHub Desktop



Optional: FTP Client



Extensions



Choose default formatter in settings

The screenshot shows the VS Code settings interface with the search bar containing 'formatt'. The results show 21 settings found, with the 'User' tab selected. The 'Text Editor (4)' section includes 'Formatting (3)'. The 'Extensions (17)' section lists 'ESLint (1)', 'HTML (7)', 'JSON (1)', 'Prettier (5)', and 'TypeScript (3)'. A callout highlights the 'Prettier - Code formatter' entry under 'Text Editor (4)'. The 'Editor: Default Formatter' setting is described as defining a default formatter that takes precedence over all other formatter settings. It must be the identifier of an extension contributing a formatter. The 'Prettier - Code formatter' option is selected. Other settings shown include 'Editor: Format On Paste' (checked), 'Editor: Format On Save' (checked), and 'Editor: Format On Type' (unchecked). The 'HTML > Format: Content Unformatted' setting is also visible at the bottom.

Settings — mdu-frontend

Extension: Prettier - Code formatter

Settings

formatt

21 Settings Found

User Workspace

Turn on Settings Sync

INSTALLED 6

Atom One Dark Theme
One Dark Theme based...
Mahmoud Ali

Auto Close Tag 2ms
Automatically add HTM...
Jun Han

es6-string-html
Syntax highlighting in e...
Tobermory

ESLint 9ms
Integrates ESLint JavaS...
Dirk Baeumer

Live Server 280ms
Launch a development ...
Ritwick Dey

Prettier - C... 44ms
Code formatter using p...
Prettier

Editor: Default Formatter
Defines a default formatter which takes precedence over all other formatter settings. Must be the identifier of an extension contributing a formatter.
Prettier - Code formatter

Editor: Format On Paste
 Controls whether the editor should automatically format the pasted content. A formatter must be available and the formatter should be able to format a range in a document.

Editor: Format On Save
 Format a file on save. A formatter must be available, the file must not be saved after delay, and the editor must not be shutting down.

Editor: Format On Type
 Controls whether the editor should automatically format the line after typing.

HTML > Format: Content Unformatted
List of tags, comma separated, where the content shouldn't be reformatted. `null` defaults to the `pre` tag.

Edit in [settings.json](#)

Workflow

1. FETCH & PULL THE LATEST FROM GITHUB REPO WEB-FRONTEND (REACT-CDN-STARTERS) OR REACT-STARTERS)
2. CREATE YOUR OWN LOCAL DEV FOLDER
3. COPY PROJECTS FROM WEB-FRONTEND INTO YOUR OWN DEV FOLDER.
4. OPEN YOUR OWN LOCAL DEV FOLDER IN VS CODE.
5. USE LIVE SERVER OR PHP SERVER EXTENSION TO RUN PROJECTS (PHP SERVER WHEN RUNNING PHP PROJECTS ).
6. START WORKING IN YOUR OWN LOCAL DEV FOLDER.

OPTIONAL: SETUP GIT FOR YOUR OWN DEV FOLDER

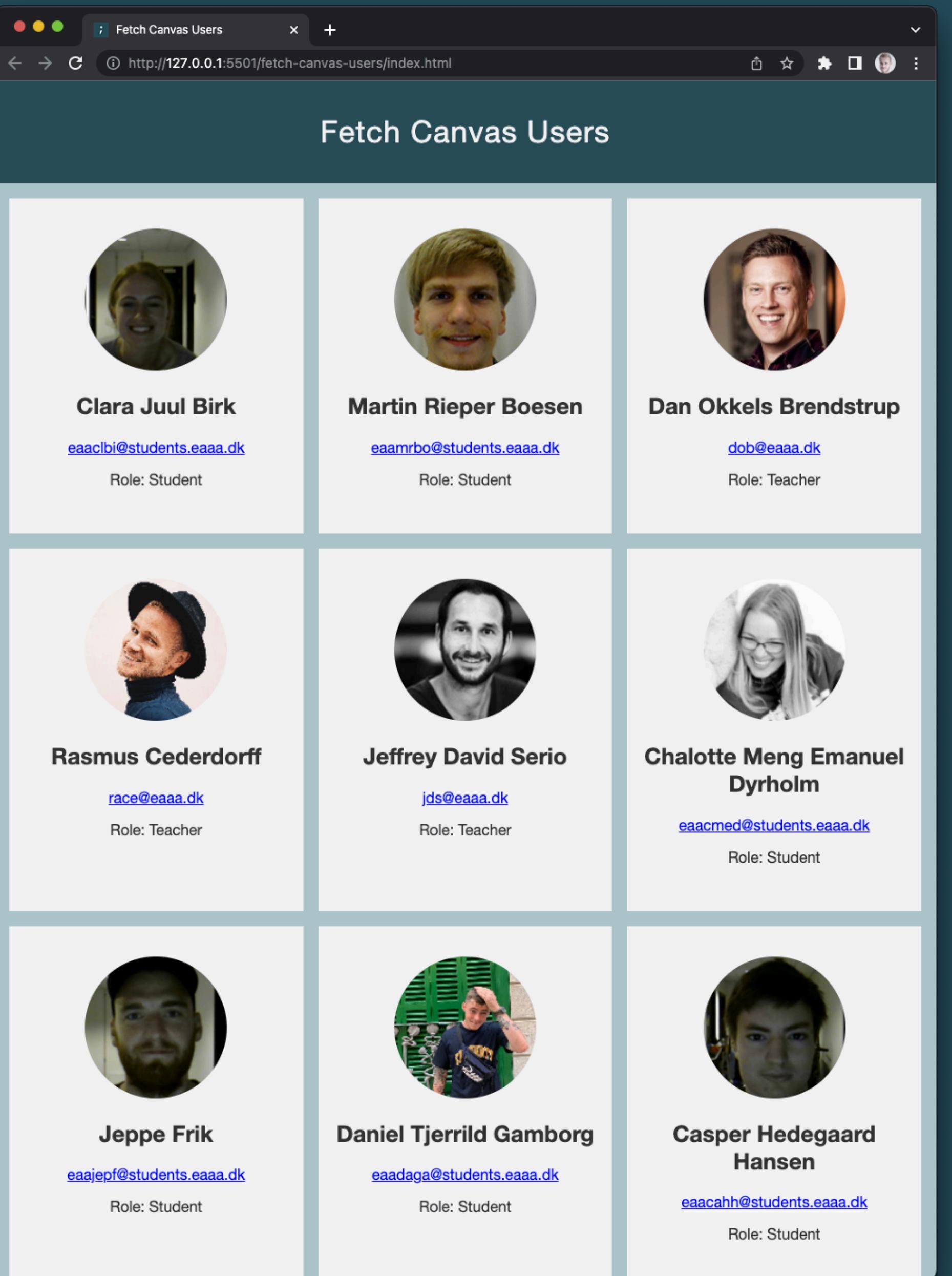
Setup dev environment

1. install extensions.
2. Choose default formatter.
3. Follow bullets on the workflow slide.

Canvas Users

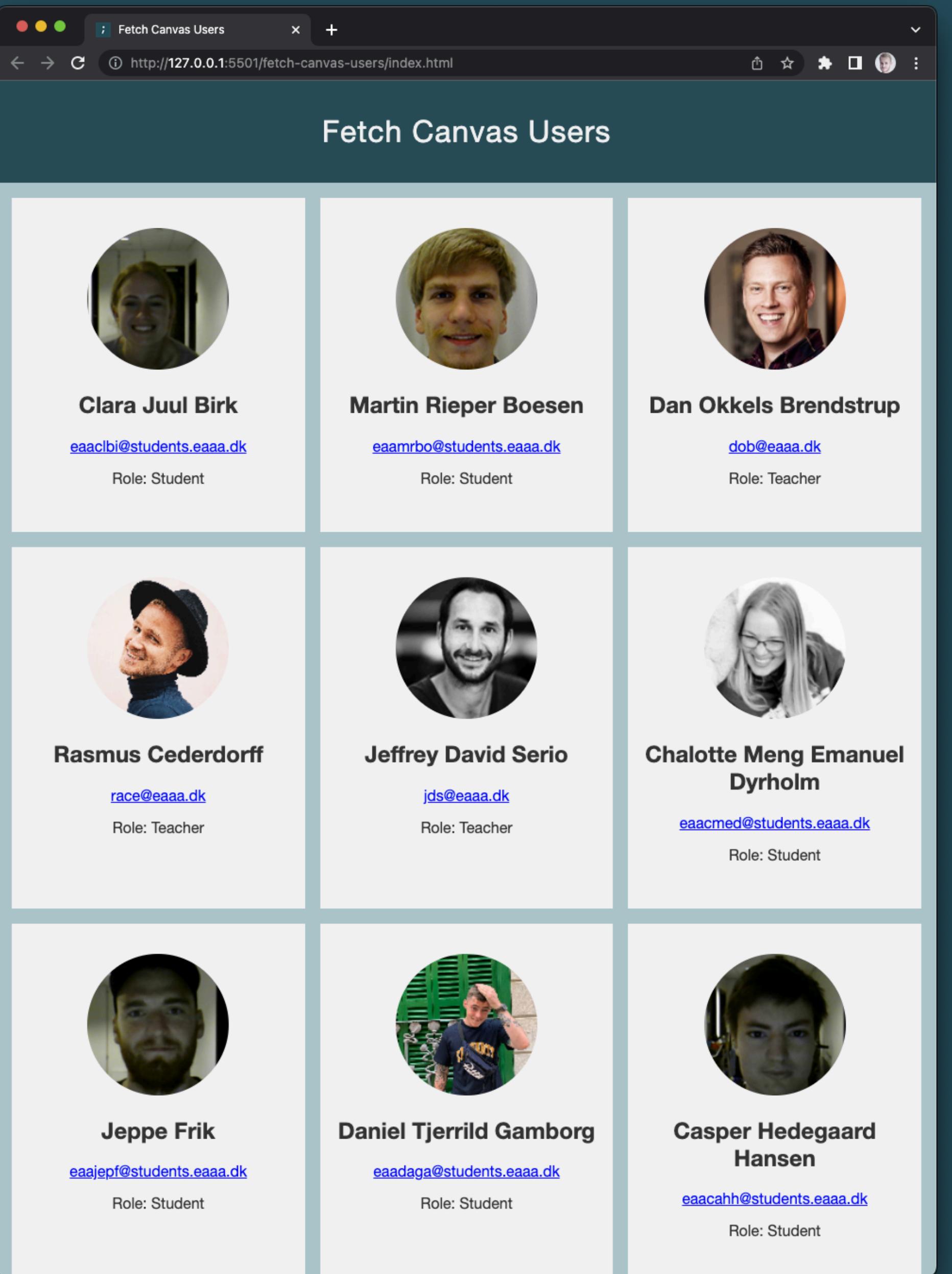
Fetch and display users from Canvas (data source) in a grid or flexbox.

1. Use [project-template](#) from GitHub.
2. In your `app.js` define a function named `getUsers`, fetching data from raw.githubusercontent.com/cederdorff/web-frontend/main/data/wu-e22a.json
3. Use `console.log()` to debug and test the data.
4. Create a function called `appendUsers(...)`. The function must loop through the user data and append each user to a grid container in the DOM.
 - 4.1. Use a `for of` loop. Inside of the loop you have to create a template for each user using a `template string`. The template is similar to how static-defined HTML would be.
 - 4.2. Make sure to execute the function(s).
 - 4.3. Add and/or customise CSS.



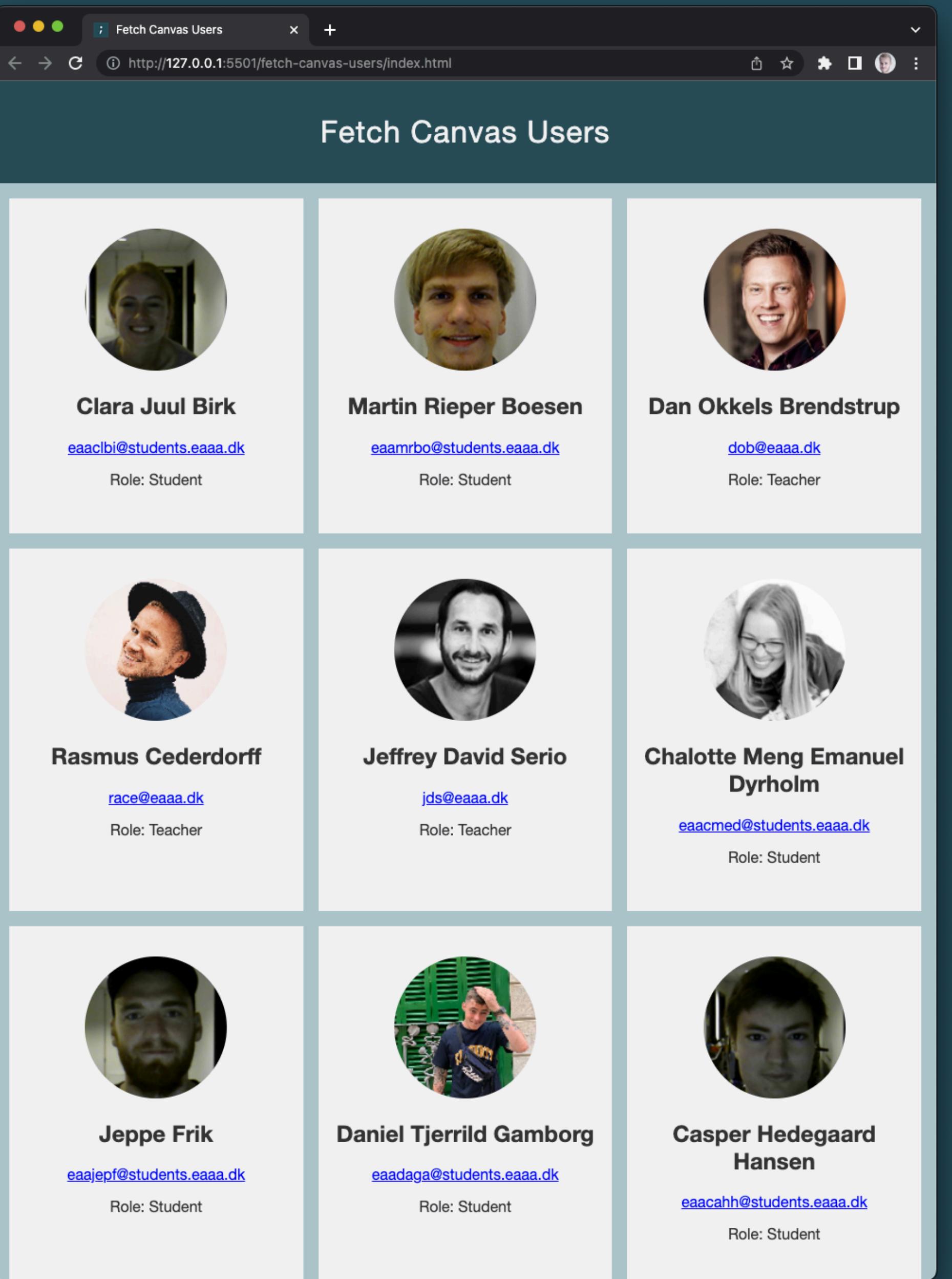
Canvas Users Cards

1. Explore and use [canvas-users-no-styling](#) from GitHub.
2. Make sure you understand index.html and how the CSS and JS are loaded.
3. Go through app.js and try to explain line by line.
 1. Identify and discuss the following: variables, functions, arguments/ parameters, template strings, DOM manipulation, loops, and the use of objects and arrays. Mark the different terms and concepts with comments in the js file. Use the linked slides to gain a deeper understanding of the terms and concepts.
 2. Note questions and comments.
4. When you have a solid knowledge of the project structure, start implementing CSS.
 1. Use CSS grid or flexbox to display all users nicely in a grid. Make sure to add a gap between the items (cards). Get inspired by the slides ([Flexbox & Grid](#)).
 2. Style the users as cards. Get inspired by this resource.



Canvas Users Extra

1. Implement search functionality searching on `name.includes(searchValue)`.
2. Implement sort functionality in order to sort by `name`, `sortable_name`, etc.
3. Make it possible to filter users on `enrollment_type` (students and teachers)
4. Implement create functionality using `array.push()`. Remember to generate a dummy `id` when adding a new user.
5. Create and implement an update-view using the `id` of the user to find and update properties of the user in the `users` array.
6. Implement delete functionality using the `id` to filter the `users` array.
7. Create a detail-view displaying all properties of the selected user.



WHAT IS HTML?

index.html ×

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>Page Title</title>
5    </head>
6    <body>
7      <h1>This is a Heading</h1>
8      <p>This is a paragraph.</p>
9    </body>
10   </html>
```

Hyper Text Markup Language

Standard markup language for creating Web pages

Describes the structure of a Web page

Consists of a series of elements

HTML elements tell the browser how to display the content

https://www.w3schools.com/html/html_intro.asp

HTML SEMANTIC ELEMENTS

... clearly describes its meaning to both the browser and the developer.

WHY USE SEMANTIC ELEMENTS?

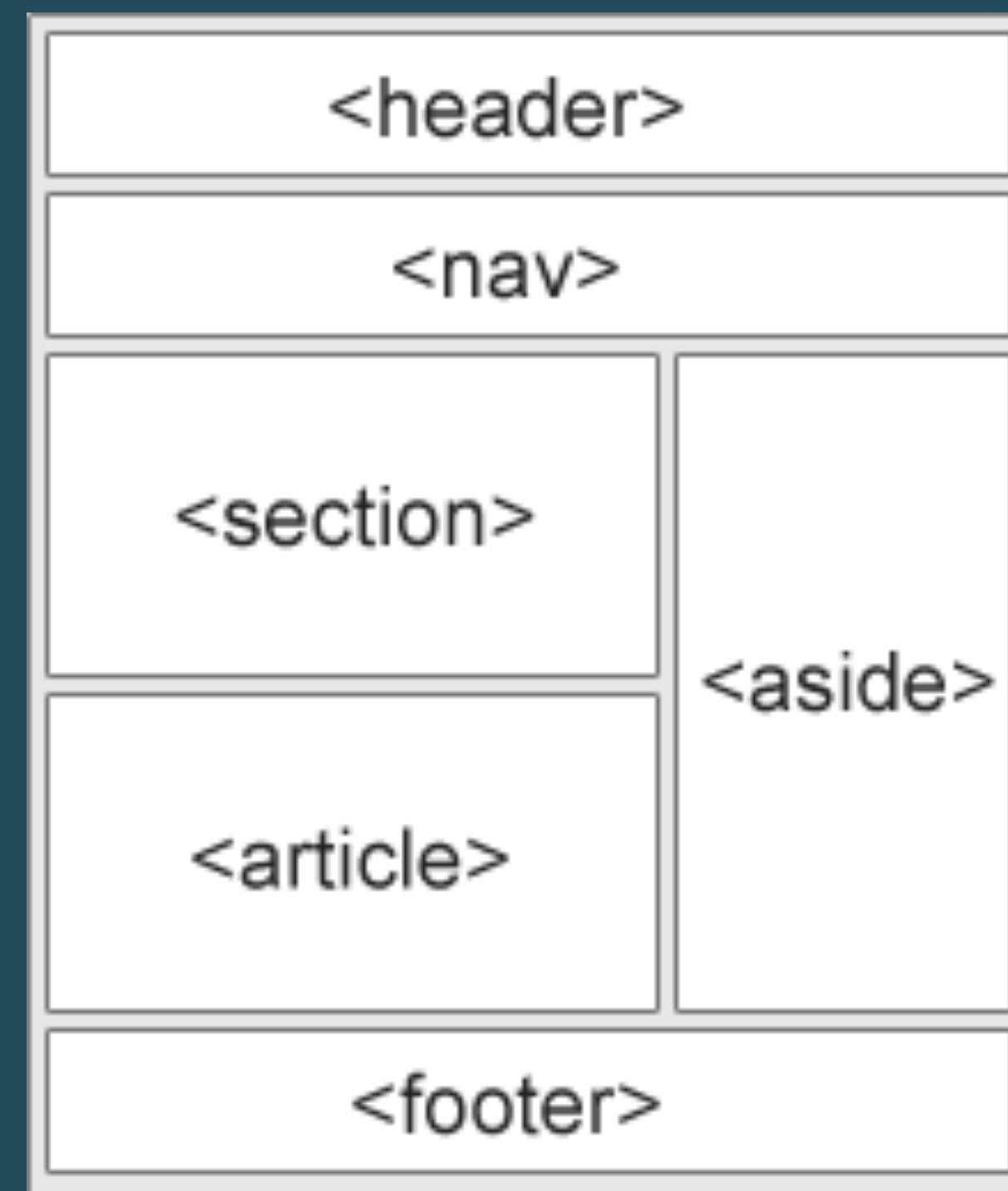
```
<body>
  <div id="header">
    <h1>PROJECT TEMPLATE</h1>
  </div>
  <div class="sections">
    <div class="article">
      <div class="figure">
        <img>
        <div class="figcaption"></div>
      </div>
    </div>
  </div>
  <div id="footer"></div>
  <!-- main js file -->
  <script src="js/main.js"></script>
</body>
```

```
<body>
  <header>
    <h1>PROJECT TEMPLATE</h1>
  </header>
  <section>
    <article>
      <figure>
        <img>
        <figcaption></figcaption>
      </figure>
    </article>
  </section>
  <footer></footer>
  <!-- main js file -->
  <script src="js/main.js"></script>
</body>
```

HTML SEMANTIC ELEMENTS

Semantic elements that can be used to define different parts of a website

- <article>
- <aside>
- <details>
- <figcaption>
- <figure>
- <footer>
- <header>
- <main>
- <mark>
- <nav>
- <section>
- <summary>
- <time>



D I V T A G S ?

Many web sites contain HTML code like:

```
<div id="nav">  
<div class="header">  
<div id="footer">
```

to indicate navigation, header, and footer.

Can we still use div tags?

Apple

apple.com

Store Mac iPad iPhone Watch TV Music Support

We look forward to welcoming you to our stores. Whether you [shop in a store](#) or shop online, our Specialists can help you buy the products you love.

iPhone 13 Pro

Oh. So. Pro.

[Learn more >](#) [Buy >](#)



iPhone 13

Your new superpower.

[Learn more >](#) [Buy >](#)

Elements Console Sources Network

```
<aside id="ac-localeswitcher" data-analytics-region="locale switcher" data-analytics-activitymap-region-id="locale switcher" lang="da-DK" dir="ltr">...</aside>
<h1 class="visuallyhidden">Apple</h1>
<meta name="ac-gn-store-key" content="SFX9YPYY9PPXCU9KH">
<aside id="ac-gn-segmentbar" class="ac-gn-segmentbar" lang="en-US" dir="ltr" data-strings="{ 'exit': 'Exit', 'view': '%STOREFRONT% Store Home', 'segments': { 'smb': 'Business Store Home', 'eduInd': 'Education Store Home', 'other': 'Store Home' } }"></aside>
<input type="checkbox" id="ac-gn-menustate" class="ac-gn-menustate">
►<nav id="ac-globalnav" class="js no-touch no-windows" role="navigation" aria-label="Global" data-hires="false" data-analytics-region="global nav" lang="en-US" dir="ltr" data-www-domain="www.apple.com" data-store-locale="us" data-store-root-path="/us" data-store-api="/[storefront]/shop/bag/status" data-search-locale="en_US" data-search-suggestions-api="/search-services/suggestions/defaultlinks/" data-search-defaultlinks-api="/search-services/suggestions/defaultlinks/">...</nav>
...
<div class="ac-gn-blur"></div> == $0
<div id="ac-gn-curtain" class="ac-gn-curtain"></div>
<div id="ac-gn-placeholder" class="ac-gn-placeholder"></div>
<script type="text/javascript" src="/ac/globalnav/6/en_US/scripts/ac-globalnav.built.js"></script>
►<div id="ac-gn-viewport-emitter">...</div>
<script src="/metrics/ac-analytics/2.13.1/scripts/ac-analytics.js" type="text/javascript" charset="utf-8"></script>
▼<main class="main" role="main">
  ▼<section class="homepage-section collection-module" data-module-template="ribbon"> grid
    ►<div data-unit-id="shop-online" data-analytics-activitymap-region-id="ribbon-shop-online">...</div>
  </section>
  ▼<section class="homepage-section collection-module" data-module-template="heroes" data-analytics-region="hero"> grid
    ►<div data-unit-id="iphone-13-pro" data-analytics-section-engagement="name:hero-iphone-13-pro">...</div>
    ►<div data-unit-id="iphone-13" data-analytics-section-engagement="nam...
... .enhanced-layout body.page-home.ac-nav-overlap.body-with-ribbon div.ac-gn-blur ...

```

Styles Computed Layout Event Listeners DOM Breakpoints Properties

Filter :hov .cls +

```
element.style {
}
html #ac-globalnav, html #ac-globalnav ~ .ac-gn-blur, html #ac-gn-segmentbar, html #ac-localeswitcher, html div.adv-wrapper {
  position: fixed;
}
```

Google

google.com

Gmail Billeder Log ind

Google

Google-søgning Jeg prøver lykken

Google er tilgængelig på: Føroyskt

Danmark

CO2-neutral siden 2007

Om Annoncering Erhverv Sådan fungerer Google Søgning Privatliv Vilkår Indstillinger

Elements Console ↗ 1 ↘ 1 ⚙ ⋮ X

```
<!DOCTYPE html>
<html itemscope itemtype="http://schema.org/WebPage" lang="da">
  <head>...</head>
  <body jsmodel="hspDDf" jsaction="YUC7He:.CLIENT;vPBs3b:.CLIENT;IVKTfe:.CLIENT;KsNBn:.CLIENT;sbTXNb:.CLIENT;xjhTIf:.CLIENT;02vyse:.CLIENT;Ez7VMc:.CLIENT;qqf0n:.CLIENT;me3ike:.CLIENT;IrNywb:.CLIENT;Z94jBf:.CLIENT;A8708b:.CLIENT;YcfJ:.CLIENT;A6SDQe:.CLIENT;LjVEJd:.CLIENT;VM8bg:.CLIENT;hWT9Jb:.CLIENT;wCulWe:.CLIENT;NTJodf:.CLIENT;szjOR:.CLIENT;PY1zjf:.CLIENT;wnJTPd:.CLIENT;JL9QDc:.CLIENT;kWlxhc:.CLIENT;qGMTIf:.CLIENT">
    <style data-iml="1634195875072">...</style>
    ... <div class="L3eUgb" data-hveid="1"> flex == $0
      <div class="o3j99 n1xJcf Ne6nSd">...</div> flex
      <div class="o3j99 LLD4me yr19Zb LS80J">...</div> flex
      <div class="o3j99 ikrT4e om7nvf">...</div>
      <div class="o3j99 qarstb">...</div>
      <div class="o3j99 c93Gbe">...</div>
    </div>
    <div class="Fgvgjc">
      <style data-iml="1634195875111">
        .Fgvgjc{height:0;overflow:hidden}</style>
      <div class="gTMtLb fp-nh" id="lb">...</div>
      <div jscontroller="fKZehd" style="display:none" data-u="0" jsdata="C4mkuf;_;AzQv+I" jsaction="rcuQ6b:npT2md">
      <span style="display:none">...</span>
      <script nonce="4eyH3+1nxzKy0S3HAsRNUA==">...</script>
      <div>...</div>
    </div>
  html body div.L3eUgb
  Styles Computed Layout Event Listeners DOM Breakpoints >
```

Filter :hov .cls + []

```
element.style {
}
.L3eUgb {
  display: flex;
  flex-direction: column;
}
```

The screenshot shows a web browser window with the title "HTML Semantic Elements" from the website w3schools.com. The page content is as follows:

HTML Semantic Elements

Semantic elements = elements with a meaning.

What are Semantic Elements?

A semantic element clearly describes its meaning to both the browser and the developer.

Examples of **non-semantic** elements: `<div>` and `` - Tells nothing about its content.

Examples of **semantic** elements: `<form>`, `<table>`, and `<article>` - Clearly defines its content.

Semantic Elements in HTML

Many web sites contain HTML code like: `<div id="nav">` `<div class="header">` `<div id="footer">` to indicate navigation, header, and footer.

In HTML there are some semantic elements that can be used to define different parts of a web page:

- `<article>`
- `<aside>`
- `<details>`
- `<figcaption>`

The right side of the screenshot shows the browser's developer tools with the "Elements" tab selected. The DOM tree displays the following structure:

```
<body> == $0
  > <div class="w3-bar w3-card-2 notranslate" style="position: relative; z-index: 4; font-size: 18px; background-color: white; color: #282A35; padding-left: 12px; padding-right: 16px; font-family: 'Source Sans Pro', sans-serif;">...</div>
  > <div style="display: none; position: absolute; z-index: 4; right: 52px; height: 44px; background-color: #282A35; letter-spacing: normal;" id="googleSearch">...</div>
  > <div style="display: none; position: absolute; z-index: 3; right: 111px; height: 44px; background-color: #282A35; text-align: right; padding-top: 9px;" id="google_translate_element"></div>
  > <div class="w3-card-2 topnav notranslate" id="topnav" style="position: relative;">...</div>
  > <div id="myAccordion" class="w3-card-2 w3-center w3-hide-large w3-hide-medium" style="width: 100%; position: absolute; display: none; background-color: rgb(231, 233, 235); padding-top: 0px;">...</div>
  > <script> </script>
  > <div class="w3-sidebar w3-collapse" id="sidenav" style="top: 118px; display: none;">...</div>
  > <div class="w3-main w3-light-grey" id="belowtopnav" style="margin-left: 220px; padding-top: 0px;">
    > <div class="w3-row w3-white">
        > ::before
        > <div class="w3-col l10 m12" id="main">
            > <div id="mainLeaderboard" style="overflow: hidden;">...</div>
            > <h1>...</h1>
            > <div class="w3-clear nextprev"> </div>
    > </div>
  > </div>
  > <html> <body>
```

The developer tools also show a "Styles" tab with a CSS rule for `element.style` and a "Computed" tab showing styles for `html` and `body` elements, including `font-family: Verdana, sans-serif;` and `font-size: 15px;`. The URL in the address bar is `w3schools.com/html/html5_semantic_elements.asp`.

DIV TAGS?

Use semantic elements as much as possible (!)

... but don't let it stop you from building

something amazing 

- RACE.js

WHAT IS CSS?

CSS stands for **Cascading Style Sheets**

CSS describes how HTML elements are to be displayed on screen, paper, or in other media

CSS **saves a lot of work**. It can control the layout of multiple web pages all at once

External stylesheets are stored in **CSS files**

https://www.w3schools.com/css/css_intro.asp

```
styles.css x
1 body {
2   font-family: Helvetica, Arial, sans-serif;
3   text-align: center;
4 }
5
6 h1 {
7   font-weight: lighter;
8 }
9
10 #my-paragraph {
11   color: blue;
12 }
13
14 .big {
15   font-size: 25px;
16 }
17
```

CSS SELECTORS

CSS selectors are used to "find" (or select) the HTML elements you want to style.

Simple selectors (name, id, class)

Combinator selectors (a specific relationship between them)

Pseudo-class selectors (a certain state)

Pseudo-elements selectors (a part of an element)

Attribute selectors (attribute or attribute value)

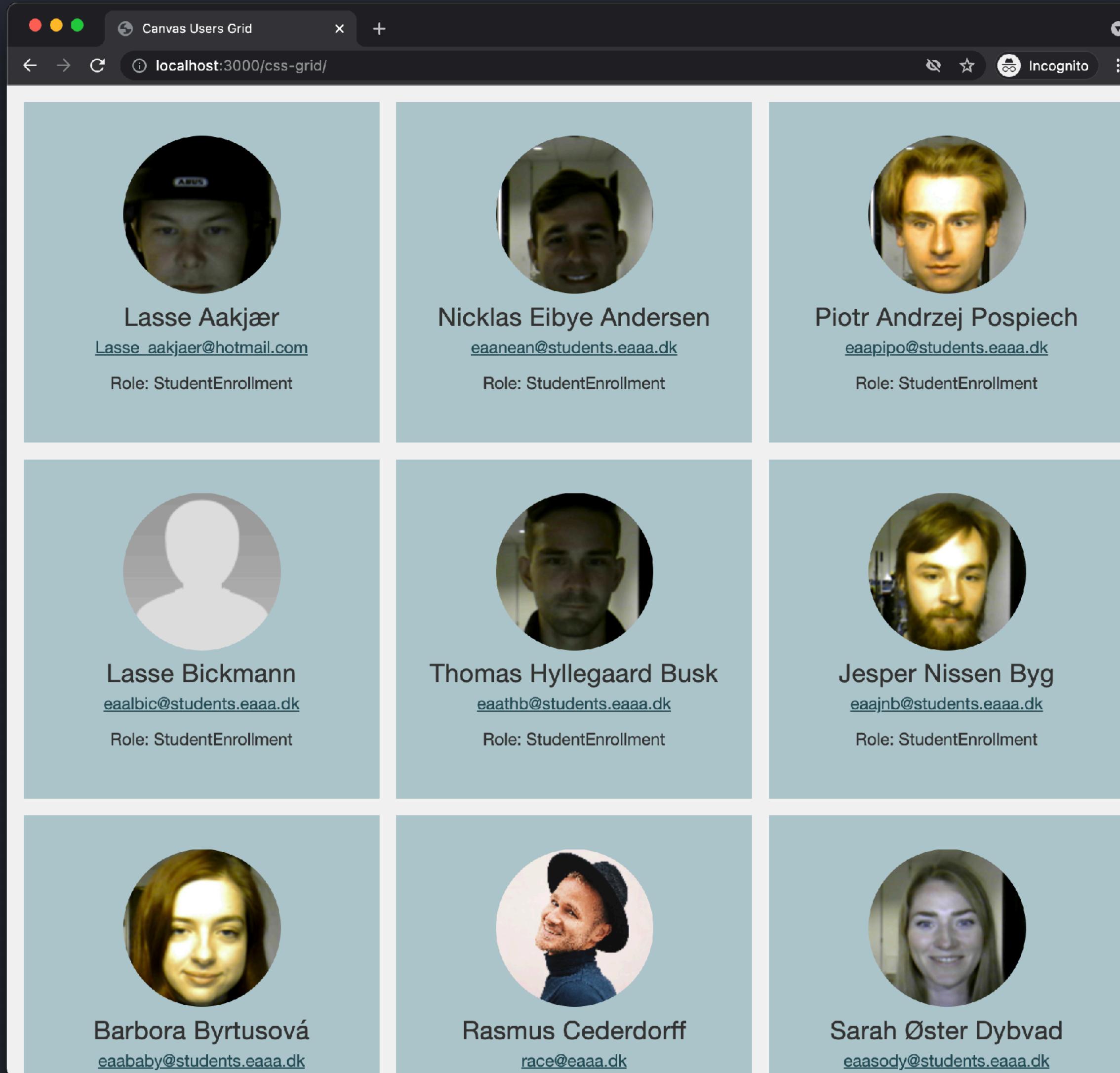
CSS LAYOUTS

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 GRID



```
<section id="users" class="grid"></section>
```

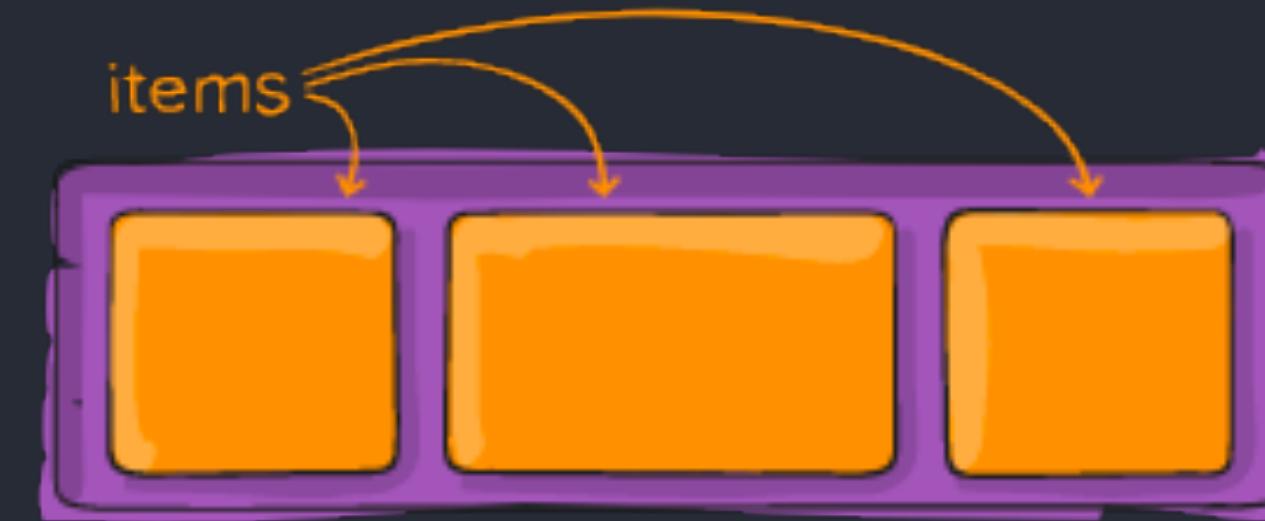
```
/* ----- grid styling ----- */
```

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

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

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

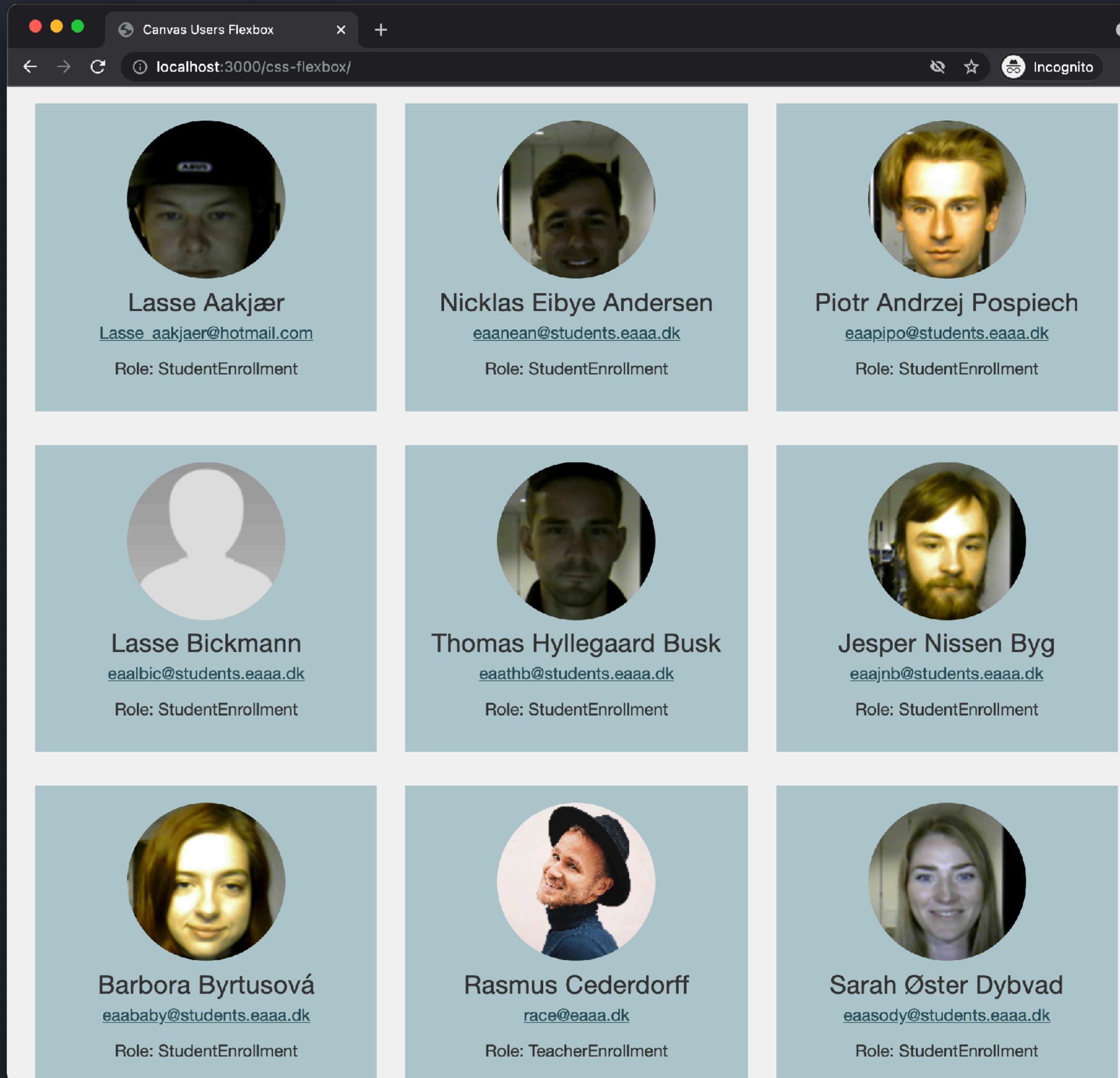
CSS GRID



```
.grid {  
  display: grid;  
  grid-template-columns: 1fr;  
  padding: 1em;  
  gap: 1em;  
}  
  
@media (min-width: 600px) {  
  .grid {  
    grid-template-columns: 1fr 1fr;  
  }  
}  
  
@media (min-width: 992px) {  
  .grid {  
    grid-template-columns: 1fr 1fr 1fr;  
  }  
}
```

```
.grid > article {  
  text-align: center;  
  padding: 2em 1em;  
  background-color: var(--light-green);  
  transition: 0.3s;  
}  
  
.grid > article:hover {  
  box-shadow: 0 8px 16px 0 var(--green);  
}  
  
.grid > article img {  
  width: 100%;  
  max-width: 150px;  
  height: auto;  
  border-radius: 50%;  
}
```

CSS FLEXBOX



```
<section id="users" class="flexbox">...</section>

/* ----- flexbox styling ----- */

.flexbox {
  display: flex;
  flex-wrap: wrap;
  justify-content: space-evenly;
}

.flexbox article {
  margin: 1em;
  padding: 1em;
  text-align: center;
  background-color: var(--light-green);
  transition: 0.3s;
  flex-basis: 100%; /*default size of an element before the remaining space is distributed*/
  flex-grow: 1; /*ability for a flex item to grow. 1 = all items equally large*/
}

@media (min-width: 600px) {
  .flexbox article {
    flex-basis: 40%;
  }
}

@media (min-width: 992px) {
  .flexbox article {
    flex-basis: 27%;
  }
}
```

CSS FLEXBOX



```
.flexbox {  
  display: flex;  
  flex-wrap: wrap;  
  justify-content: space-evenly;  
}
```

```
.flexbox article {  
  margin: 1em;  
  padding: 1em;  
  text-align: center;  
  background-color: var(--light-green);  
  transition: 0.3s;  
  flex-basis: 100%; /*default size of an element before the remaining space is distributed*/  
  flex-grow: 1; /*ability for a flex item to grow. 1 = all items equally*/  
}  
  
@media (min-width: 600px) {  
  .flexbox article {  
    flex-basis: 40%;  
  }  
}  
  
@media (min-width: 992px) {  
  .flexbox article {  
    flex-basis: 27%;  
  }  
}
```

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 Responsive
RWD Intro
RWD Viewport
RWD Grid View
RWD Media Queries
RWD Images
RWD Videos
RWD Frameworks
RWD Templates

CSS Grid
Grid Intro
Grid Container
Grid Item

CSS SASS
SASS Tutorial

CSS Examples
CSS Templates
CSS Examples
CSS Quiz
CSS Exercises
CSS Certificate

CSS References
CSS Reference
CSS Selectors

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:

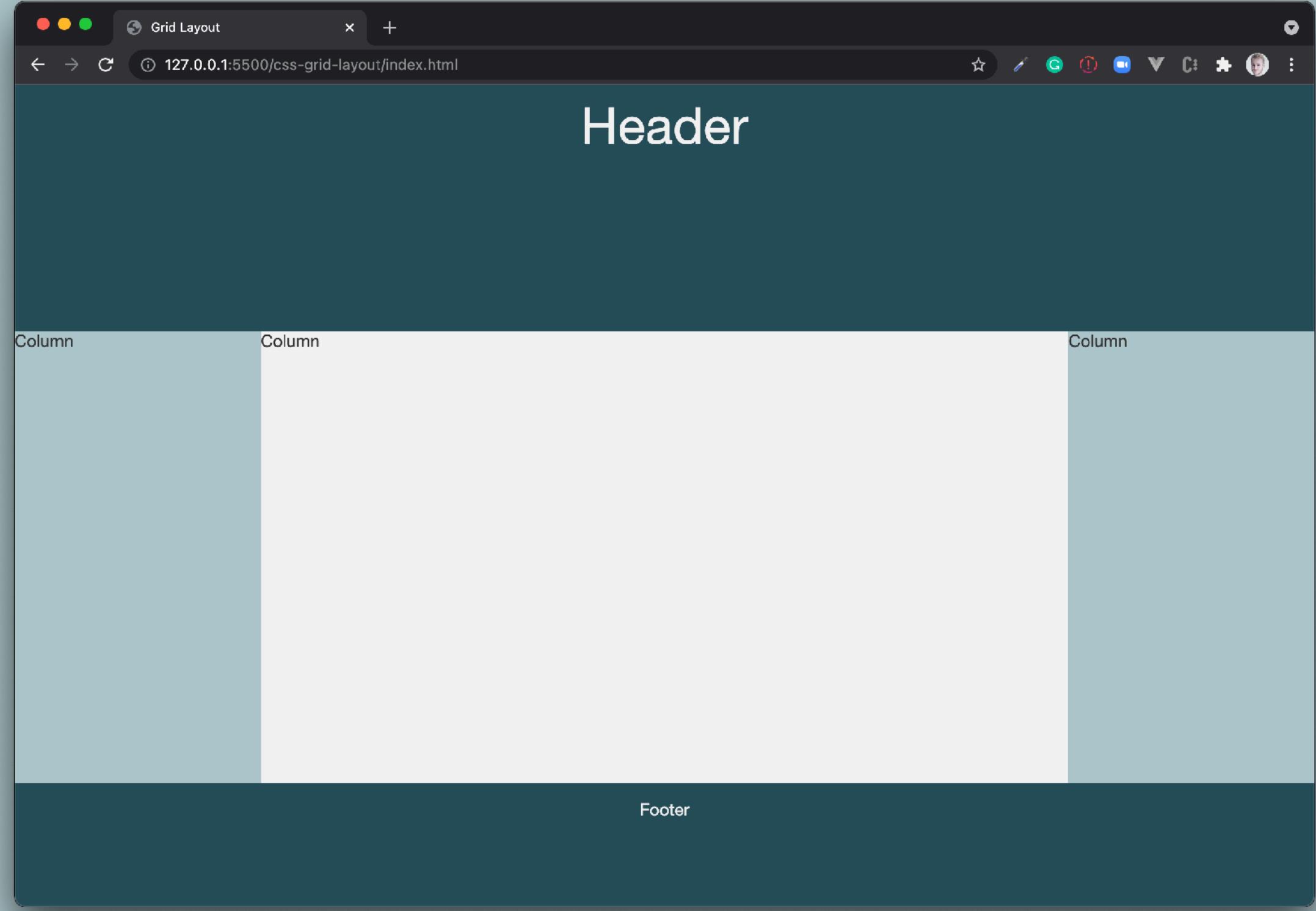
Sidenav and content:

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

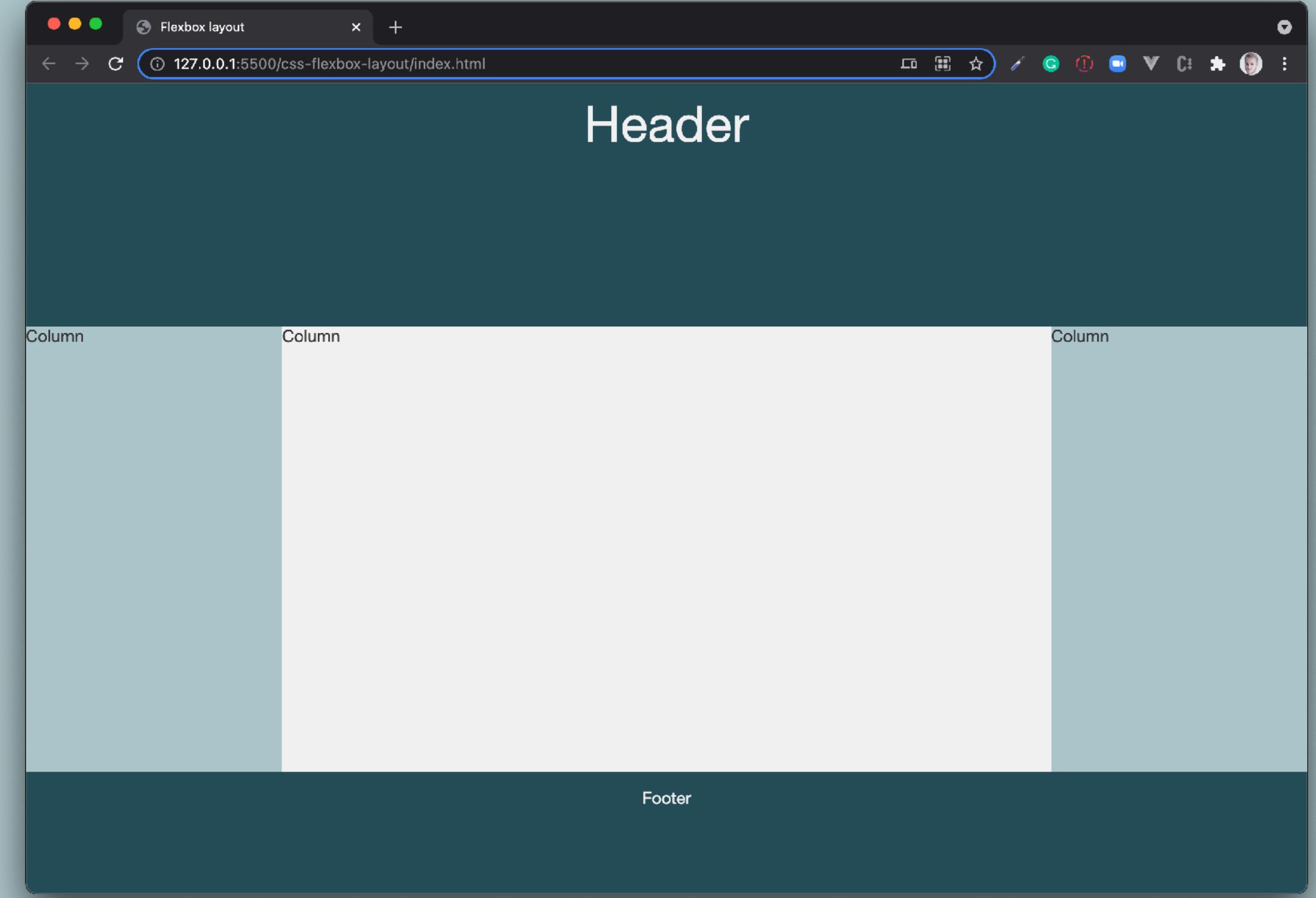
NEW
We just launched W3Schools videos
Explore now

COLOR PICKER

LIKE US



css-grid-layout



css-flexbox-layout

CSS LAYOUT.IO

LAYOUTS AND PATTERNS MADE WITH CSS

CSS TEMPLATES

https://www.w3schools.com/css/css_templates.asp

CSS VARIABLES

`var()` - refer to a defined CSS variable.

CSS variables are declared inside
the `:root` selector

CSS variables can be accessed through
the entire CSS document.

Makes the code easier to read (more
understandable)

Makes it much easier to change the color
values

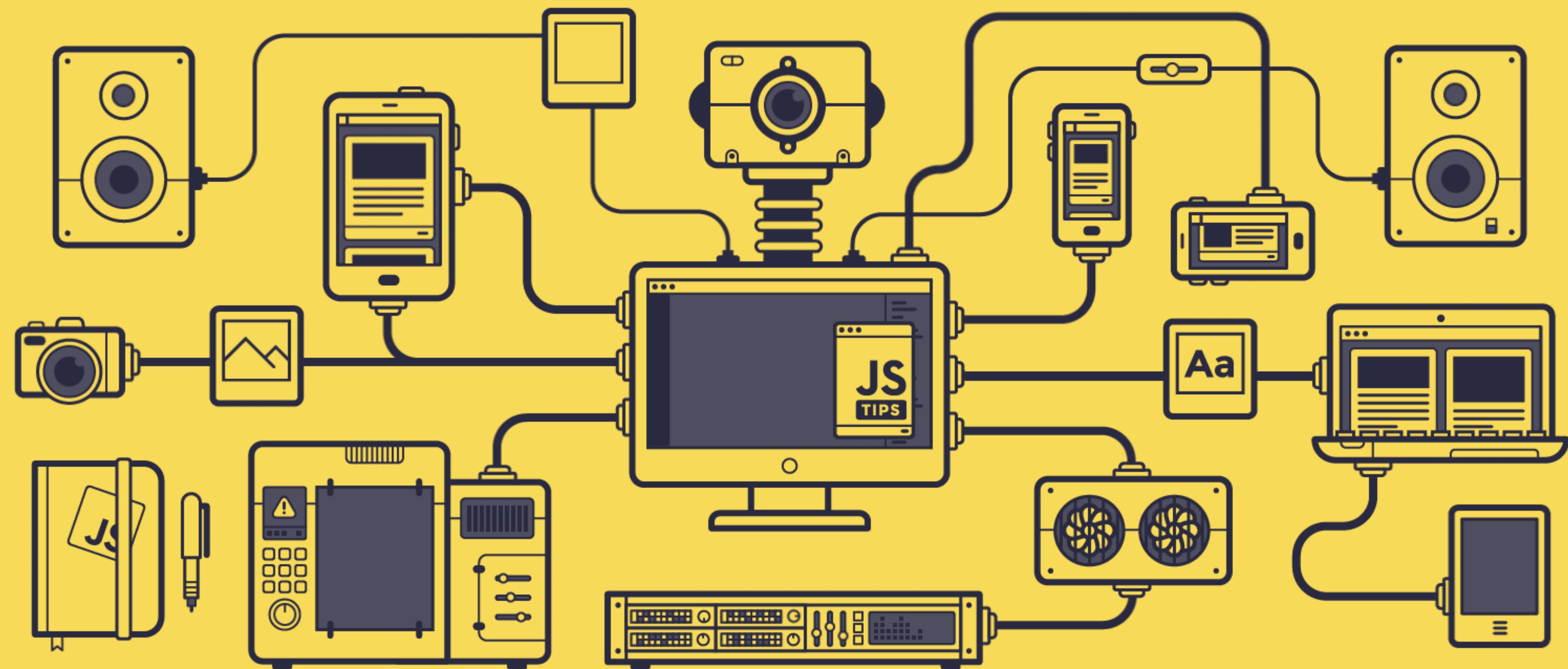
https://www.w3schools.com/css/css3_variables.asp

```
/* ----- root variables ----- */
:root {
    --green: #rgb(38, 76, 89);
    --green-opacity: #rgba(38, 76, 89, 0.2);
    --light-green: #rgb(172, 198, 201);
    --light-grey: #f1f1f4;
    --text-color-light: #f1f1f1;
    --text-color-dark: #333;
    --white: #fff;
    --font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;
}

/* ----- styling ----- */
html,
body {
    color: var(--text-color-dark);
    font-family: var(--font-family);
    margin: 0;
    padding: 0;
    border: 0;
    outline: 0;
    background-color: var(--light-grey);
}

a {
    color: var(--green);
}

.grid > article {
    text-align: center;
    padding: 2em 1em;
    background-color: var(--light-green);
    transition: 0.3s;
}
```



```
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.

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

... can change styling of HTML.

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

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 

DOM MANIPULATION

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

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

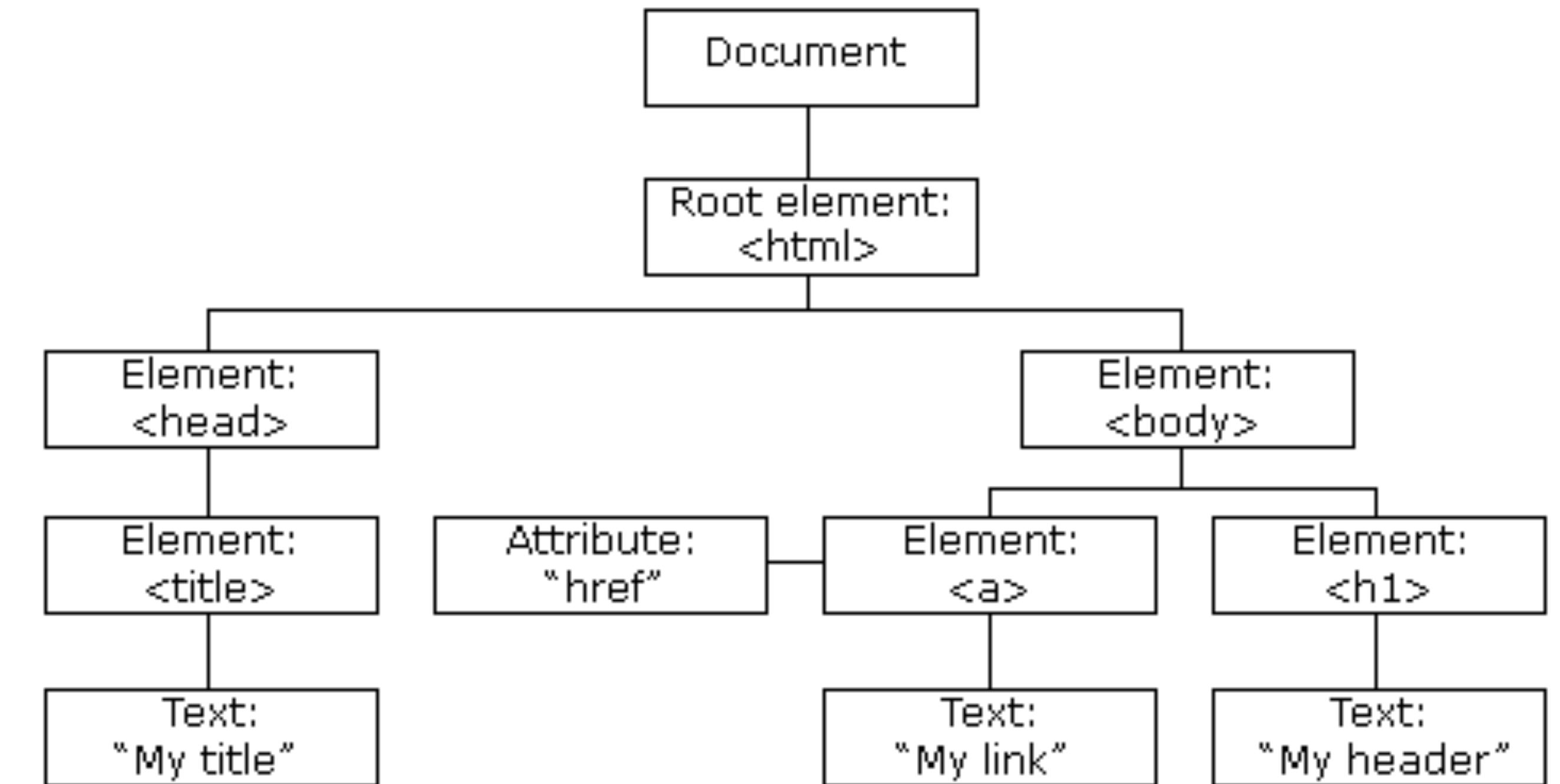
// appending the variable (the string) to the DOM element #content
document.querySelector("#content").innerHTML = message;
```

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

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://javascript.info/dom-nodes>

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

DOM



The screenshot shows the DOM tab of a browser's developer tools. At the top, there are three colored circular icons (red, yellow, green) followed by the word "Elements". Below this, the DOM structure is listed with line numbers:

```
1 <html>
2   <head></head>
3   <body>
4     <div id="app">
5       <h1>Develop. Preview.
6     Ship. 🚀</h1>
7     </div>
8     <script type="text/
9 javascript">...</script>
10    </body>
11  </html>
```

SOURCE CODE (HTML)



The screenshot shows the Source tab of a browser's developer tools. At the top, there are three colored circular icons (red, yellow, green) followed by the file name "index.html". Below this, the raw HTML code is listed with line numbers:

```
1 <html>
2   <head></head>
3   <body>
4     <div id="app"></div>
5     <script type="text/
6 javascript">...</script>
7   </body>
8 </html>
9
10
11
```

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: getElementsBy*

```
<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.getElementsByClassName('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>
```

JS HTML DOM

getElement* or querySelector*?

ES6 +

MODERN JAVASCRIPT

MODERN JAVASCRIPT

LET & CONST
TEMPLATE STRING
ARROW FUNCTIONS
FETCH
PROMISES
ASYNC & AWAIT
FOR OF LOOP
ARRAY.FIND()
ARRAY.MAP()
ARRAY.REDUCE()
ARRAY.FILTER()
ARRAY.SORT()
ARRAY.CONCAT()
DEFAULT PARAMS
DESTRUCTURING OBJECTS
DESTRUCTURING ARRAYS
OBJECT LITERAL
SPREAD OPERATOR
CLASSES
MODULES
IMPORT & EXPORT

Variables

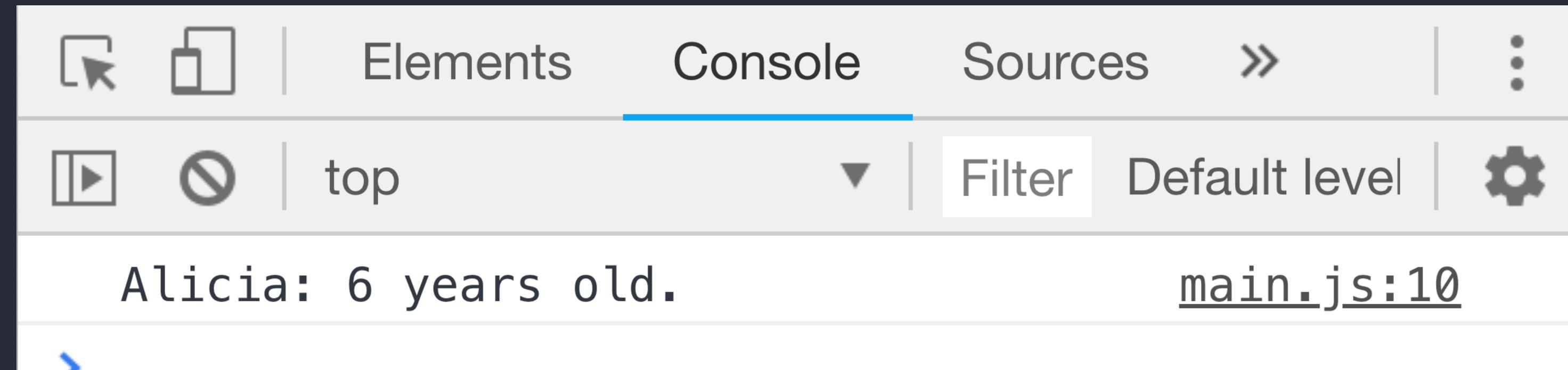
... are used to store data (values, objects, collections) in the memory

Variables

Store data in the memory

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

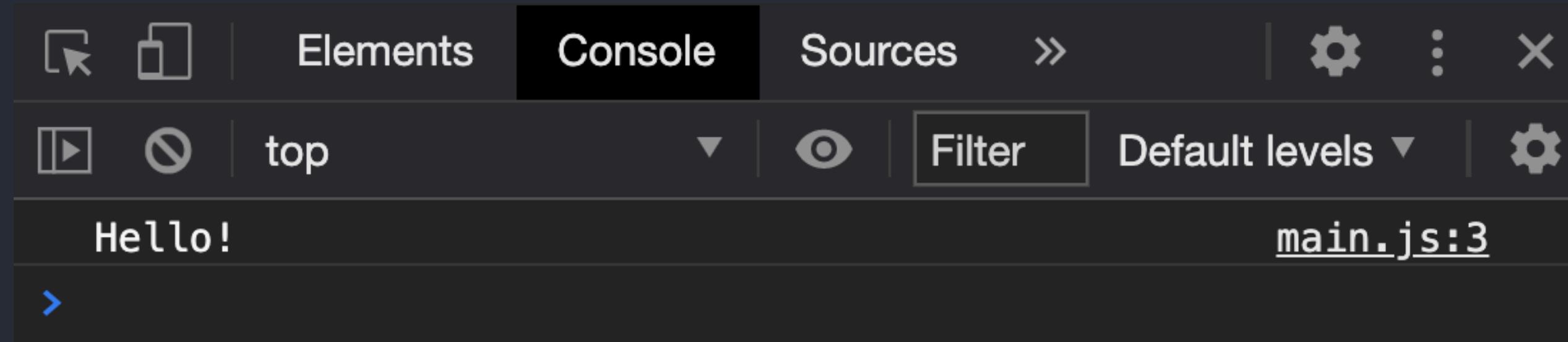
console.log(name + ": " + age + " years old.");
```



Variables

Store data in the memory

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

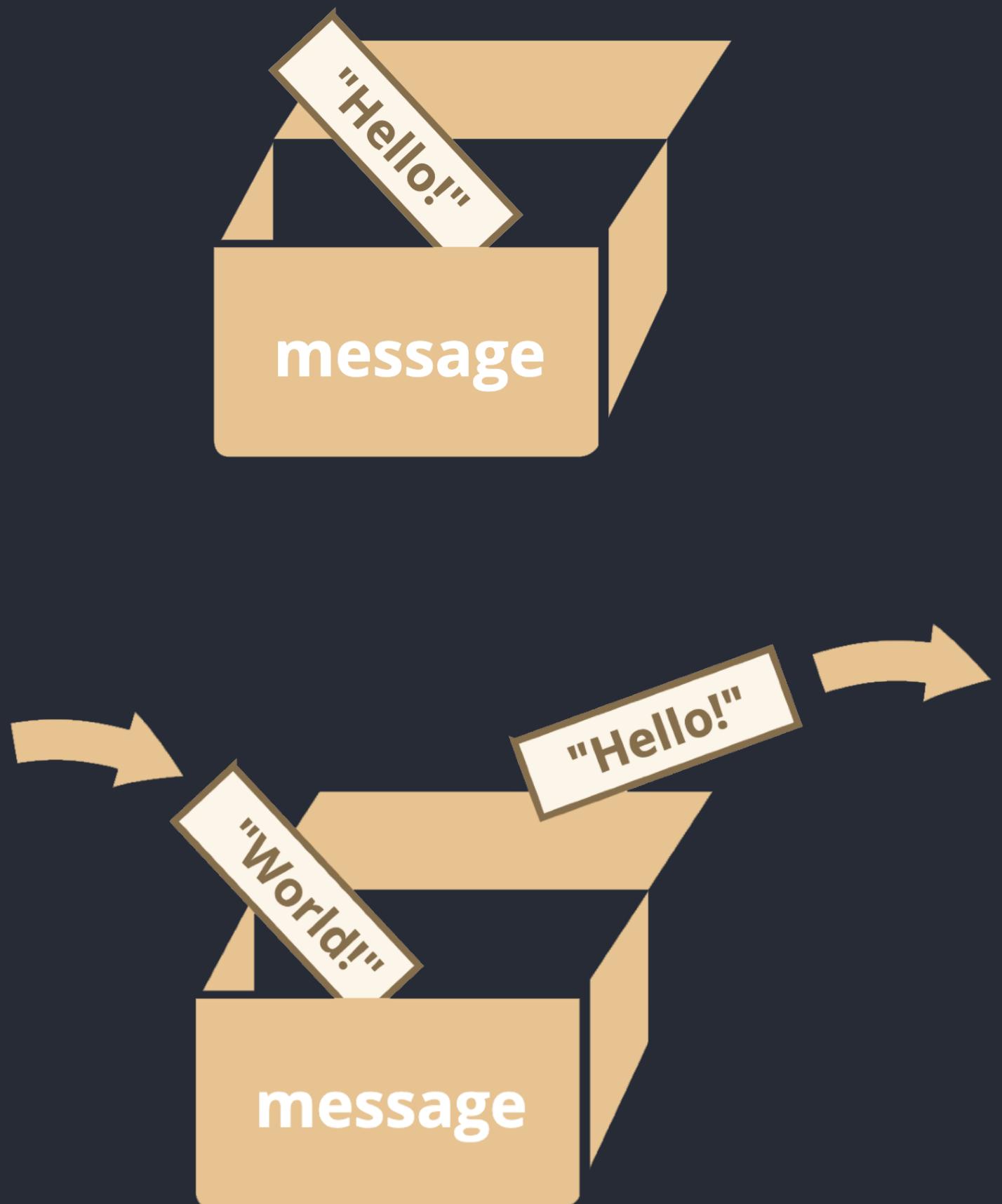
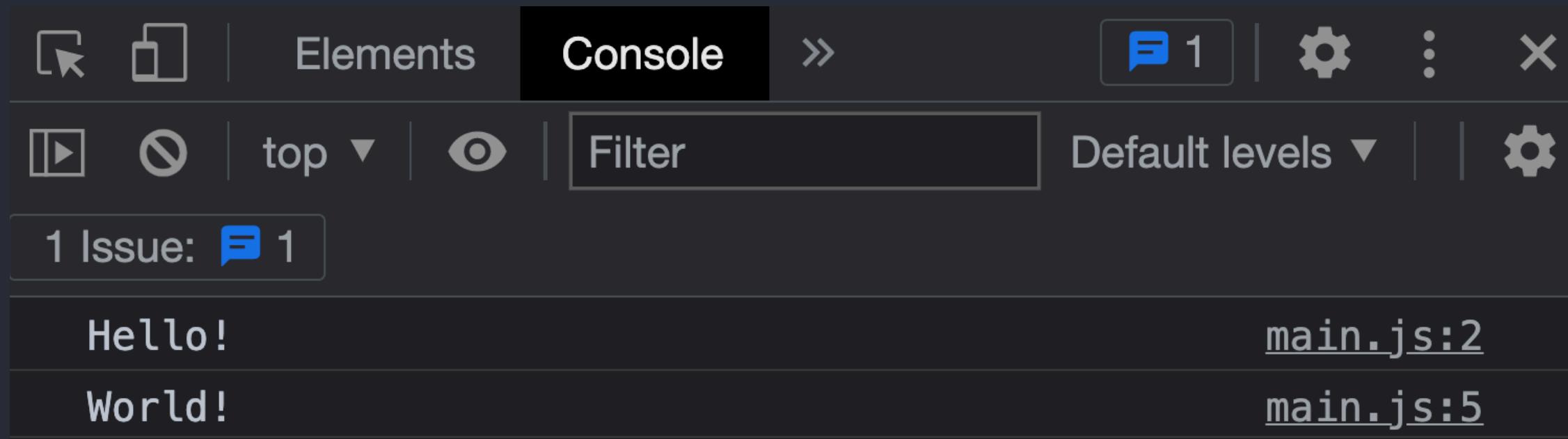


Variables

Store data in the memory

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

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



```
// 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>
```

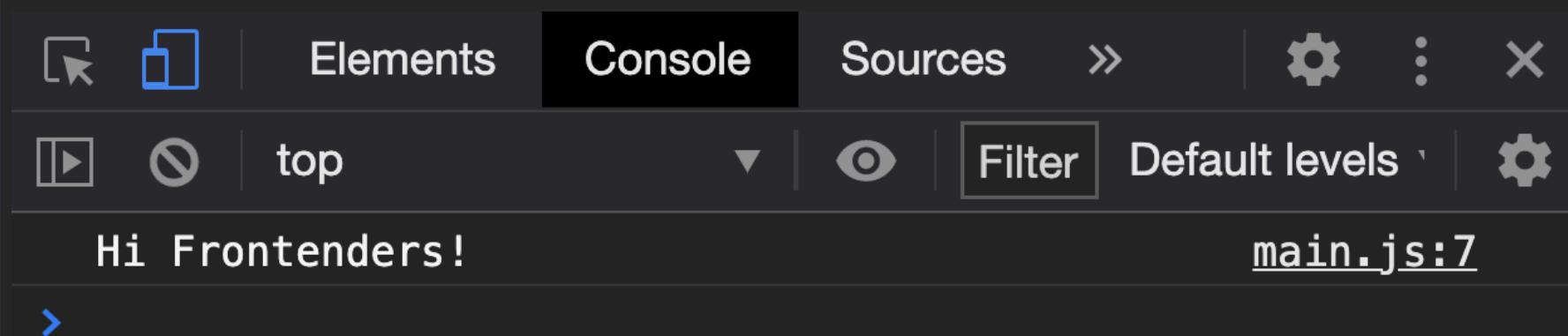


```
// declaring a variable with a value
let message = "Hi Frontenders!"

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

// changing the value of the variable
message = "Hello World";

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



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.

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>

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`.

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

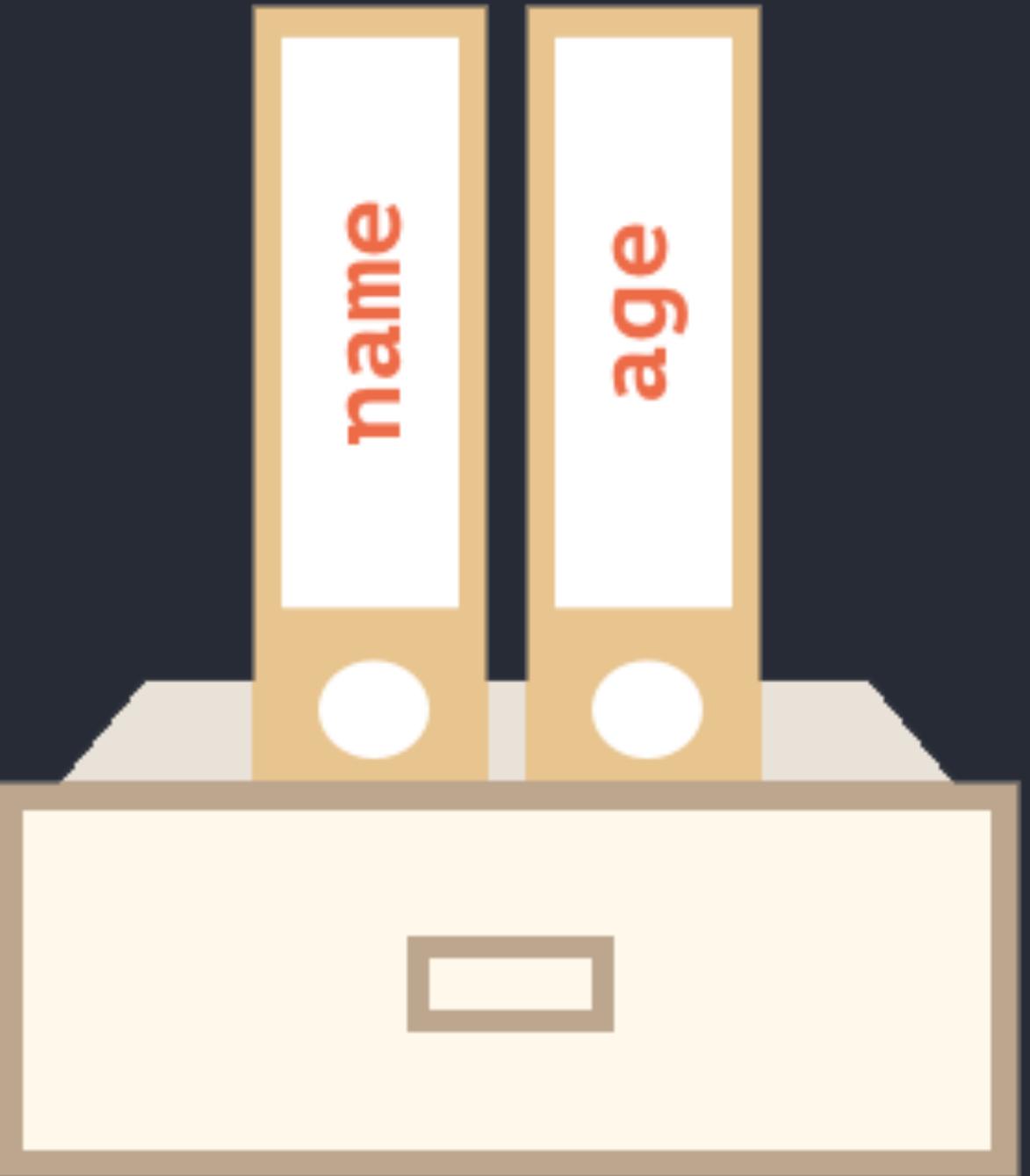
Objects

A set of named values

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

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

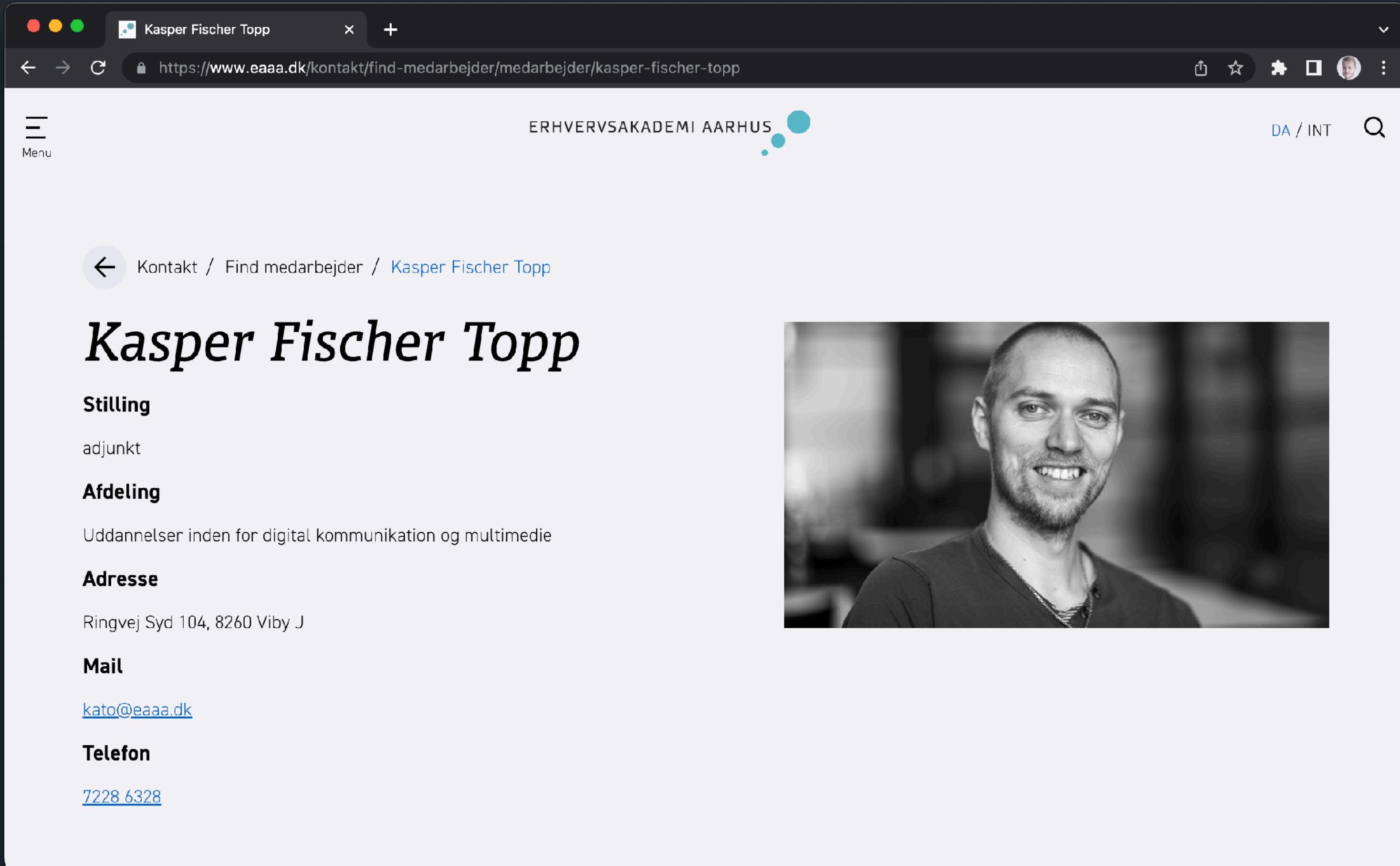
user



Alicia is 6 years old.

main.js:11

Objects



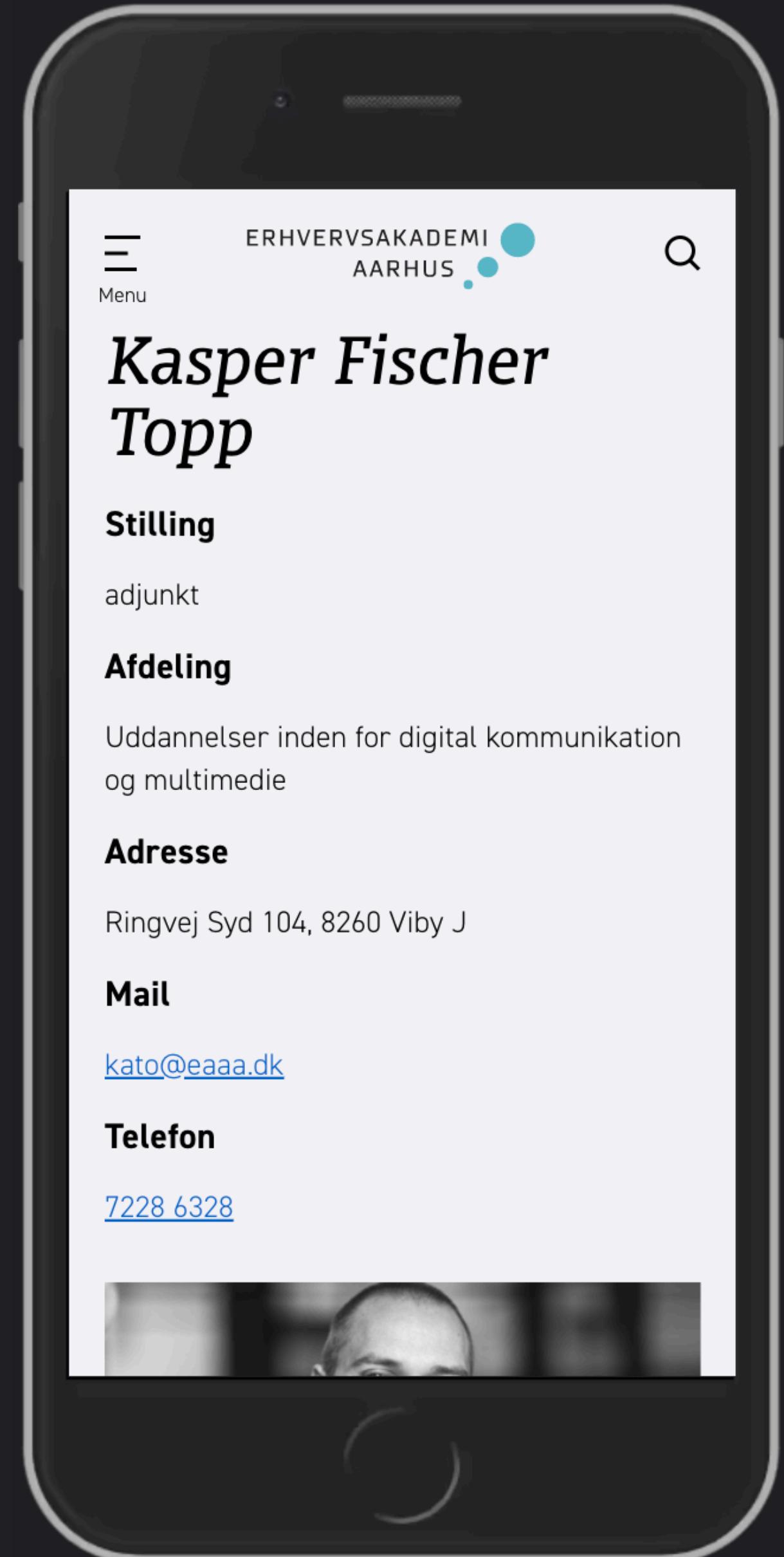
A screenshot of a web browser window displaying a profile page for Kasper Fischer Topp. The browser has a dark theme with red, yellow, and green window control buttons. The title bar shows the page is titled "Kasper Fischer Topp". The address bar contains the URL <https://www.eaaa.dk/kontakt/find-medarbejder/medarbejder/kasper-fischer-topp>. The page header includes the "ERHVERVSAKADEMI AARHUS" logo with three blue dots. The main content area shows a navigation breadcrumb: "Kontakt / Find medarbejder / Kasper Fischer Topp". Below this, the name "Kasper Fischer Topp" is displayed in a large, bold, italicized font. To the right of the name is a black and white portrait photograph of a smiling man with short hair and a beard. On the left side of the profile page, there are sections for "Stilling" (adjunkt), "Afdeling" (Uddannelser inden for digital kommunikation og multimedie), "Adresse" (Ringvej Syd 104, 8260 Viby J), "Mail" (kato@eaaa.dk), and "Telefon" ([7228 6328](tel:72286328)). The browser interface also features a menu icon, a search bar, and language selection "DA / INT".

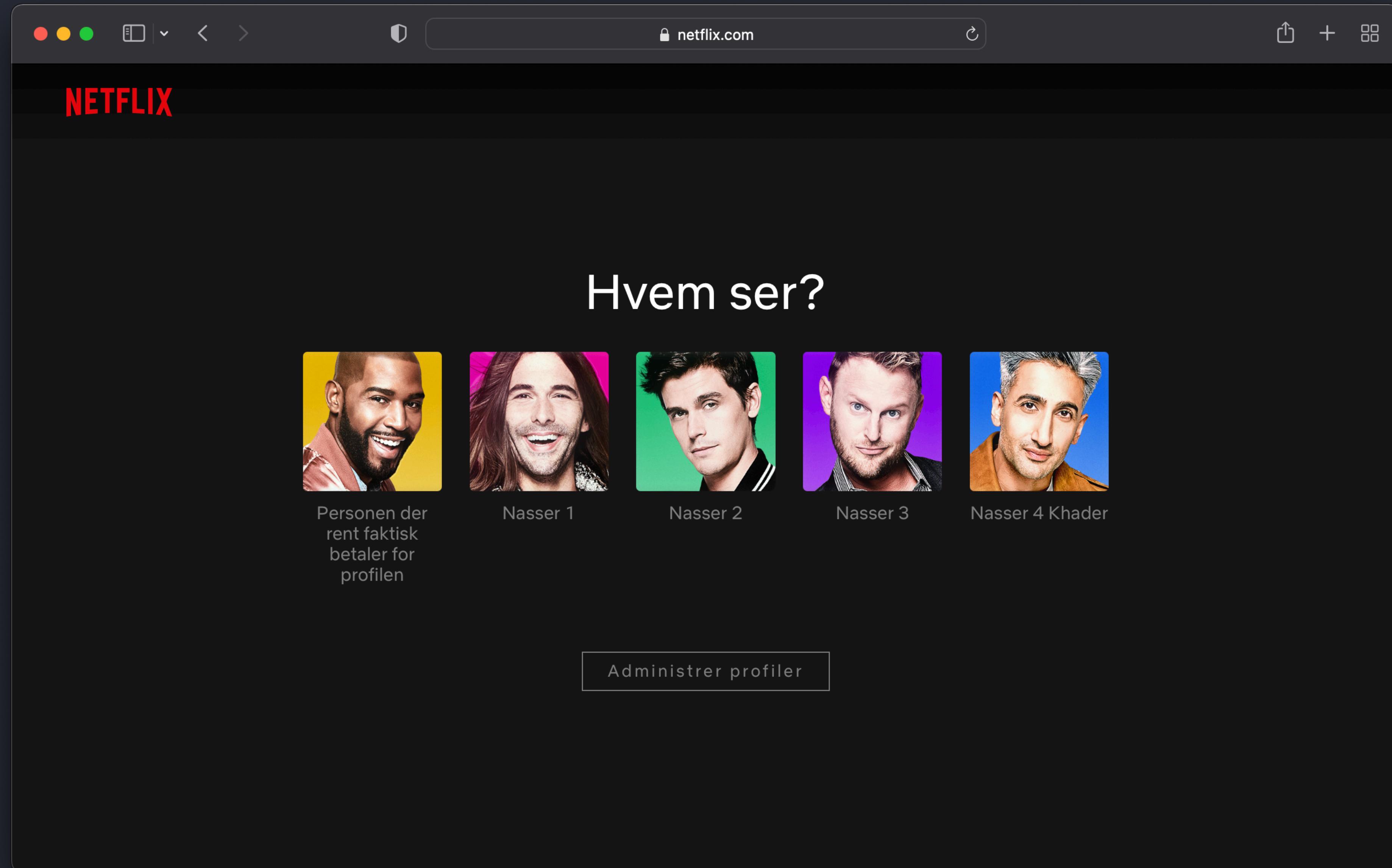
<https://www.eaaa.dk/kontakt/find-medarbejder/medarbejder/kasper-fischer-topp>

Objects

A set of named values

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





● ○ ● | □ | < > ⌂ +

netflix.com

NETFLIX Start Serier Film Nyt og populært Min liste

N SERIE

TOO HOT TO HANDLE

TOP 10 Nr. 4 i Danmark i dag

På paradisets kyst mødes de lækkere singler og mingler. Men der er et tvist. For at vinde den attraktive pengepræmie, må de give afkald på at have sex.

Afspil Mere info

13+

Kun på Netflix

TOO HOT TO HANDLE NYE EPISODER

EMILY IN PARIS

QUEER EYE more than a makeover

The Woman in the House Across the Street From the Girl in the Window

BRIDGERTON NYE EPISODER

Se videre med profilen Nasser 1

the office

TIGER KING

Don't Look UP

JEFFREY EPSTEIN: FILTHY RICH

THE MIND explained

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 | [Share](#)

+ 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

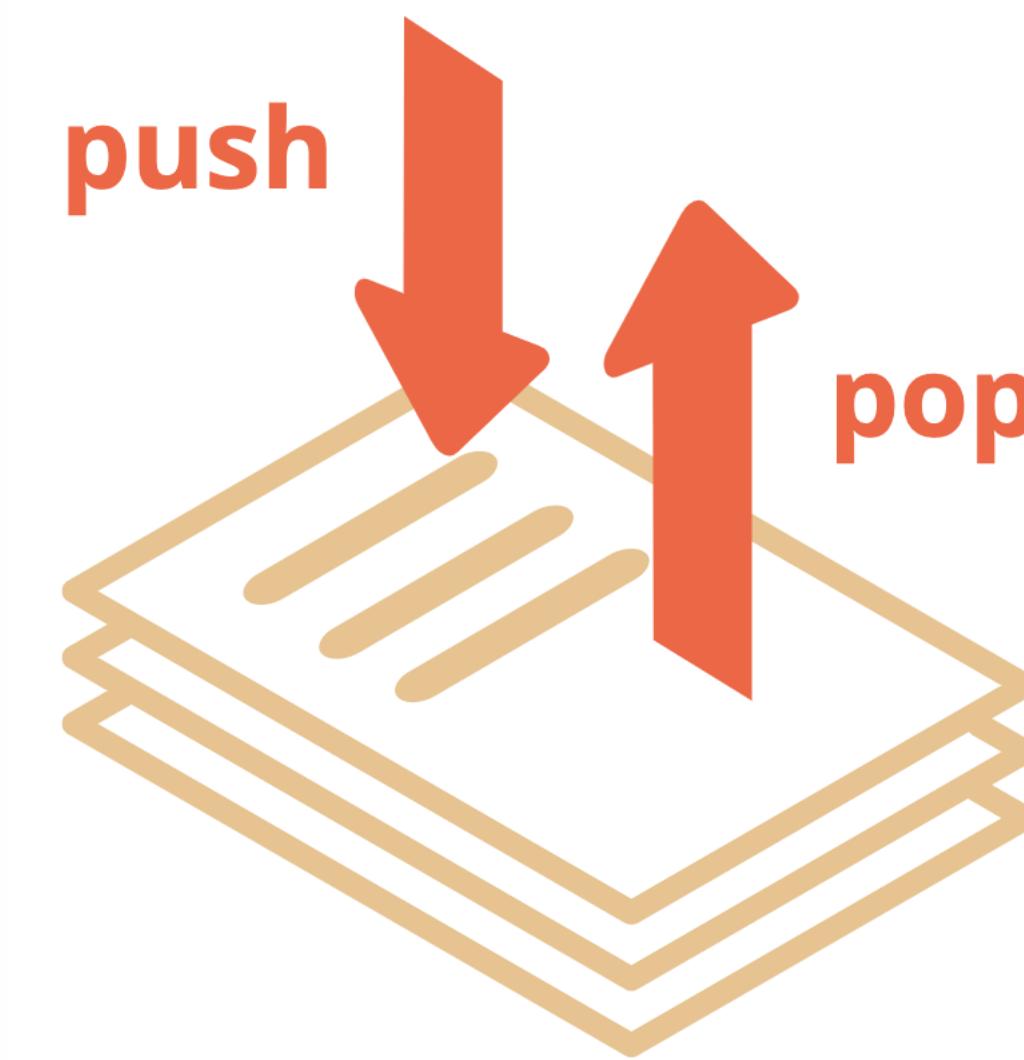
Writers

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

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.

```
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	
Position: Lecturer	<i>Michael Hvidtfeldt</i>
Department/ Multimedia De Digital Conce	
Address: Ringvej Syd 10	Position: Lecturer <i>Birgitte Kirk Iversen</i>
Mail: race@baaa.dk	Department/ Multimedia Di
Phone: 7228 6318	Address: Senior Lecturer Ringvej Syd 10
	Department/programme: Multimedia Design
Mail: mhv@baaa.dk	Address: Sønderhøj 30, 8260 Viby J
Phone: 7228 6328	Mail: bki@baaa.dk
	Phone: 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', initials: 'mlbe'}
  ▶ 2: {name: 'Kim Elkjær Marcher-Jepsen', initials: 'kje'}
  ▶ 3: {name: 'Rasmus Cederdorff', initials: 'race'}
  length: 4
  ► [[Prototype]]: Array(0)
```



Birgitte Kirk Iversen

Senior Lecturer
bki@baaa.dk



Maria Louise Bendixen

Senior Lecturer
mlbe@baaa.dk



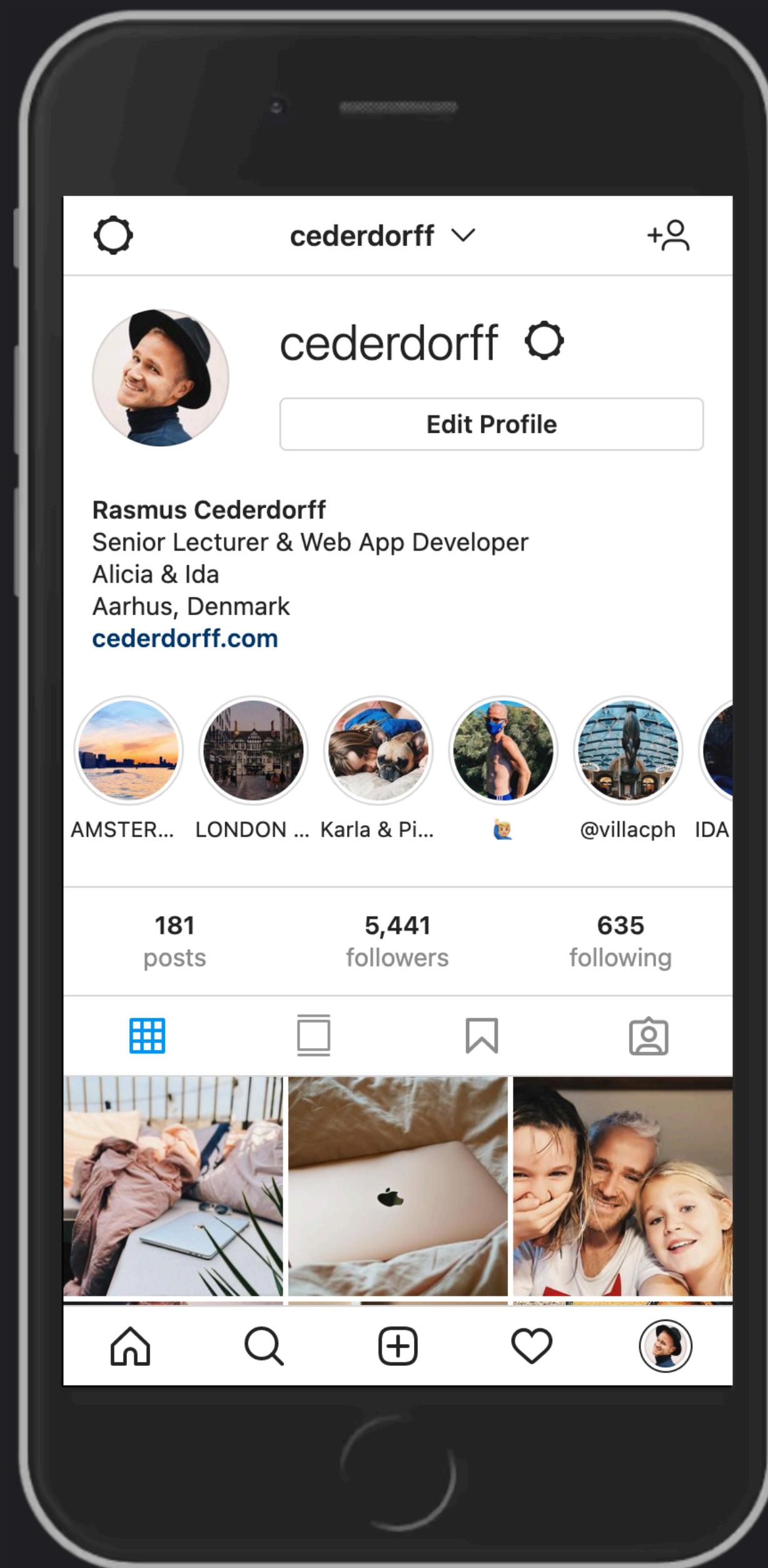
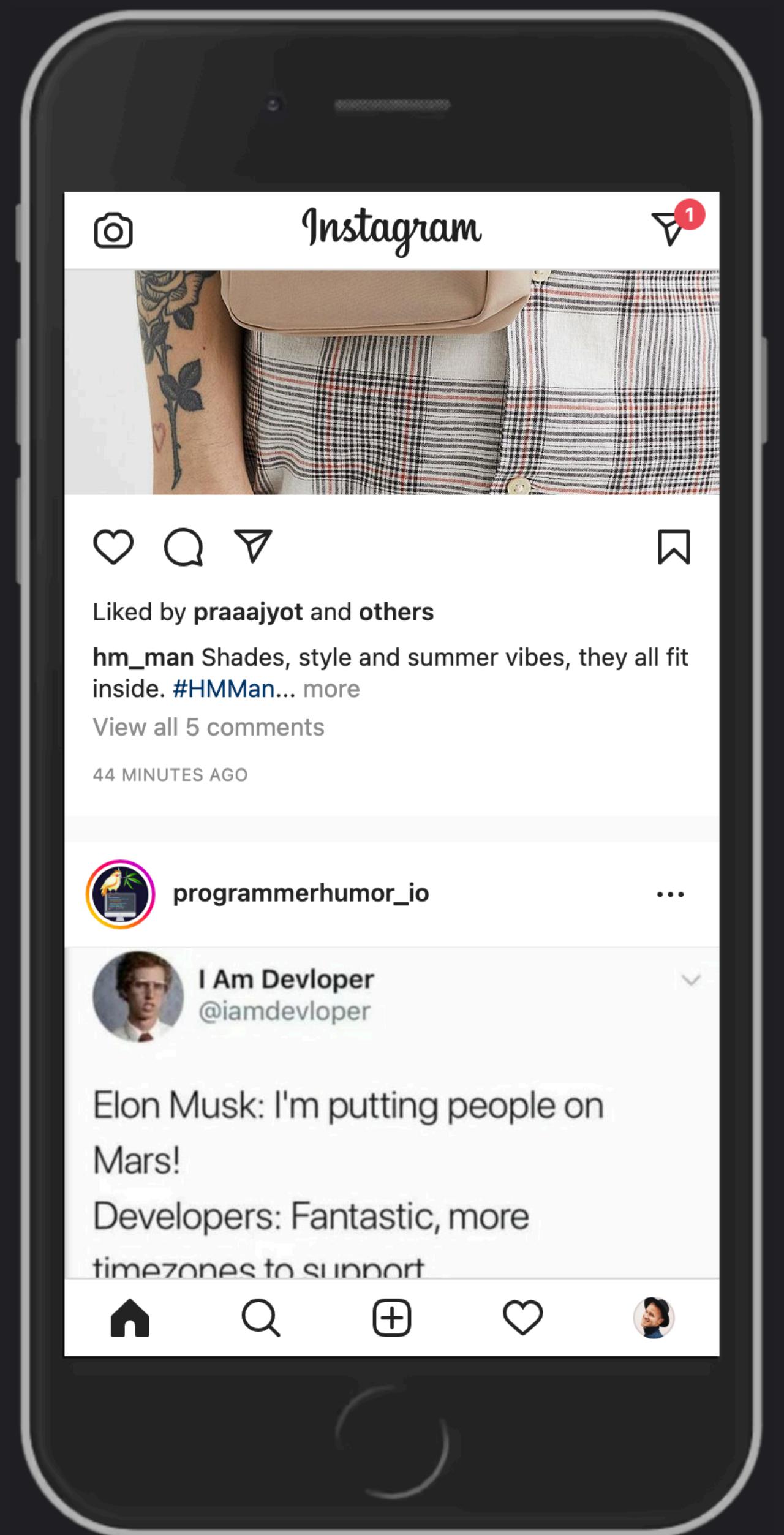
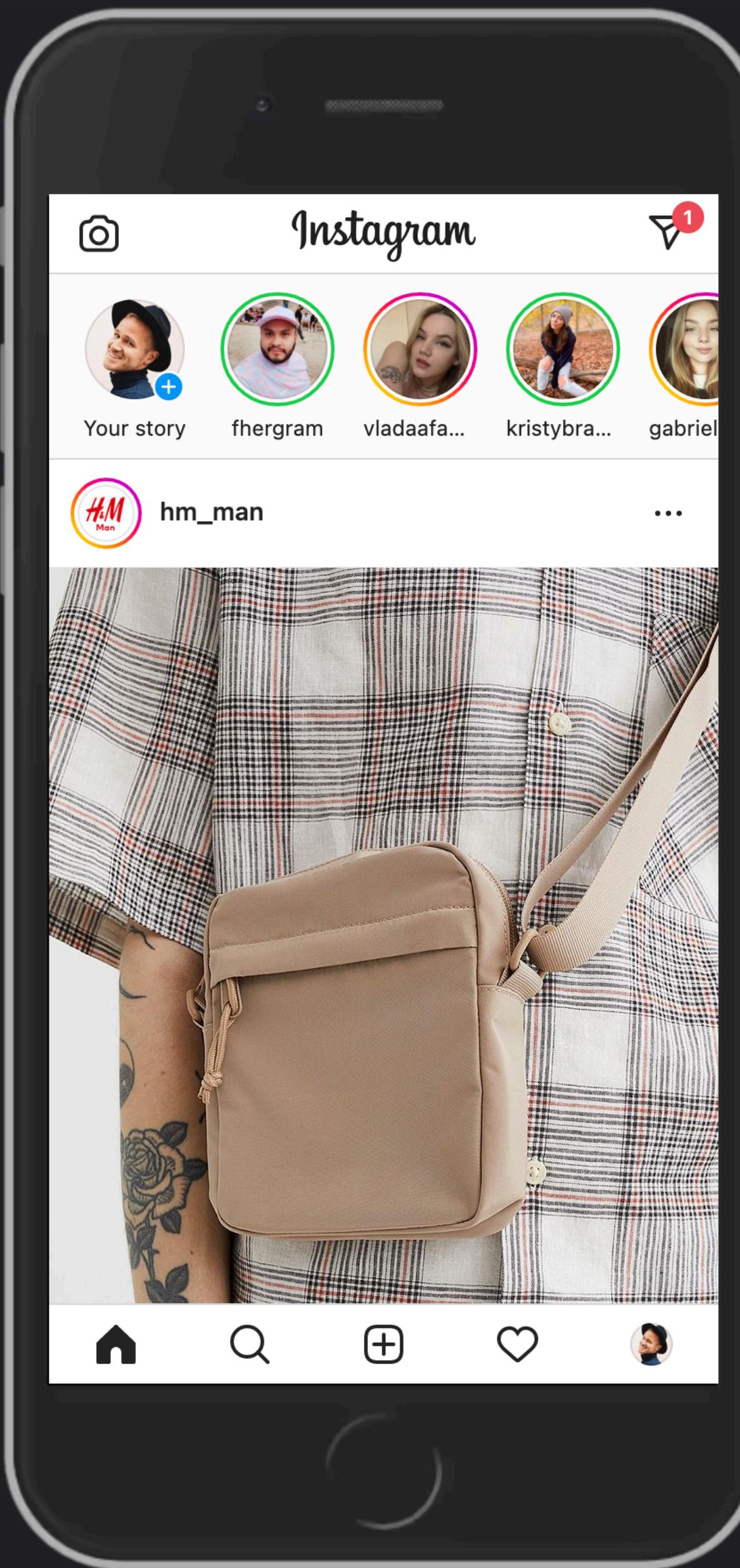
Kim Elkjær Marcher-Jepsen

Lecturer
kje@baaa.dk



Rasmus Cederdorff

Lecturer
race@baaa.dk



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	eaaclbi@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

6 / 157 requests

Elements Components Network

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

Has blocked cookies Blocked Requests 3rd-party requests

5000 ms 10000 ms 15000 ms 20000 ms 25000 ms 30000 ms 35000 ms

property Headers Payload Preview Response Initiator

```
[{"id": "23974", "name": "Clara Juul Birk", "created_at": "2020-08-10T10:45:00+02:00", "email": "eaaclbi@students.eaaa.dk", "sis_user_id": null, "short_name": "Clara Juul Birk", "sortable_name": "Birk, Clara Juul", "integration_id": null, "login_id": "eaaclbi@students.eaaa.dk", "custom_links": []}, {"id": "36267", "name": "Martin Rieper Boesen", "created_at": "2021-07-30T00:46:05+02:00", "email": "eaamrbo@students.eaaa.dk", "sis_user_id": null, "short_name": "Martin Rieper Boesen", "sortable_name": "Boesen, Martin Rieper", "integration_id": null, "login_id": "eaamrbo@students.eaaa.dk", "custom_links": []}, {"id": "29923", "name": "Dan Okkels Brendstrup", "created_at": "2021-07-30T00:46:05+02:00", "email": "dob@eaaa.dk", "sis_user_id": null, "short_name": "Dan Okkels Brendstrup (adjunkt – dob@eaaa.dk)", "sortable_name": "Brendstrup, Dan Okkels", "integration_id": null, "login_id": "dob@eaaa.dk", "custom_links": []}, {"id": "14427", "name": "Rasmus Cederdorff", "created_at": "2021-07-30T00:46:05+02:00", "email": "race@eaaa.dk", "sis_user_id": null, "short_name": "Rasmus Cederdorff", "sortable_name": "Cederdorff, Rasmus", "integration_id": null, "login_id": "race@eaaa.dk", "custom_links": []}, {"id": "41", "name": "Jeffrey David Serio", "created_at": "2021-07-30T00:46:05+02:00", "email": "jds@eaaa.dk", "sis_user_id": null, "short_name": "Jeffrey David Serio", "sortable_name": "Serio, Jeffrey David", "integration_id": null, "login_id": "jds@eaaa.dk", "custom_links": []}, {"id": "24043", "name": "Charlotte Meng Emanuel Dyrholm", "created_at": "2021-07-30T00:46:05+02:00", "email": "eaacmed@students.eaaa.dk", "sis_user_id": null, "short_name": "Charlotte Meng Emanuel Dyrholm", "sortable_name": "Dyrholm, Charlotte Meng Emanuel", "integration_id": null, "login_id": "eaacmed@students.eaaa.dk", "custom_links": []}, {"id": "23978", "name": "Jeppe Frik", "created_at": "2020-08-03T10:45:00+02:00", "email": null, "sis_user_id": null, "short_name": "Jeppe Frik", "sortable_name": "Frik, Jeppe", "integration_id": null, "login_id": null, "custom_links": []}, {"id": "23963", "name": "Daniel Tjerrild Gamborg", "created_at": "2021-07-30T00:46:05+02:00", "email": null, "sis_user_id": null, "short_name": "Daniel Tjerrild Gamborg", "sortable_name": "Gamborg, Daniel Tjerrild", "integration_id": null, "login_id": null, "custom_links": []}, {"id": "23992", "name": "Casper Hedegaard Hansen", "created_at": "2021-07-30T00:46:05+02:00", "email": null, "sis_user_id": null, "short_name": "Casper Hedegaard Hansen", "sortable_name": "Hansen, Casper Hedegaard", "integration_id": null, "login_id": null, "custom_links": []}, {"id": "36266", "name": "Morten Gedsted Hansen", "created_at": "2021-07-30T00:46:05+02:00", "email": null, "sis_user_id": null, "short_name": "Morten Gedsted Hansen", "sortable_name": "Hansen, Morten Gedsted", "integration_id": null, "login_id": null, "custom_links": []}, {"id": "23980", "name": "Anders Husted", "created_at": "2020-08-03T10:45:00+02:00", "email": null, "sis_user_id": null, "short_name": "Anders Husted", "sortable_name": "Husted, Anders", "integration_id": null, "login_id": null, "custom_links": []}, {"id": "23531", "name": "Søren Bo Jørgensen", "created_at": "2021-07-30T00:46:05+02:00", "email": null, "sis_user_id": null, "short_name": "Søren Bo Jørgensen", "sortable_name": "Jørgensen, Søren Bo", "integration_id": null, "login_id": null, "custom_links": []}]
```

Objects? Arrays?

The screenshot shows the DR Nyheder - Breaking - TV website. At the top, there are six thumbnail images with titles: DR1: Løvens Hule, DR3: Nationens stærkste, P1: LSD kælderen, DR LYD: Annas Margrethe, DR3: Du fucker med de forkerte, and A Very British Scandal. Below this, a section titled "Seneste nyt" displays three news items: "EU klager over Kinas hårde kurs over for Litauen" (5 MIN. SIDEN), "Børn og skoleelever opfordres stadig til to ugentlige coronatest" (13 MIN. SIDEN), and "England skrætter størstedelen af coronarestriktionerne fra i dag" (25 MIN. SIDEN). The main content area features a large graphic of a face wearing a mask, surrounded by a COVID-19 test kit, hand sanitizer, and a bottle of liquid. Below the graphic, a headline reads "15 lande bakker Danmark op: Danske soldater skal blive i Mali". At the bottom, a red banner says "Liveblog" followed by three dots.

The screenshot shows the Erhvervsakademiet website. The header includes the logo "ERHVERVSAKADEMIAARHUS" and a search bar. The page displays eight vocational training programs in a grid format:

- Byggekoordinator (lukket)**: 2-årig erhvervsakademimuddannelse. Uddannelsen optager ikke nye studerende fra og med 2022.
- Datamatiker**: 2-årig erhvervsakademimuddannelse. Få en bred viden inden for systemudvikling og programmering, og bliv klar til et job med udvikling, implementering og drift af IT-systemer i virksomheden.
- Financial controller**: 2-årig erhvervsakademimuddannelse. Bliv klar til et job i en økonomiafdeling eller som revisor. Du får karriere inden for fx regnskab, likviditetsstyring, årsrapporter, skat, moms og husholdningen.
- Finansøkonom**: 2-årig erhvervsakademimuddannelse. Bliv rustet til en karriere i investerings-, forsikrings- eller ejendomsbranchen - eller i kreditforsørger. Du bliver en god rådgiver, der løser kundebehov med din virksomheds ressourcer.
- It-teknolog**: 2-årig erhvervsakademimuddannelse. Brænder du for at følge med i udviklingen af den nyeste teknologi? Så får karriere inden for computer-, server- og netværksteknologi eller elektronik.
- Jordbrugsteknolog**: 2-årig erhvervsakademimuddannelse. Bliv specialist inden for miljø og natur, jordbrugsekonomi og driftsledelse, husdyrproduktion, landskab og anlæg og planteproduktion.
- Laborant**: 2½-årig erhvervsakademimuddannelse. Få job i et kontrol-, forsknings- eller udviklingslaboratorium og arbejd inden for fx fødevare sikkerhed, miljøbeskyttelse, medicinalindustrien eller bioteknologi.
- Markedsføringøkonom**: 2-årig erhvervsakademimuddannelse. For dig, der vil arbejde med markedsføring, salg og kommunikation. Gør karriere som fx indkøber, sæger, marketingkoordinator eller projektleder. Vi tilbyder specialiseringer inden for helsekønns design.
- Miljøteknolog**: 2-årig erhvervsakademimuddannelse. For dig, der vil arbejde med kvalitets sikring og forbedring af virksomhedens miljøindsats. Som miljøteknolog sikrer du, at virksomheden er beredvilligt mindsker forurenningen om overhinder.
- Multimediedesigner**: 2-årig erhvervsakademimuddannelse. Få viden om user interface design (UI), user experience design (UX), programmering og formændsforståelse. Bliv klar til job som fx frontendumulværker UX-designer, minid manager eller

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
```

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/Javascript/Javascript Loop For.ASP](https://www.w3schools.com/js/js_loop_for.asp)

[HTTPS://JAVASCRIPT.INFO/ARRAY#LOOPS](https://javascript.info/array#loops)

[HTTPS://JAVASCRIPT.INFO/WHILE-FOR](https://javascript.info/while-for)

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.

Array Methods

.FIND()

.MAP()

.REDUCE()

.FILTER()

.SORT()

.CONCAT()

...

■ ■ ■ ■	.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
```

FUNCTIONS

...A BLOCK OF CODE TO PERFORM A SPECIFIC
TASK & TO MAKE OUR CODE REUSABLE

A WAY OF STORING OUR CODE SO WE CAN USE
IT AGAIN AND AGAIN AND REUSE IT

BEST PRACTICE: WRITE REUSABLE CODE

FUNCTIONS

FUNCTION DECLARATION

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

```
log("Hi Frontenders!");
```

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");
```

FUNCTIONS

FUNCTION DECLARATION

```
function append(htmlTemplate, idOfElement) {  
    console.log(htmlTemplate);  
    document.getElementById(idOfElement).innerHTML += htmlTemplate;  
    alert("Yaaaaah, you did it!");  
}  
  
append("<h2>Hi Frontenders!</h2>", "content");
```

FUNCTIONS

FUNCTION DECLARATION

```
function append(htmlTemplate, idOfElement) {  
    console.log(htmlTemplate);  
    document.getElementById(idOfElement).innerHTML += htmlTemplate;  
    alert("Yaaaaah, you did it!");  
}  
  
append("<h2>Hi Frontenders!</h2>", "content");
```

The name of the function

Parameters

Body of the function
(code block)

How to call the function

The diagram illustrates the structure of a JavaScript function declaration. It shows a code block with two main parts: the function definition and its invocation. The function definition starts with 'function' followed by the name 'append'. It takes two parameters: 'htmlTemplate' and 'idOfElement'. The body of the function contains three statements: logging the template to the console, setting the innerHTML of the specified element, and displaying an alert. An annotation 'The name of the function' points to the word 'append'. Another annotation 'Parameters' points to the two variables listed after the function name. A large bracket on the right side of the body is labeled 'Body of the function (code block)'. A final annotation 'How to call the function' points to the line where 'append' is invoked with its arguments.

FUNCTIONS

ARRAY & 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



Michael Hvidtfeldt

Senior Lecturer
mhv@baaa.dk



Rasmus Cederdorff

Lecturer
race@baaa.dk

FUNCTIONS

```
function logPersons(persons) {  
  for (var i = 0; i < persons.length; i++) {  
    console.log(persons[i]);  
  }  
}
```

FUNCTION DECLARATION

```
const logPersons = function(persons) {  
  for (var i = 0; i < persons.length; i++) {  
    console.log(persons[i]);  
  }  
}
```

FUNCTION EXPRESSION

```
const logPersons = (persons) => {  
  for (var i = 0; i < persons.length; i++) {  
    console.log(persons[i]);  
  }  
}
```

ARROW FUNCTION

`TEMPLATE STRING`

BACKTICK STRING / TEMPLATE LITERALS

...EMBED VARIABLES AND EXPRESSIONS IN A STRING

<https://scrimba.com/p/p4Mrt9/c4vJdha>

`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.

[main.js:10](#)

Alicia is 6 years old.

[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



Michael Hvidtfeldt

Senior Lecturer
mhv@baaa.dk



Rasmus Cederdorff

Lecturer
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>`;  
  }  
}
```

`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>`;
  }
}
```



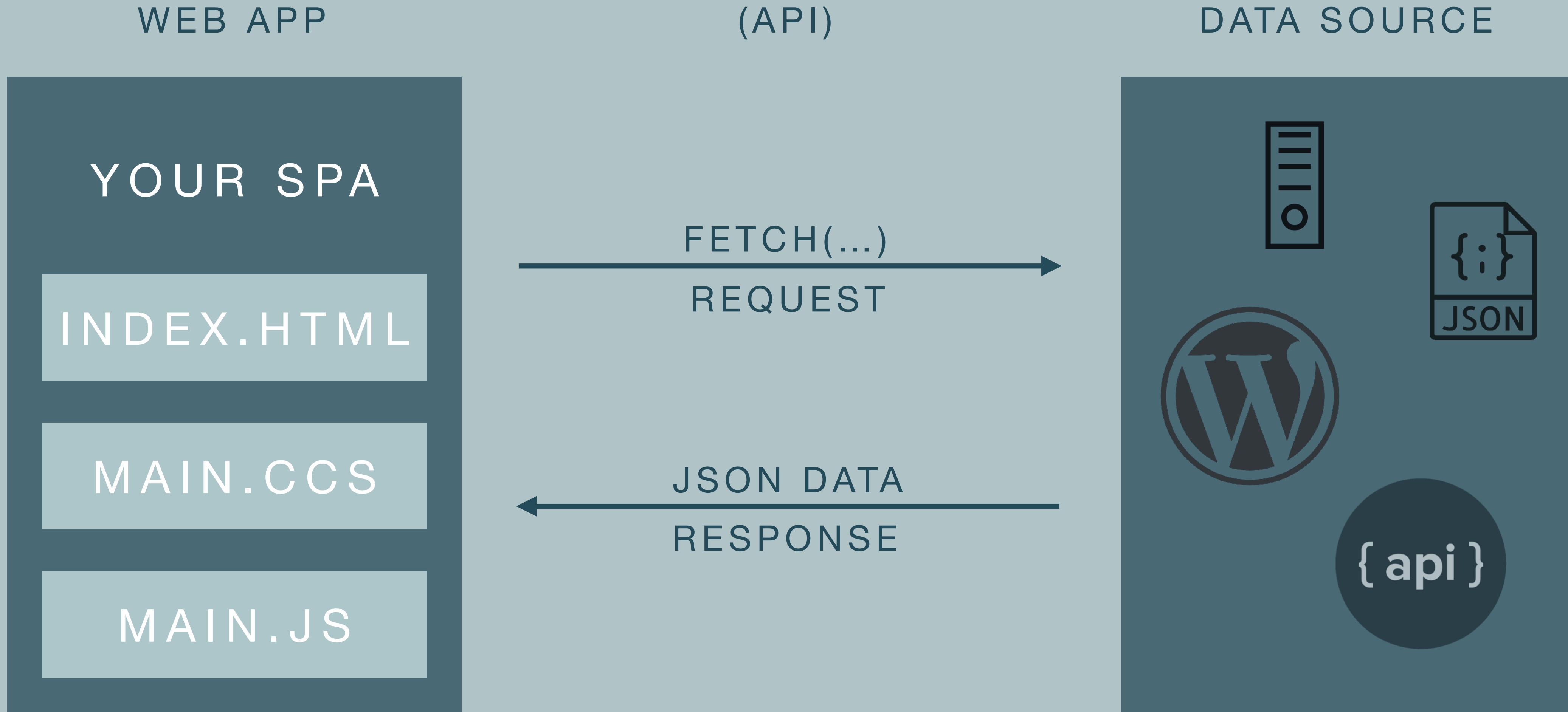
```
function appendTeachers(teachers) {
  for (let teacher of teachers) {
    console.log(teacher);
    document.querySelector("#grid-teachers").innerHTML += /*html*/
      <article>
        <img src='${teacher.img}'>
        <h3>${teacher.name}</h3>
        ${teacher.position}<br>
        <a href='mailto:${teacher.mail}'>${teacher.mail}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

FETCH(...)

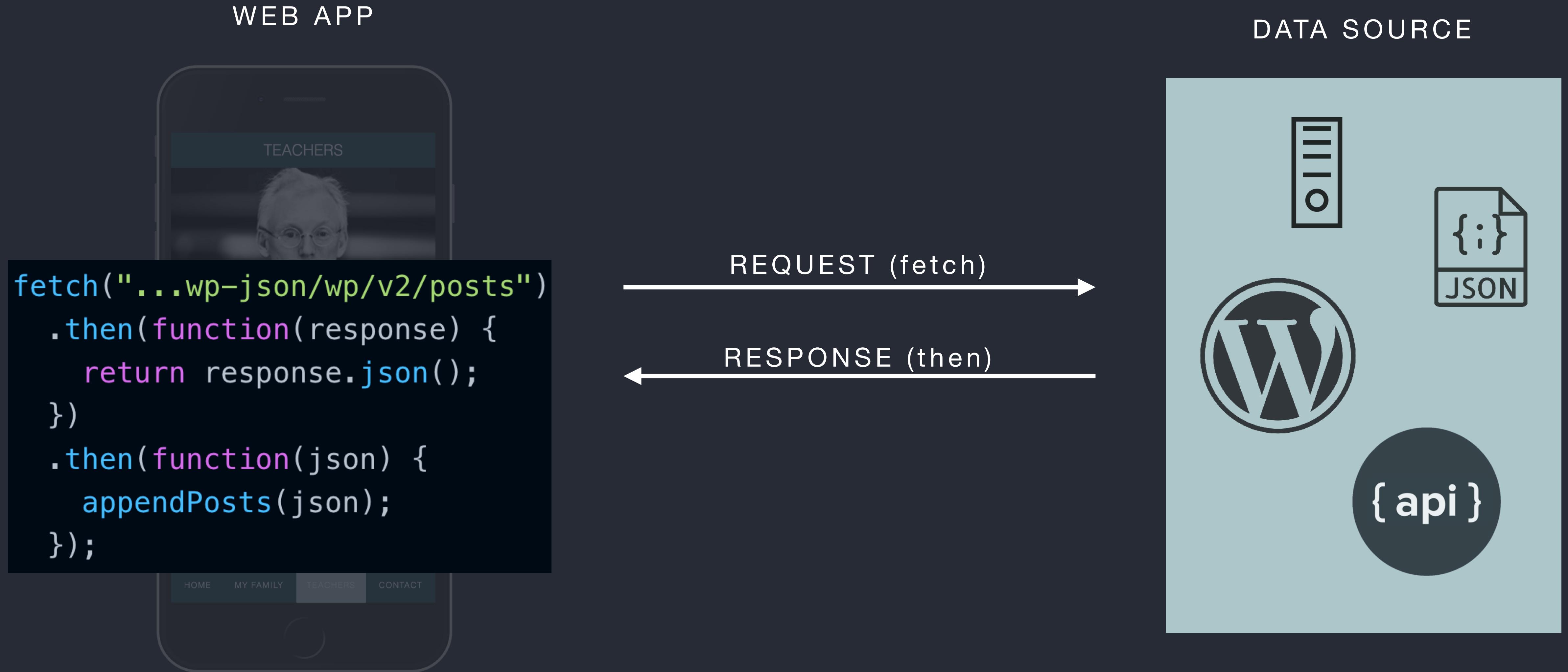
HTTP REQUEST IN JAVASCRIPT

...A WAY TO GET & POST DATA FROM & TO A DATA SOURCE

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>

`;
  }
  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"
}]
```

```
/*
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"
},
{
```

FETCH

... IN THREE DIFFERENT WAYS

```
// 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();
```

Computer science student



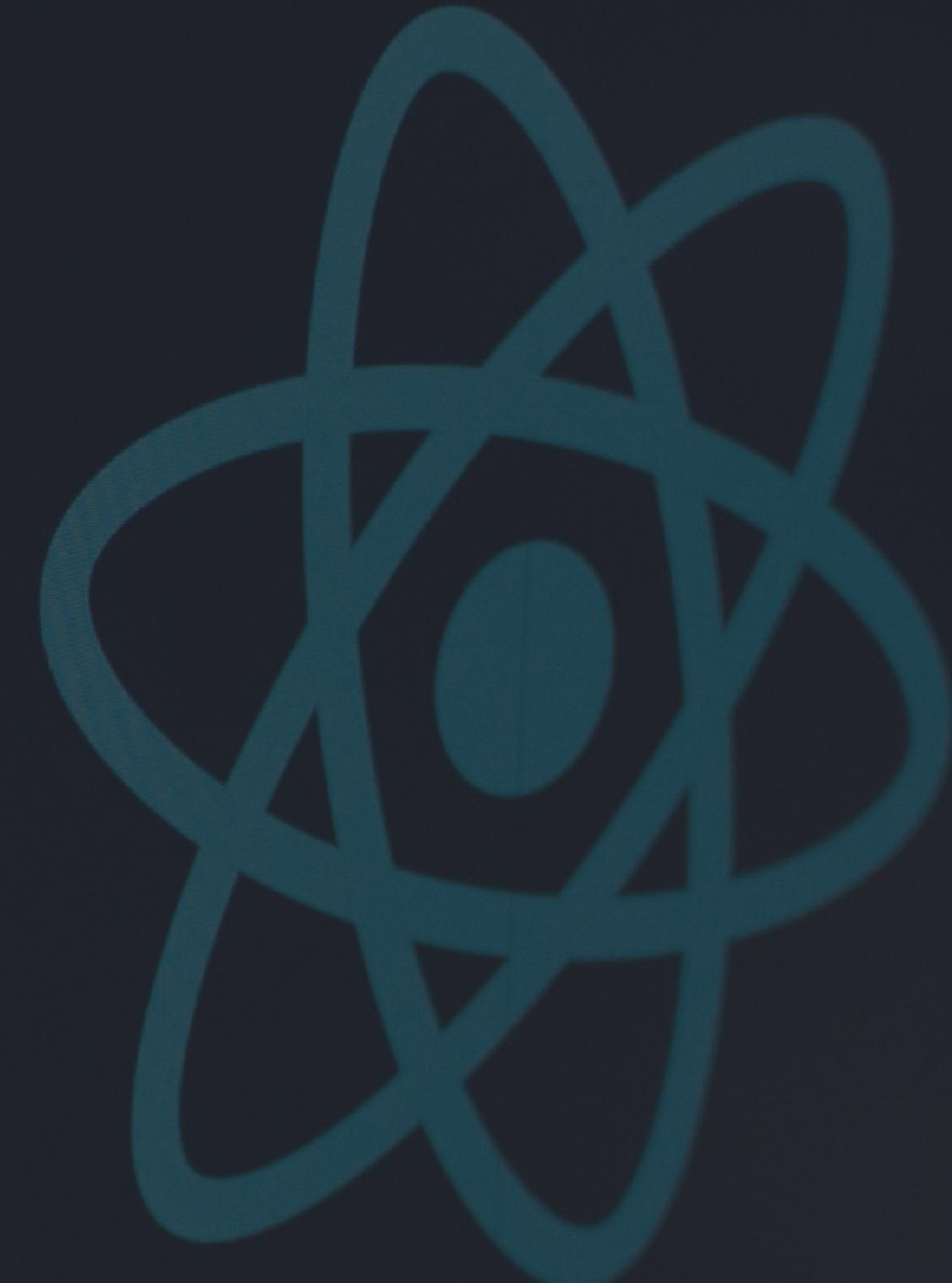
Senior developer, 10+ years experience



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

React

React slides in Canvas/ GitHub



Edit src/App.js and save to reload
Learn React

Code Every Day

