

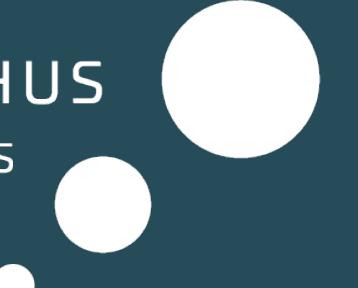


# INTRODUCTION TO THE COURSE

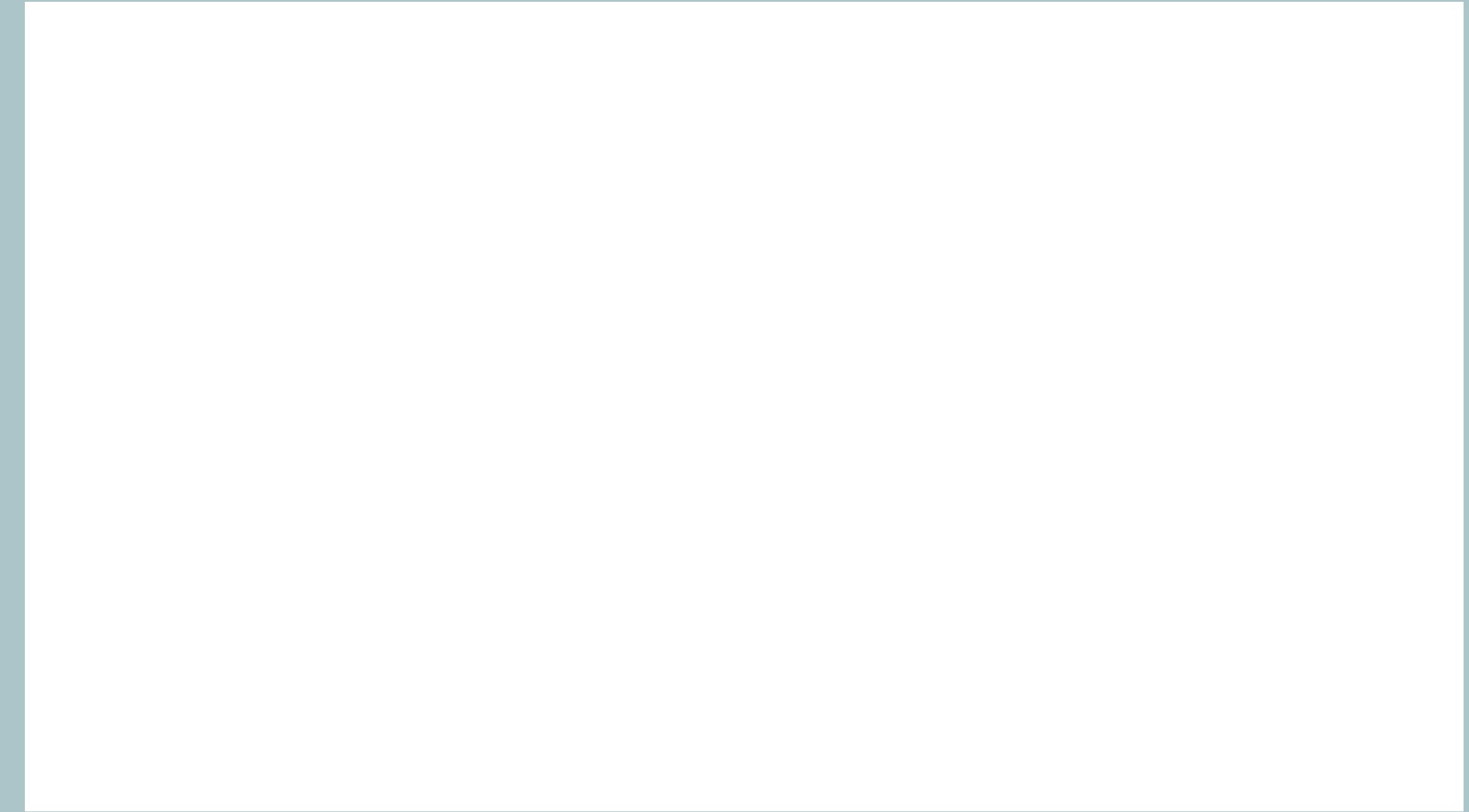
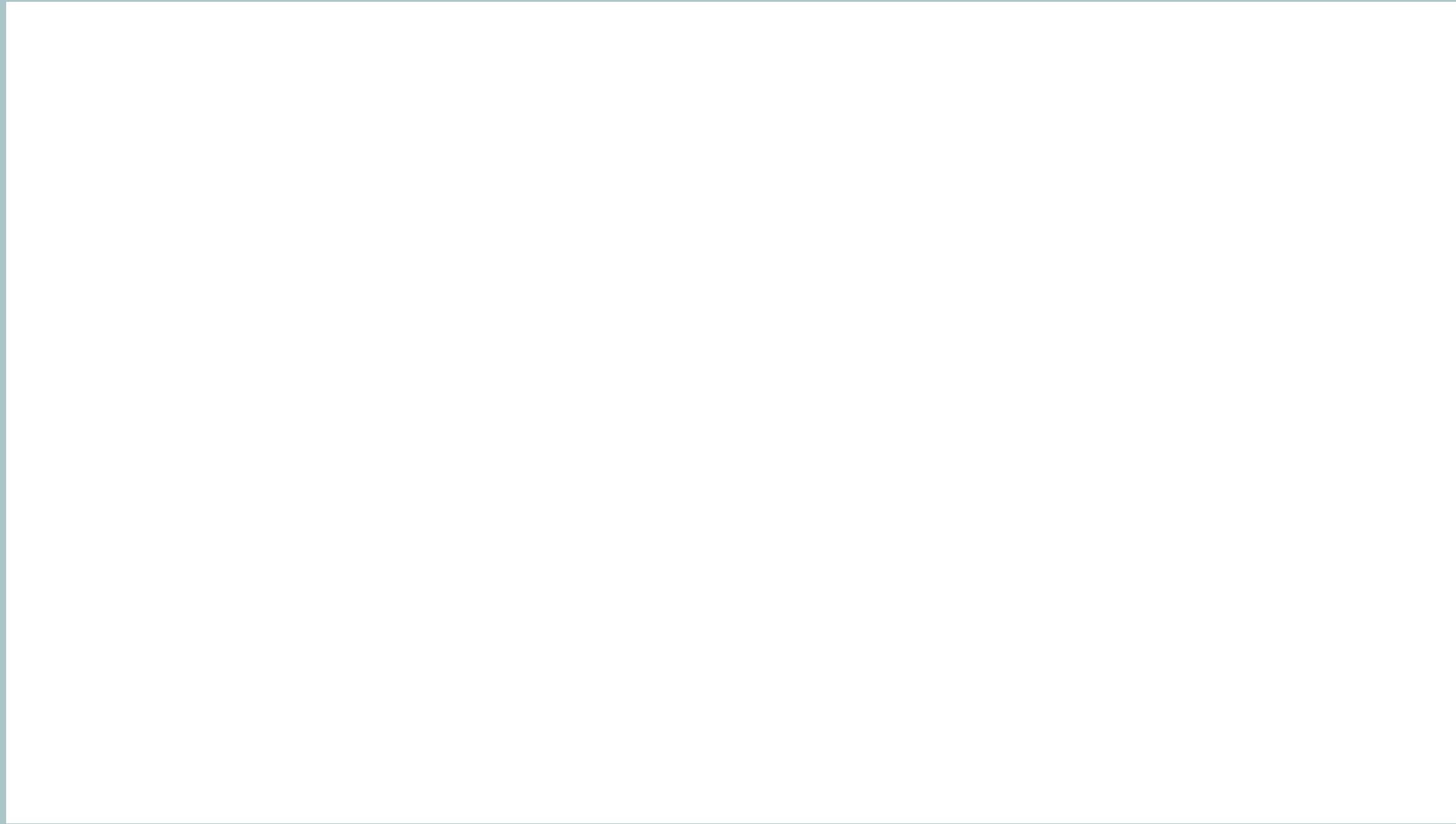
# HTML, CSS, JAVASCRIPT & WEB APPS

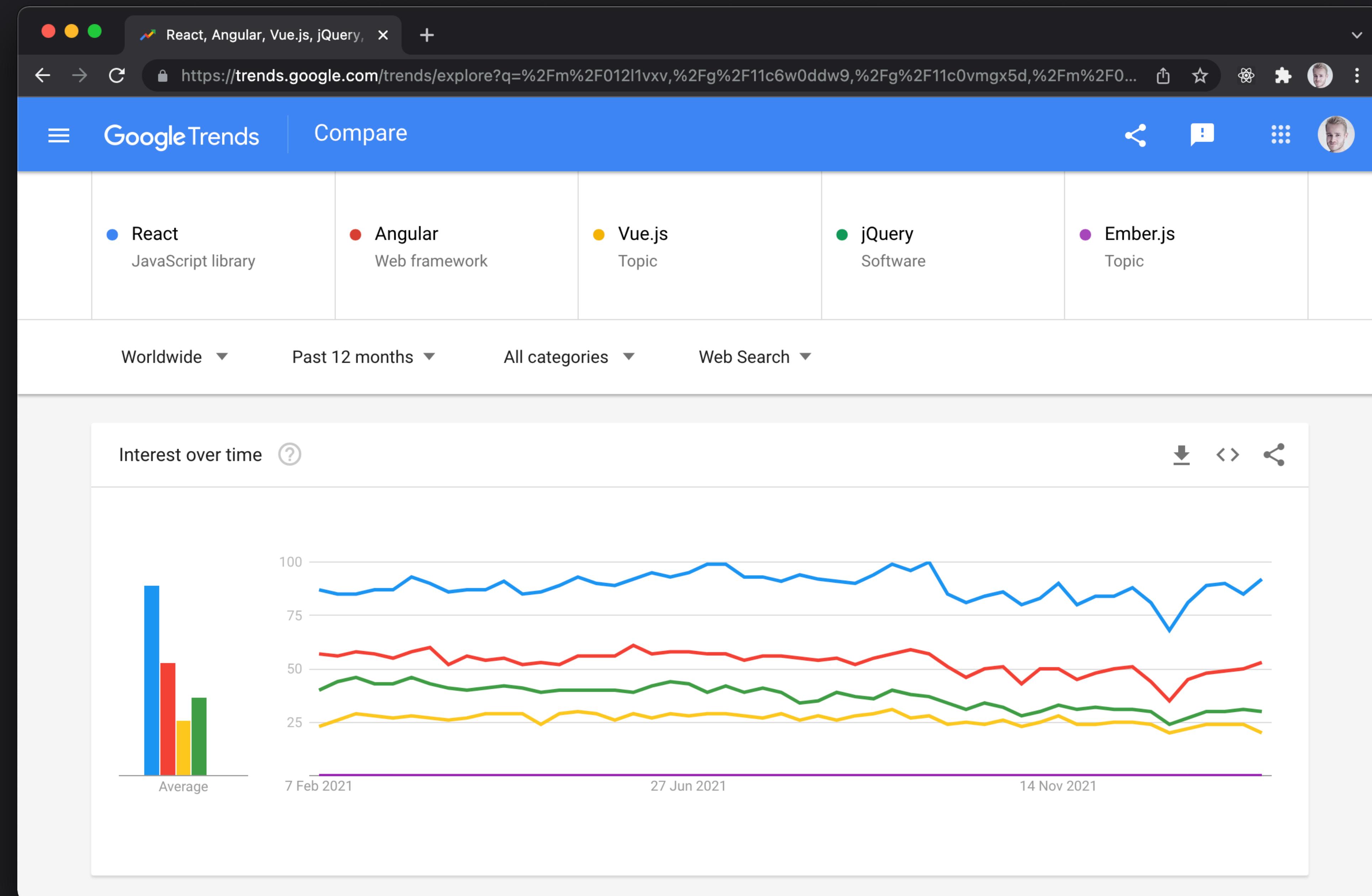
FRONTEND PROGRAMMING

ERHVERVSAKADEMI AARHUS  
BUSINESS ACADEMY AARHUS



# REMINDER







# AGENDA

Introduction to the course

Frontend Dev Stack

CSS layouts

JavaScript Basics

Canvas Users Case

# TODAY'S FUN



Test Firebase Post App

Setup Dev Environment

Grid Layout & Page Layout

Grid & Card Layout

JS HTML DOM #1 & #2

Canvas Users Case #1, #2 & #3

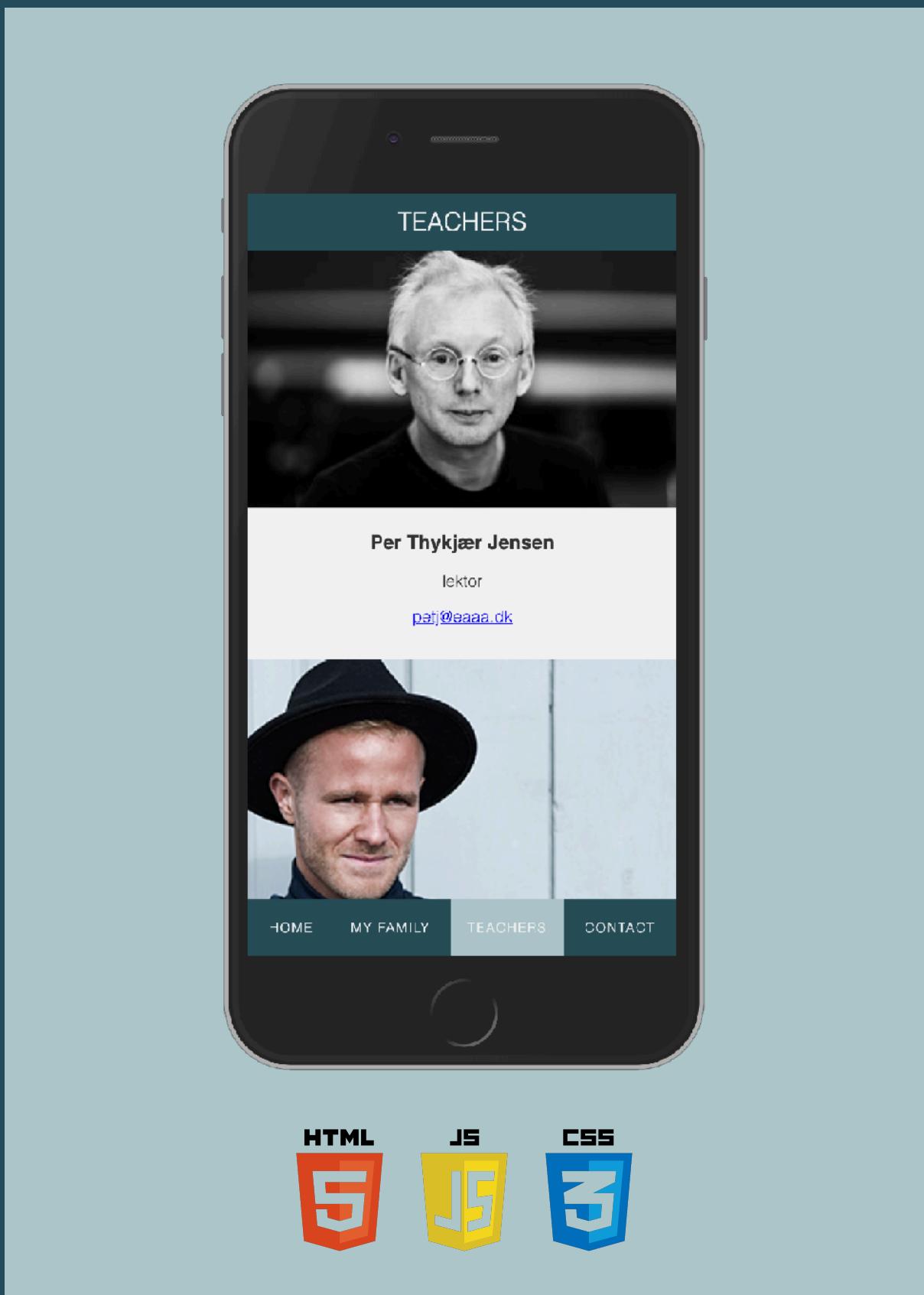
Test and explore: [react-cdn-firebase-post-app](#) ([code](#))

IN PAIRS

1. Test and explore the above project (Code be fetched from [GitHub](#)).
2. Discuss and explain the functionality of the project.
3. What's the main functionality of the project and how about the data source?
4. Put the project into the perspective of what you have learned so far (databases, backend, JSON & API). Consider why it is important to know about both backend and frontend development.

# WEB DEVELOPMENT

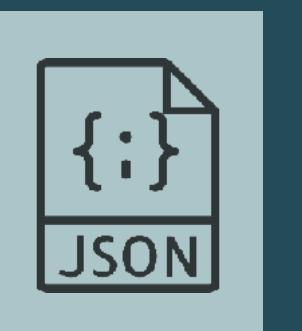
FRONTEND

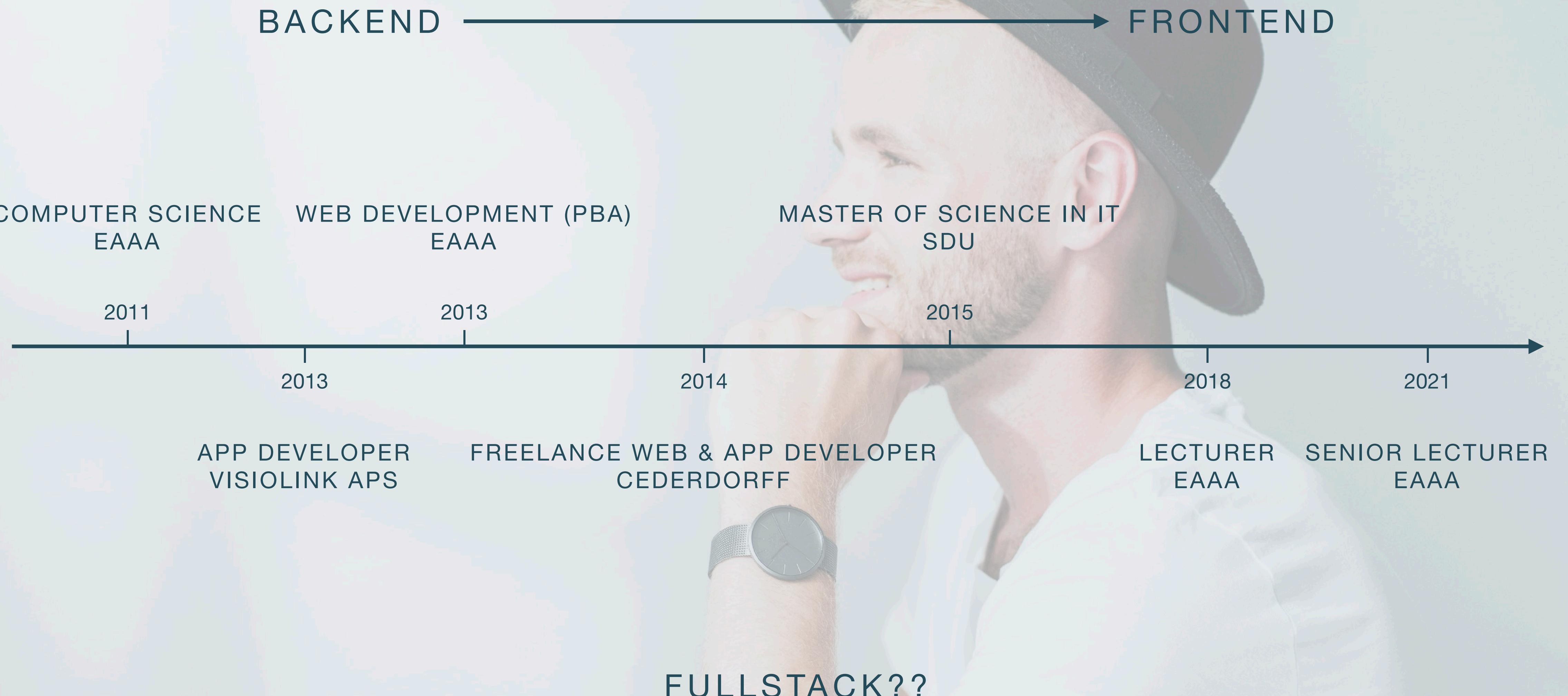


(API)

FETCH(...)  
REQUEST

BACKEND





## BACKEND

DATABASE  
ARCHITECTURE  
PERFORMANCE  
SECURITY  
SCALABILITY

## FRONTEND

UI  
INTERACTIONS  
ARCHITECTURE  
PERFORMANCE  
SECURITY  
SCALABILITY

```
index.html
firebase_crud
> firebase_crud_bootstrap
> firebase_get_started
> form_basic
> form_basic_onchange
> form_location
> form_stepper
> form_stepper_materiali
84 }
85 }
86 /*
87 * Fetches post data from my headless cms
88 */
89 function getPersons() {
90   fetch('http://headlesscms.cederdorff.com/wp-json/wp/v2/posts?_embed')
91     .then(function(response) {
92       return response.json();
93     })
94     .then(function(persons) {
95       persons;
96     });
97   }
98
99   to the DOM
100
101 persons(persons) {
102   let = "";
103   on of persons) {
104     g();
105     let += `<div><img alt="` + person.acf.featuredImage.url + `><h4>` + person.title.rendered + `</h4><p>` + person.acf.age + ` years old</p><p>Hair color: ` + person.acf.hairColor + `</p><p>Relation: ` + person.acf.relation + `</p></div>`;
106   }
107   article>
108   document.querySelector("#family-members").innerHTML = let;
109 }
110
111 }
112 }
113 }
114 }
115 }
116 }
117 }
118 }
119 }
120 }
121 }
122 }
123 }
124 }
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 }
204 }
205 }
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
888 }
889 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
898 }
899 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
978 }
979 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
987 }
988 }
988 }
989 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
```

MacBook Air

I LOVE JS

# PROGRAMMING KNOWLEDGE

## BACKEND & FRONTEND



# FRONTEND PROGRAMMING (5 ECTS)

*[...] deals with the development of simple, modern web applications. The course element contains HTML, CSS, modern web architecture and HTTP protocol. The subject element includes programming and implementation of [...] simple web applications, JavaScript programming language, debugging tools to ensure the quality of the development process and data from web services.”*

*Curriculum, Part 2: Institutional Part*

The Frontend Programming Course will improve  
your knowledge of HTML, CSS, and JavaScript  
(and React), and prepare you for the Advanced  
Web Programming course in the 2nd semester.

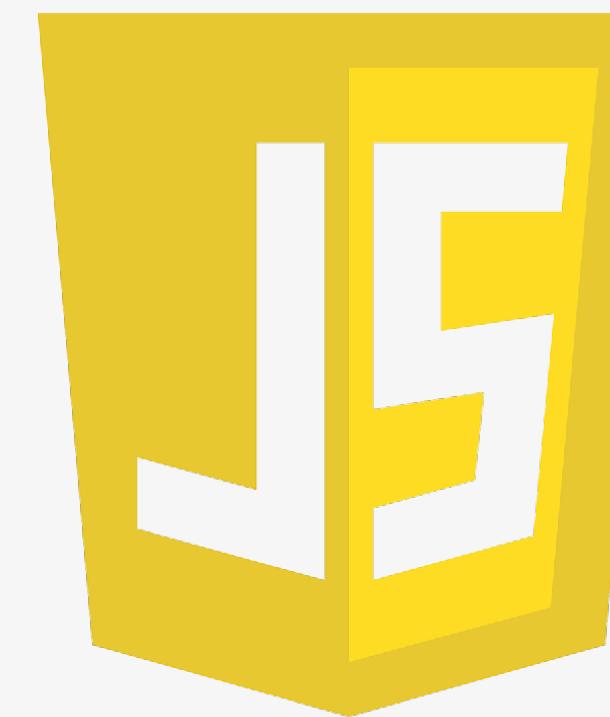


# YOUR EXPERIENCE?

**HTML**



**JS**



**CSS**



HTML, CSS & JAVASCRIPT?

LIRABRIES & FRAMEWORKS?

TOOLS & OTHER FRONTEND STUFF?

# Course Overview

Table

## Lectures

Date	Time	Lecture	Lecturers
21/03/2022	08:30 - 14:00	<a href="#"> 1. HTML, CSS, JavaScript &amp; Web Apps</a>	RACE
28/03/2022	08:30 - 14:00	<a href="#"> 2. From JavaScript to React</a>	RACE
07/04/2022	08:30 - 14:00	<a href="#"> 3. React &amp; Single Page Apps</a>	RACE
22/04/2022	08:30 - 14:00	<a href="#"> 4. React &amp; ASYNC JavaScript</a>	RACE
26/04/2022	08:30 - 14:00	<a href="#"> 5. React CRUD App</a>	RACE
29/04/2022	08:30 - 14:00	<a href="#"> 6. JavaScript meeting Backend</a>	RACE KATO

Course Overview

# EXAM

Individual digitally written 1-hour exam  
(multiple choice)

Online in WISEFlow

<https://race.notion.site/Exam-f03c68da4d5040539923187d714db650>

# INTERDISCIPLINARY PROJECT

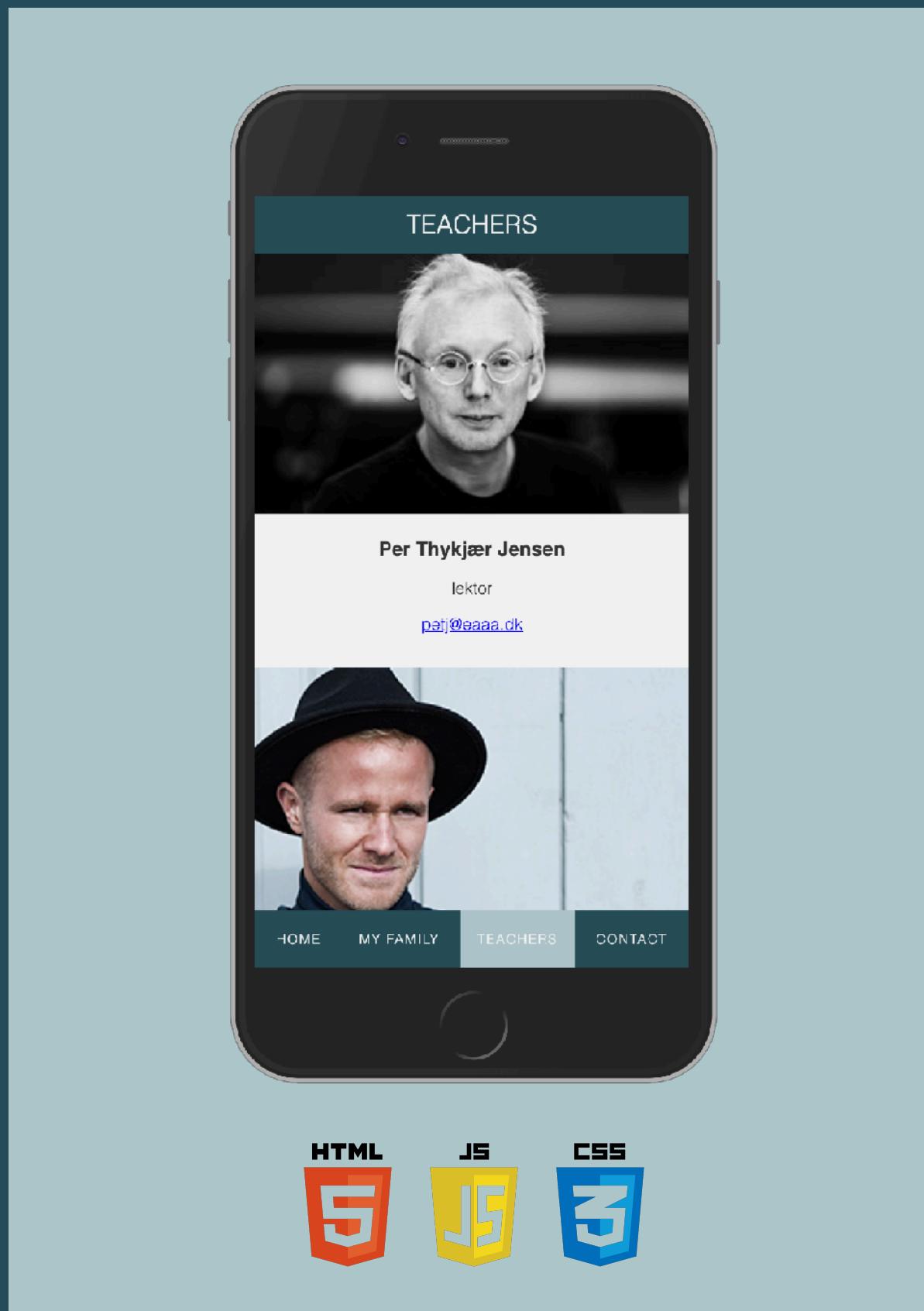
In groups, you have to develop a full-stack web app demonstrating the main parts of your learned skills and competencies from the 1st semester (Interaction and experience design, Databases, Backend & Frontend).

The project is finalized by a group presentation of your web app.  
The project will be used in addition to your exams in January.

**Format:** Full-stack Web App developed in groups of 2-3 students and a group presentation.  
**Project Kick-Off:** April 29  
**Hand in:** December 25  
**Group presentations:** May 25

# INTERDISCIPLINARY PROJECT

FRONTEND



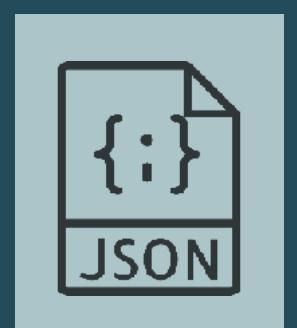
(API)

FETCH(...)  
REQUEST

BACKEND



JSON DATA  
RESPONSE



# TOOLS



VISUAL STUDIO CODE  
EXTENSIONS: ESLINT & LIVE SERVER



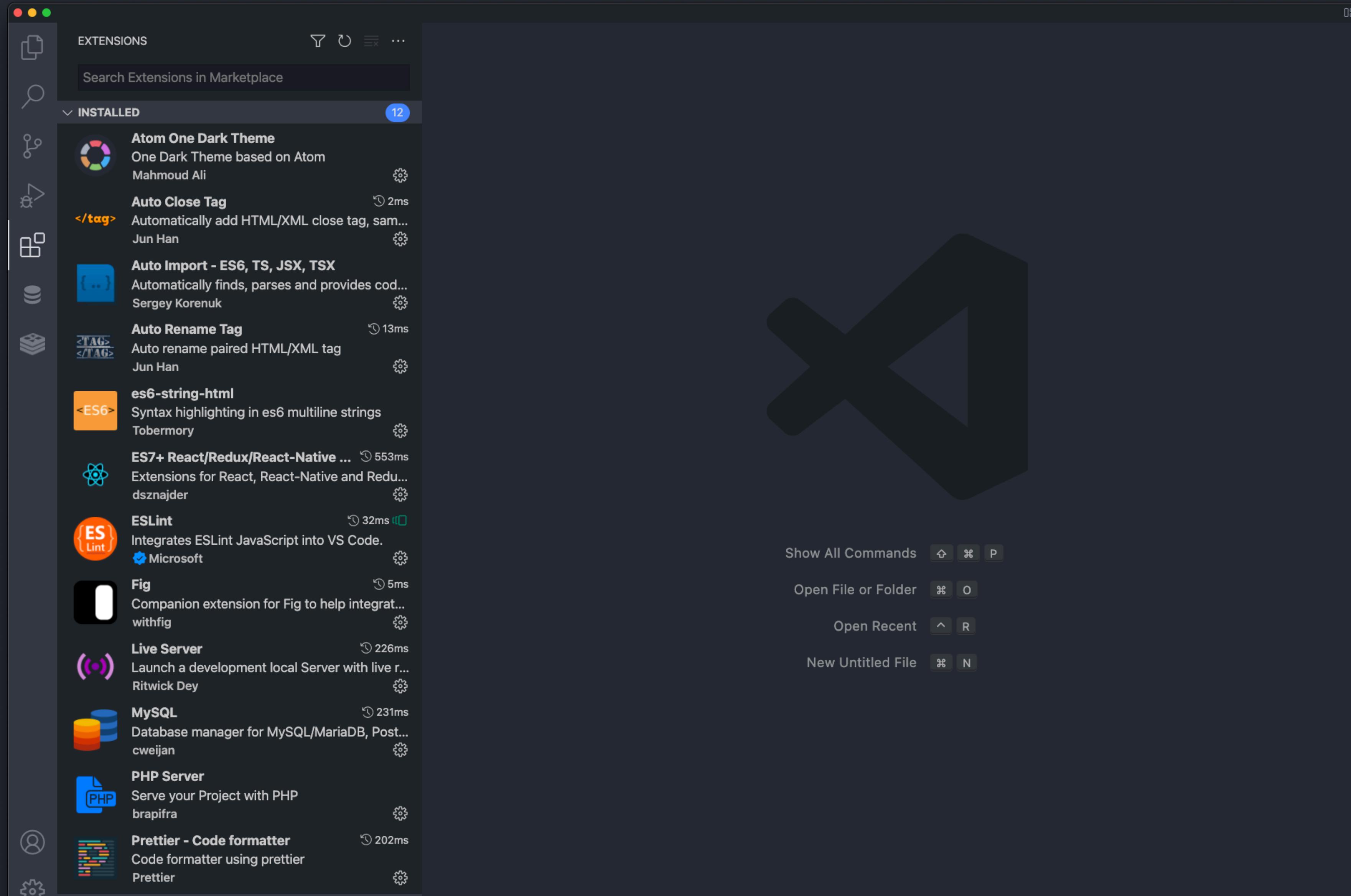
GITHUB DESKTOP



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 are filtered under the 'User' tab. The 'Text Editor (4)' section includes 'Formatting (3)' and 'Prettier - Code formatter'. The 'Extensions (17)' section includes 'ESLint (1)', 'HTML (7)', 'JSON (1)', 'Prettier (5)', and 'TypeScript (3)'. A large callout box highlights the 'Prettier - Code formatter' entry under 'Text Editor'. The 'Editor: Default Formatter' setting is described as defining a default formatter that takes precedence over others. The 'Editor: Format On Paste' setting has a checked checkbox, indicating it controls whether the editor automatically formats pasted content. The 'Editor: Format On Save' setting also has a checked checkbox, indicating it formats files on save. The 'Editor: Format On Type' setting has an unchecked checkbox, indicating it controls automatic line-by-line formatting.

Settings — mdu-frontend

Extension: Prettier - Code formatter

Search Extensions in ...

INSTALLED 6

User Workspace

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

formatt 21 Settings Found Turn on Settings Sync

**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.

# FRONTEND STACK

**HTML**



**JS**



**CSS**



STRUCTURE  
CONTENT

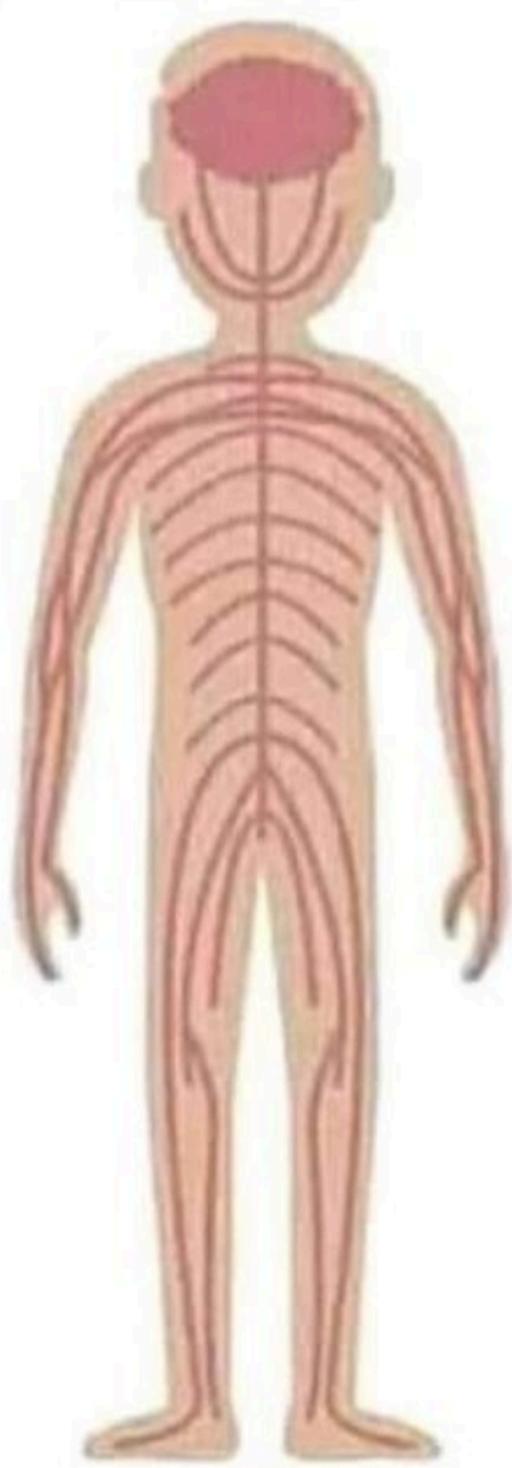
FUNCTIONALITY  
BEHAVIOR

LAYOUT/STYLING  
PRESENTATION

HTML



JS



CSS



# JAVASCRIPT, HTML & CSS

YOU CAN LITERALLY BUILD ANYTHING WITH IT

# SEPARATION OF CONCERNS

**HTML**



**JS**



**CSS**



STRUCTURE  
CONTENT

FUNCTIONALITY  
BEHAVIOR

LAYOUT/STYLING  
PRESENTATION

# FRONTEND PROJECT STRUCTURE

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

```
index.html — css-grid
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
```

```
index.html — project-template
1  <!doctype html>
2  <html lang="en">
3
4  <head>
5    <meta charset="utf-8">
6    <title>PROJECT TEMPLATE</title>
7    <meta name="viewport" content="width=device-width, initial-scale=1.0">
8    <meta name="description" content="Get started with your new project template!">
9    <meta name="author" content="Rasmus Cederdorff">
10   <link rel="stylesheet" href="css/main.css">
11   <link rel="shortcut icon" type="image/png" href="img/favicon.png" />
12 </head>
13
14 <body>
15   <header>
16     <h1>PROJECT TEMPLATE</h1>
17   </header>
18   <button onclick="showAlert()">Hit me</button>
19   <section id="content"></section>
20   <!-- main js file -->
21   <script src="js/main.js"></script>
```

# 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](https://www.w3schools.com/html/html_intro.asp)

```
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>
```

< > ⌂ https://9nypo.csb.app/

# This is a Heading

This is a paragraph.

```
<> 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>  
11
```

The `<!DOCTYPE html>` declaration defines that this document is an HTML5 document

The `<html>` element is the root element of an HTML page

The `<head>` element contains meta information about the HTML page

The `<title>` element specifies a title for the HTML page (which is shown in the browser's title bar or in the page's tab)

The `<body>` element defines the document's body, and is a container for all the visible contents, such as headings, paragraphs, images, hyperlinks, tables, lists, etc.

The `<h1>` element defines a large heading

The `<p>` element defines a paragraph

```
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>  
11
```

```
<html>  
  
<head>  
  
  <title>Page title</title>  
  
</head>  
  
<body>  
  
  <h1>This is a heading</h1>  
  
  <p>This is a paragraph.</p>  
  
  <p>This is another paragraph.</p>  
  
</body>  
  
</html>
```

# 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>
```

## NON SEMANTIC

```
<div class='header'>  
  
<div class='section'>  
    <div class='article'>  
        <div class='article'>  
  
<div class='footer'>
```

## SEMANTIC

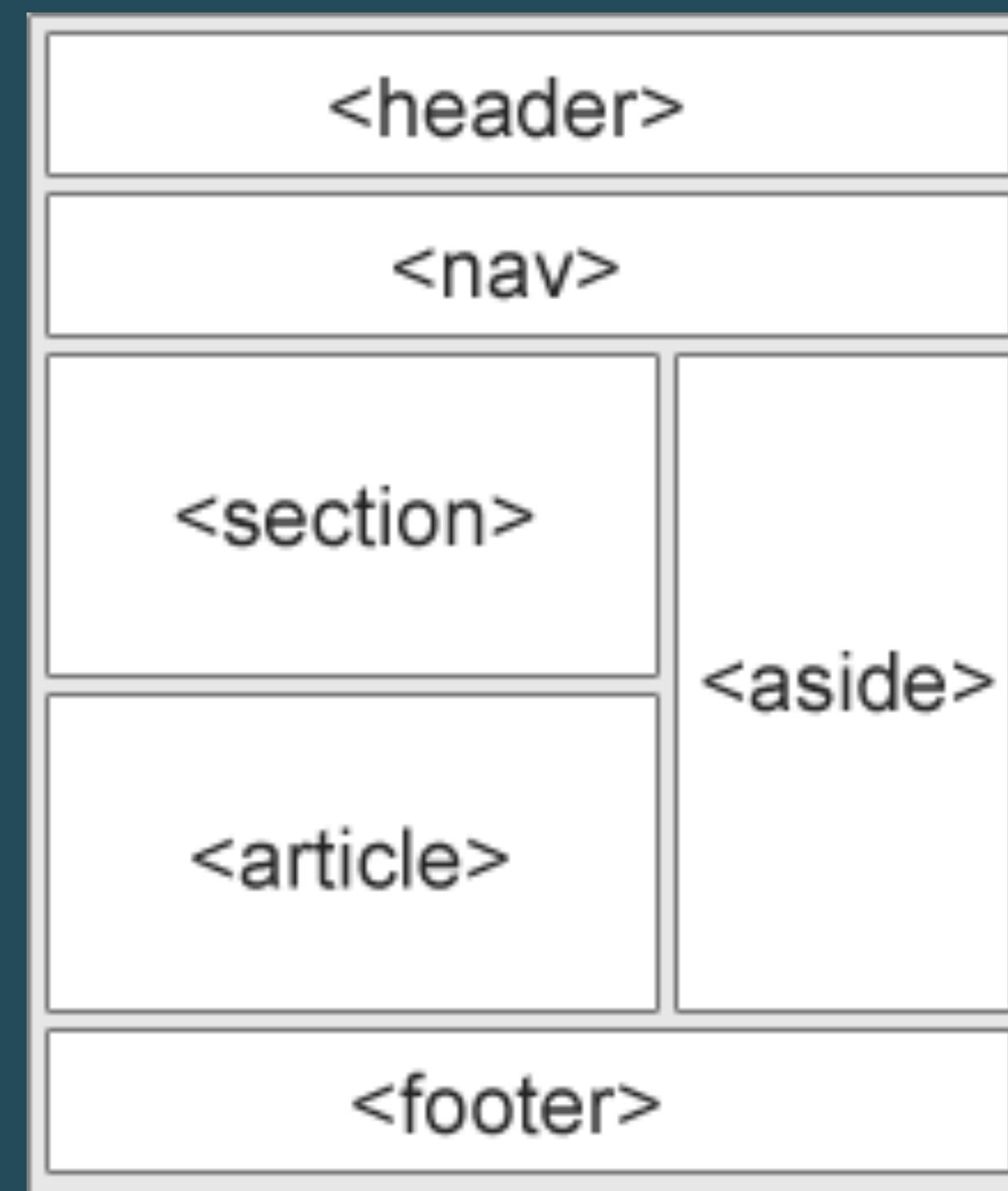
```
<header>  
  
<section>  
    <article>  
    <article>  
  
<aside>  
  
<footer>
```



# 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 `<span>` - 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 is visible, showing the structure of the HTML page. The "Styles" tab is also visible at the bottom, showing CSS rules applied to the elements.

# 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](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 Introduction

w3schools.com/css/css\_intro.asp

Incognito

HTML CSS JAVASCRIPT SQL PYTHON PHP BOOTSTRAP MORE REFERENCES CERTIFICATES

COLOR PICKER

SHOP

HOW TO

Tabs  
Dropdowns  
Accordions  
Side Navigation  
Top Navigation  
Modal Boxes  
Progress Bars  
Parallax  
Login Form  
HTML Includes  
Google Maps  
Range Sliders  
Tooltips  
Slideshow  
Filter List  
Sort List

## CSS Tutorial

- CSS HOME
- CSS Introduction**
- CSS Syntax
- CSS Selectors
- CSS How To
- CSS Comments
- CSS Colors
- CSS Backgrounds**
- CSS Borders
- CSS Margins
- CSS Padding
- CSS Height/Width
- CSS Box Model
- CSS Outline
- CSS Text
- CSS Fonts
- CSS Icons
- CSS Links
- CSS Lists
- CSS Tables
- CSS Display
- CSS Max-width
- CSS Position
- CSS Overflow
- CSS Float

# Welcome to My Homepage

Use the menu to select different Stylesheets

**Stylesheet 1**

Stylesheet 2

Stylesheet 3

Stylesheet 4

No Stylesheet

## Same Page Different Stylesheets

This is a demonstration of how different stylesheets can change the layout of your HTML page. You can change the layout of this page by selecting different stylesheets in the menu, or by selecting one of the following links:

[Stylesheet1](#), [Stylesheet2](#), [Stylesheet3](#), [Stylesheet4](#).

## No Styles

This page uses DIV elements to group different sections of the HTML page. Click here to see how the page looks like with no stylesheet:

[No Stylesheet](#).

**Side-Bar**

Lore ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

<> index.html x

Browser Tests

< > ⌂ https://9nypo.csb.app/

```
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     </body>
11 </html>
12
```

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
```

This is a Heading  
This is a paragraph.

# 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)

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     <p id="my-paragraph">This is another paragraph.</p>
11     <p class="big">This is another paragraph.</p>
12     
13   </body>
14 </html>
```

styles.css x

```
9
10 #my-paragraph {
11   color: blue;
12 }
13
14 .big {
15   font-size: 25px;
16 }
17
18 img {
19   width: 100%;
20 }
```

...

Browser

Tests



https://9nypo.csb.app/



# This is a Heading

This is a paragraph.

This is another paragraph.

This is another paragraph.



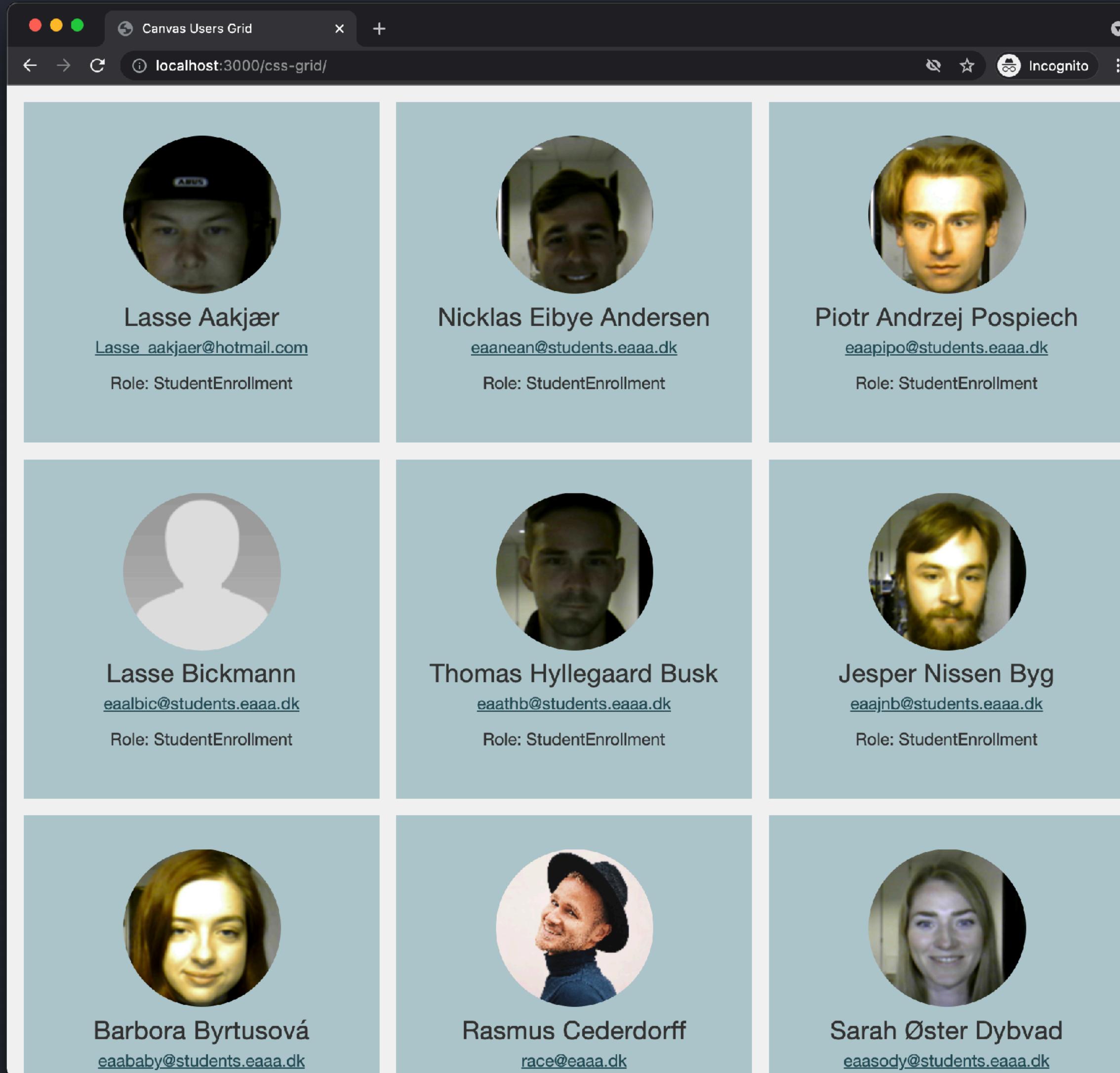
# CSS LAYOUTS

## 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>
```

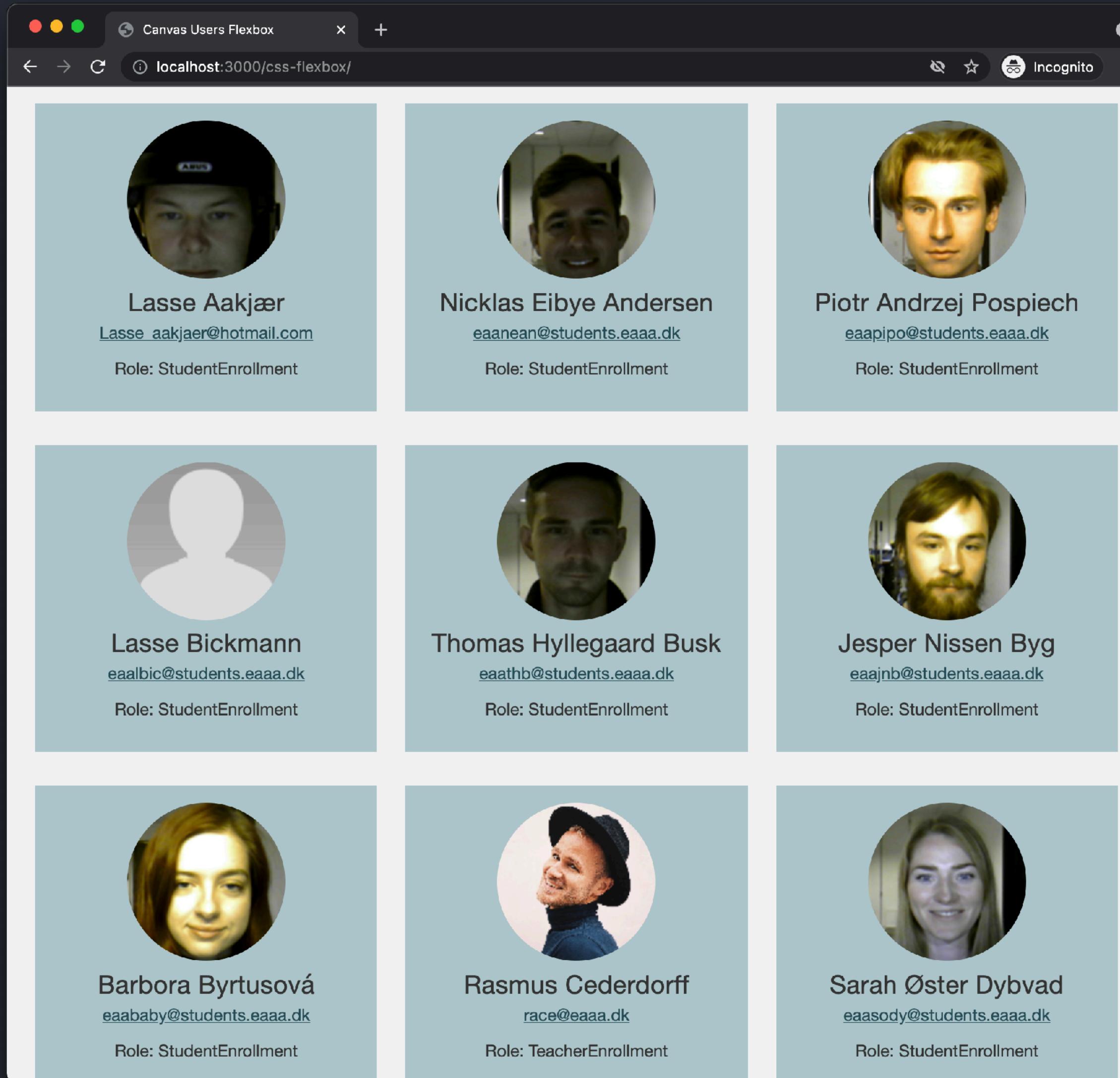
```
/* ----- grid styling ----- */
```

```
.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 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 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



```
.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:

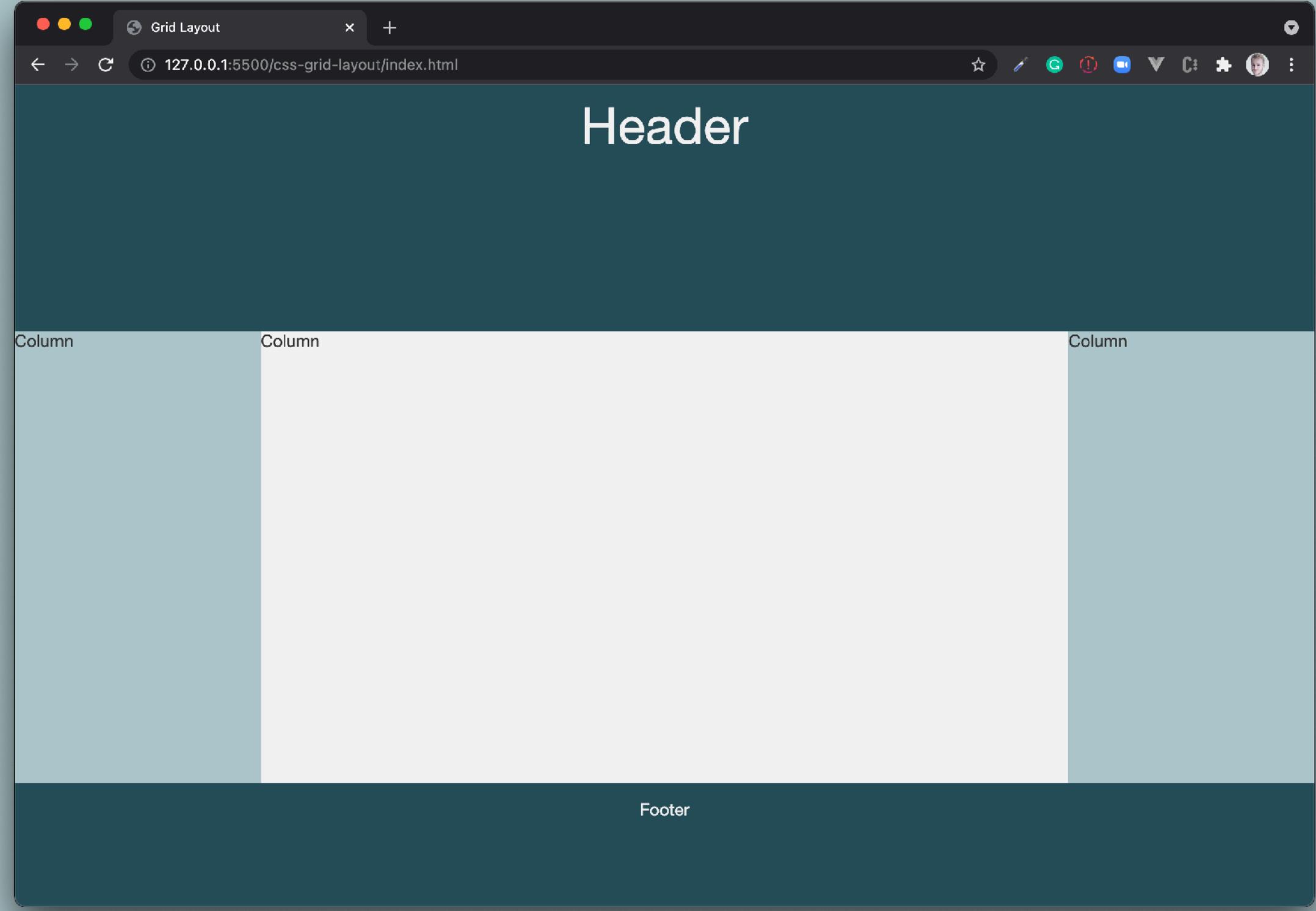
Sidenav and content:

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

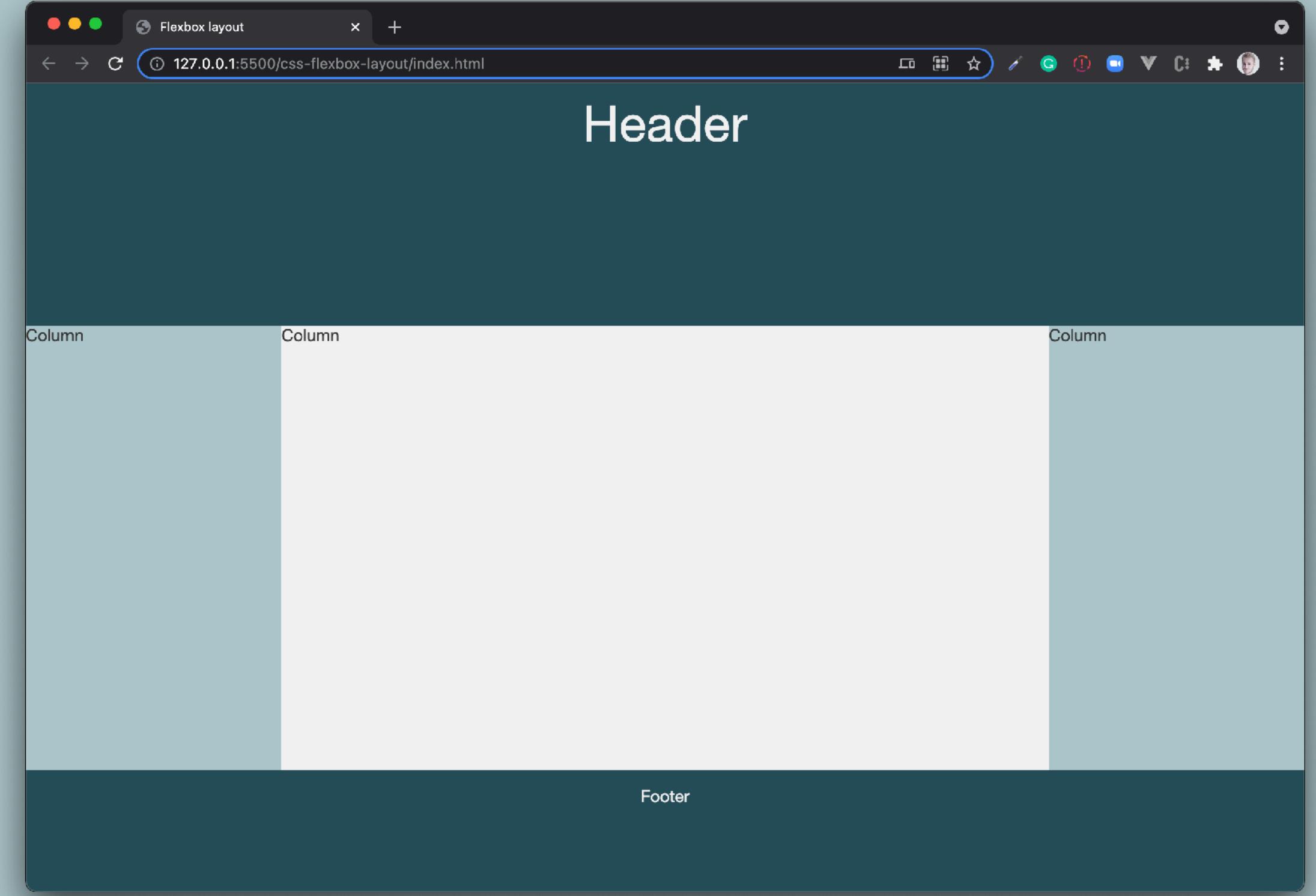
NEW  
We just launched W3Schools videos  
Explore now

COLOR PICKER

LIKE US



css-grid-layout



css-flexbox-layout

Course roster: WU-E21s - 1. se

aaaa.instructure.com/courses/13351/users

Incognito

WU-E21s > People

60 Student view

Home Announcements Modules People BigBlueButton (Formerly Conferences) Assignments Discussions 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	To A
Lasse Aakjær	eaalaak@students.eaaa.dk	WU-E21s - 1.	Student	Student	25 Aug at 22:27	02
Nicklas Eibye Andersen	eaanean@students.eaaa.dk	WU-E21s - 1.	Student	Student	25 Aug at 10:00	02
Piotr Andrzej Pospiech	eaapipo@students.eaaa.dk	WU-E21s - 1.	Student	Student	26 Aug at 12:54	08
Lasse Bickmann	eaalbic@students.eaaa.dk	WU-E21s - 1.	Student	Student		
Thomas Hyllegaard Busk	eaathb@students.eaaa.dk	WU-E21s - 1.	Student	Student	24 Aug at 17:29	
Jesper Nissen Byg	eaajnb@students.eaaa.dk	WU-E21s - 1.	Student	Student	26 Aug at 10:49	09
Barbora Byrtusová	eaababy@students.eaaa.dk	WU-E21s - 1.	Student	Student	25 Aug at 14:27	
Rasmus Cederdorff	race@eaaa.dk	WU-E21s - 1.	Teacher	Teacher	26 Aug at 17:03	02

Elements Console Sources Network Performance

Filter Hide data URLs

All Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other Has blocked cookies

Blocked Requests

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

Name Headers Preview Response Initiator Timing Cookies

unread\_count

group\_categories...

users?include\_ina...

collect?v=1&\_v=j9...

unread\_count

courses?include[]...

accounts

[{id: "17636", name: "Lasse Aakjær", created\_at: "2019-08-03T02:26:25+02:00"}]

0: {id: "17636", name: "Lasse Aakjær", created\_at: "2019-08-03T02:26:25+02:00"}  
1: {id: "17929", name: "Nicklas Eibye Andersen", created\_at: "2019-08-07T00:00:00+02:00"}  
2: {id: "17476", name: "Piotr Andrzej Pospiech", created\_at: "2019-08-02T00:00:00+02:00"}  
3: {id: "30257", name: "Lasse Bickmann", created\_at: "2021-08-05T00:58:06+02:00"}  
4: {id: "30252", name: "Thomas Hyllegaard Busk", created\_at: "2021-08-05T00:58:06+02:00"}  
5: {id: "11879", name: "Jesper Nissen Byg", created\_at: "2018-08-08T02:10:00+02:00"}  
6: {id: "17960", name: "Barbora Byrtusová", created\_at: "2019-08-07T02:34:00+02:00"}  
7: {id: "14427", name: "Rasmus Cederdorff", created\_at: "2018-09-06T02:23:57+02:00"}  
avatar\_url: "https://eaaa.instructure.com/images-thumbnails/649049/1EeRFV...  
created\_at: "2018-09-06T02:23:57+02:00"  
custom\_links: []  
email: "race@eaaa.dk"  
enrollments: [{course\_id: "13351", id: "320419", user\_id: "14427", type: "Student", id: "14427", integration\_id: null, login\_id: "race@eaaa.dk", name: "Rasmus Cederdorff", short\_name: "Rasmus Cederdorff (adjunkt – race@eaaa.dk)", sis\_user\_id: null, sortable\_name: "Cederdorff, Rasmus"}]  
8: {id: "41", name: "Jeffrey David Serio", created\_at: "2017-05-10T14:09:27+02:00"}  
9: {id: "30251", name: "Sarah Øster Dybvad", created\_at: "2021-08-05T00:57:00+02:00"}  
10: {id: "17477", name: "Kristine Dzumakaijeva", created\_at: "2019-08-02T02:00:00+02:00"}  
11: {id: "17478", name: "Ksenia Dzumakaijeva", created\_at: "2019-08-02T02:00:00+02:00"}  
12: {id: "17479", name: "Dmitry Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
13: {id: "17480", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
14: {id: "17481", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
15: {id: "17482", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
16: {id: "17483", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
17: {id: "17484", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
18: {id: "17485", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
19: {id: "17486", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
20: {id: "17487", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
21: {id: "17488", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
22: {id: "17489", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
23: {id: "17490", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
24: {id: "17491", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
25: {id: "17492", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
26: {id: "17493", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
27: {id: "17494", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
28: {id: "17495", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
29: {id: "17496", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
30: {id: "17497", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
31: {id: "17498", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
32: {id: "17499", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
33: {id: "17500", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
34: {id: "17501", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
35: {id: "17502", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
36: {id: "17503", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
37: {id: "17504", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
38: {id: "17505", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
39: {id: "17506", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
40: {id: "17507", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
41: {id: "17508", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
42: {id: "17509", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
43: {id: "17510", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
44: {id: "17511", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
45: {id: "17512", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
46: {id: "17513", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
47: {id: "17514", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
48: {id: "17515", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
49: {id: "17516", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
50: {id: "17517", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
51: {id: "17518", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
52: {id: "17519", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
53: {id: "17520", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
54: {id: "17521", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
55: {id: "17522", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
56: {id: "17523", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
57: {id: "17524", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
58: {id: "17525", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
59: {id: "17526", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
60: {id: "17527", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
61: {id: "17528", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
62: {id: "17529", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
63: {id: "17530", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
64: {id: "17531", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
65: {id: "17532", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
66: {id: "17533", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
67: {id: "17534", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
68: {id: "17535", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
69: {id: "17536", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
70: {id: "17537", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
71: {id: "17538", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
72: {id: "17539", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
73: {id: "17540", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
74: {id: "17541", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
75: {id: "17542", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
76: {id: "17543", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
77: {id: "17544", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
78: {id: "17545", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
79: {id: "17546", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
80: {id: "17547", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
81: {id: "17548", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
82: {id: "17549", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
83: {id: "17550", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
84: {id: "17551", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
85: {id: "17552", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
86: {id: "17553", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
87: {id: "17554", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
88: {id: "17555", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
89: {id: "17556", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
90: {id: "17557", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
91: {id: "17558", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
92: {id: "17559", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
93: {id: "17560", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
94: {id: "17561", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
95: {id: "17562", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
96: {id: "17563", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
97: {id: "17564", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
98: {id: "17565", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
99: {id: "17566", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
100: {id: "17567", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
101: {id: "17568", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
102: {id: "17569", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
103: {id: "17570", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
104: {id: "17571", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
105: {id: "17572", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
106: {id: "17573", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
107: {id: "17574", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:00:00+02:00"}  
108: {id

# GRID LAYOUT

BACK

1. Create a new project folder in your own local dev folder.
2. Inside the project folder, create a HTML file (`index.html`) and define the basic structure of an HTML page with `html`, `head` & `body`. Get a basic [HTML5 Template here](#).
3. Add a `section` tag as your grid or flex box container.
4. Add some `article` tags with content inside of your section. The content could be user data (name, email, image, etc. Inspiration: <https://cederdorff.github.io/web-frontend/canvas-users/data.json>). Static content - no JavaScript!
5. Add and link to an external style sheet named `main.css`
6. Make use of a CSS grid or flexbox to create a layout to display users. Inspiration: [CSS Grid Layout Module](#) & [CSS Flexbox](#)

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

# PAGE LAYOUT

BACK

1. Create a new project folder in your own local dev folder.
2. Inside the project folder, create a HTML file (`index.html`) and define the basic structure of an HTML page with `html`, `head` & `body`. Get a basic [HTML5 Template here](#).
3. Add the following tags with content: `header`, `nav`, `main`, `section`, `article`, `aside`, `footer`, some headings & paragraphs.
4. Add and link to an external style sheet named `main.css`
5. Make use of a CSS grid or flexbox to create a full page layout with navbar, header, main and footer. Inspiration: [CSS Grid Layout Module](#) & [CSS Layout Templates](#)

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

CSS LAYOUT.IO

LAYOUTS AND PATTERNS MADE WITH CSS

# CSS TEMPLATES

[https://www.w3schools.com/css/css\\_templates.asp](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](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);
}

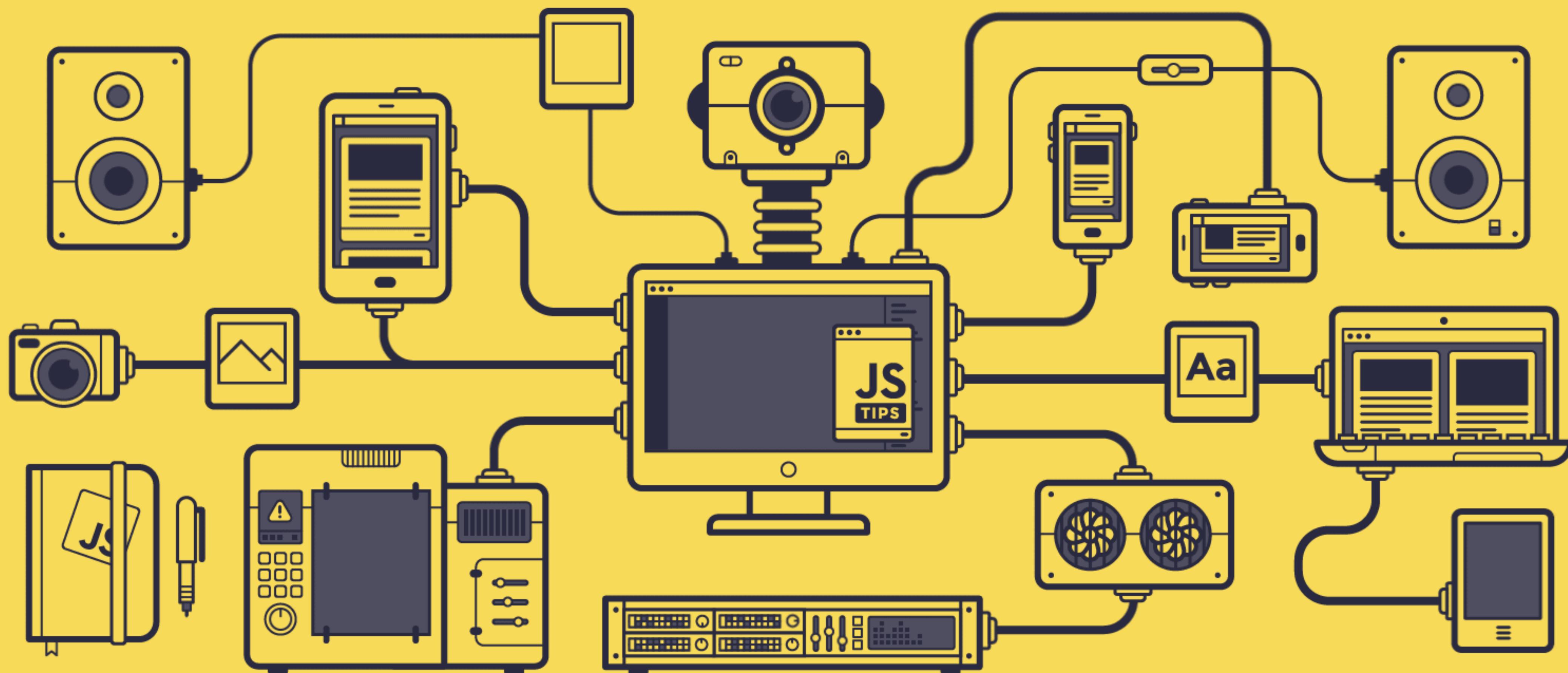
a {
    color: var(--green);
}

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

# GRID & CARD LAYOUT

BACK

1. Create a grid using a grid container and grid items. You can use the layout from the previous exercise and place the new grid and card layout inside of your main.
2. Make sure the grid adjusts to the size of the screen (adjust the number of columns with media queries).
3. Add static content to your grid items. The content must be based on Canvas user objects with values from the properties: <https://cederdorff.github.io/web-frontend/canvas-users/data.json>. Add at least 6 items (users).
4. Display the following for each user: name, image, mail & enrollment\_type
5. Style each item (user) as a card. Inspiration: <https://csslayout.io/card/>, [Cards](#), [Profile Card](#) & [Flip Card](#)
6. Add root variables at the top of your CSS script.
7. Make sure to use the variables through out your CSS script instead of keep typing the same color, font, sizes, etc.



The screenshot shows a development environment with three main panes:

- Code Editor (Top Left):** Displays the `index.html` file with the following content:

```
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>
```
- Browser Preview (Top Right):** Shows the rendered HTML. The heading "This is a Heading" is displayed in large bold black font. Below it is a paragraph "This is a paragraph." To the right of the paragraph is a button labeled "Try me". The URL in the browser bar is <https://brg12.csb.app/>.
- Terminal (Bottom Left):** Displays the `app.js` file with the following content:

```
1 function tryMe() {
2   document.body.style.backgroundColor = "red";
3   document.body.style.color = "white";
4   document.querySelector("h1").innerHTML = "My new header";
5   alert("Hi, Welcome to my web app!");
6 }
7
```

At the bottom center of the image is the URL <https://codesandbox.io/s/javascript-starter-brg12>.

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.

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

... can change styling of HTML.

app.js x

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

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

index.html x

...

Browser

Tests



https://brg12.csb.app/



```
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     <input type="text" placeholder="Type your name" id="name" />
11     <button onclick="tryMe()">Try me</button>
12
13     <script src="app.js"></script>
14   </body>
15 </html>
16
```

# This is a Heading

This is a paragraph.

Peter

Try me

app.js x

...

```
1  function tryMe() {
2    let name = document.getElementById("name").value;
3    alert("Hi " + name + ", Welcome to my web app!");
4  }
5
```

<https://codesandbox.io/s/javascript-input-okm7e>

index.html x

...

Browser

Tests

< > ⌂ https://brg12.csb.app/

```
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         <input type="text" placeholder="Type your name" id="name" />
11         <button onclick="tryMe()">Try me</button>
12         <section id="content"></section>
13         <script src="app.js"></script>
14     </body>
15 </html>
16
```

# This is a Heading

This is a paragraph.

Type your name  Try me

app.js x

□ □ ...

```
1  function tryMe() {
2      let name = document.getElementById("name").value;
3      document.getElementById("content").innerHTML += name;
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);

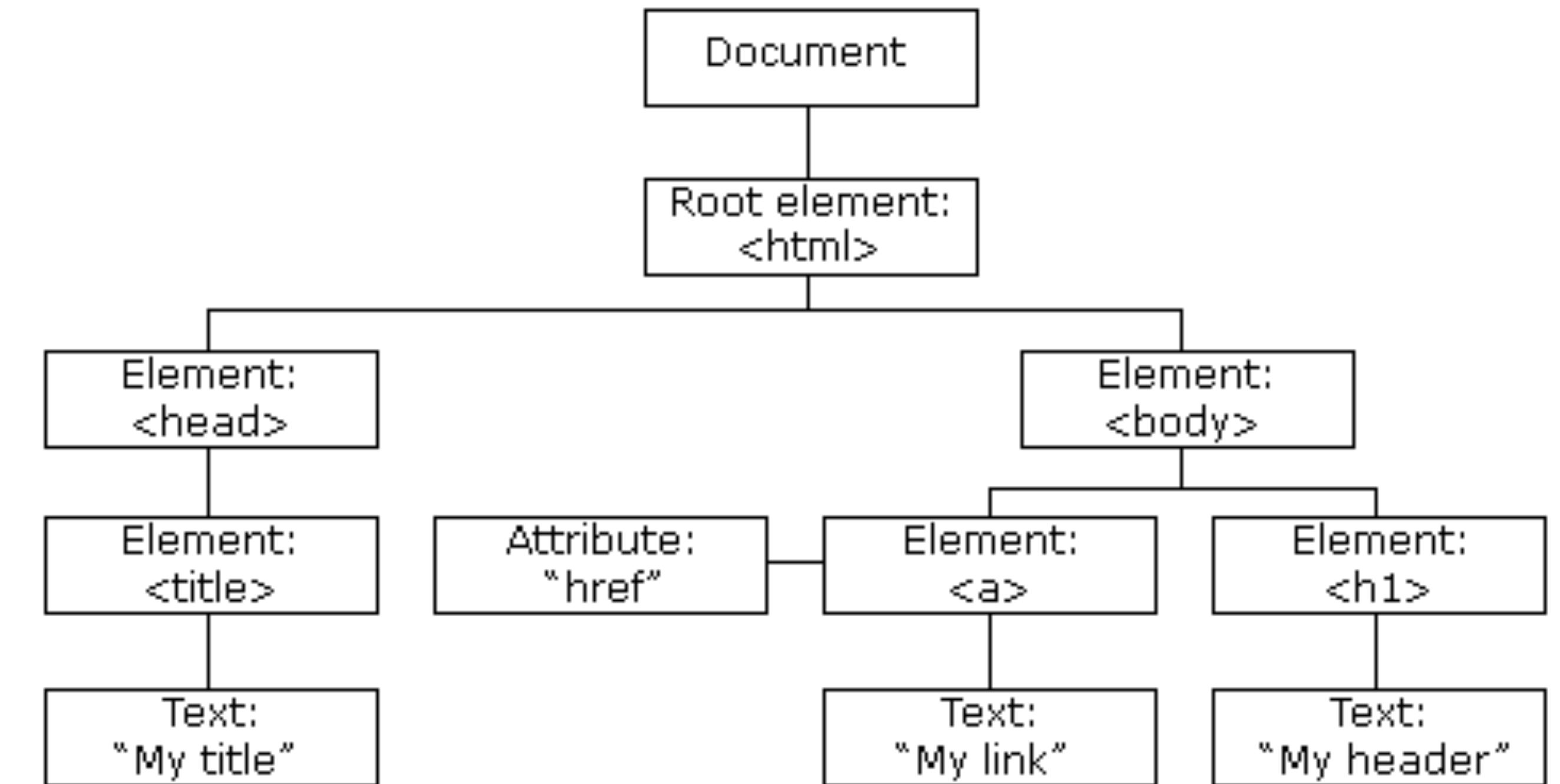
// appending the variable (the string) to the DOM element #content
document.querySelector("#content").innerHTML = message;
```

```
<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://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://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: 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\*?

# JS HTML DOM #1

BACK

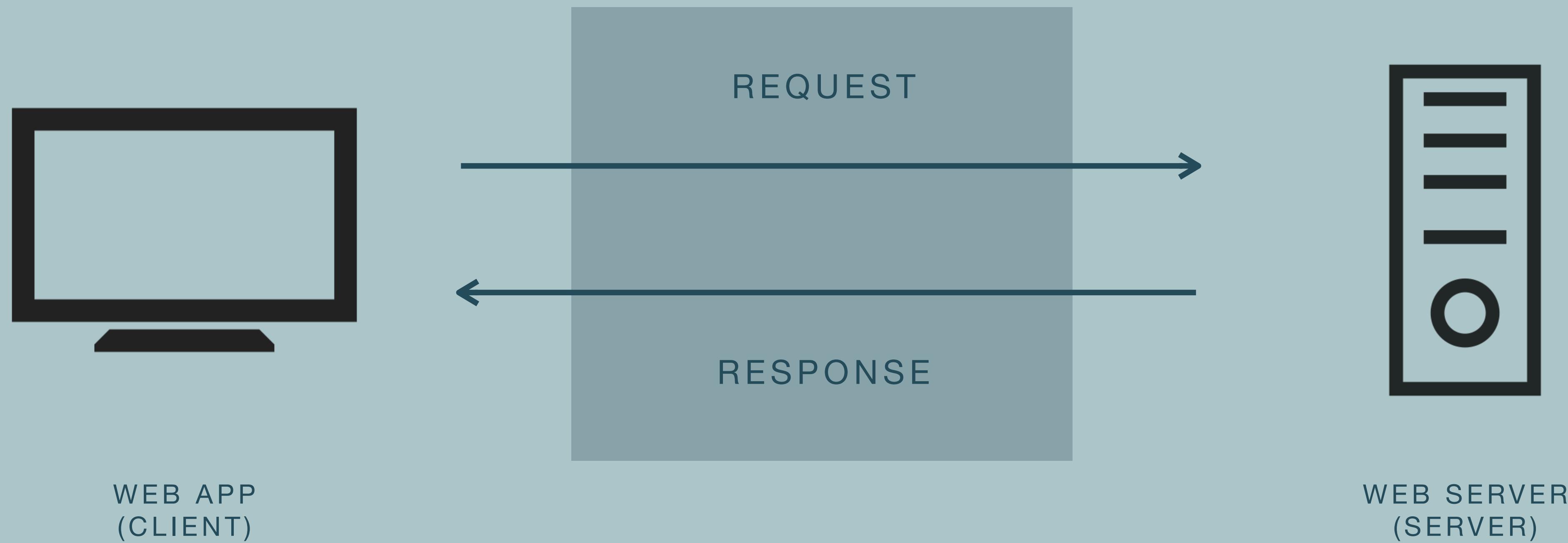
1. Use your previous codebase.
2. Create and link to an external JavaScript script called `app.js`
3. Consider where in your HTML to place the link to your external JavaScript script.
4. Make sure the script is connected by creating a `console.log(...)` at the top of the script, logging “`app.js` is ready <img alt="party hat emoji" data-bbox="358 508 378 538”/>”. Test in browser and developer tool.
5. Create a function called `changeBackgroundColor()` changing the background color of the body. Use `document.querySelector(...)` to refer to body in the DOM.
6. Create a button in your HTML file and add a `onclick` event calling `changeBackgroundColor()`. You can use the `onclick` attribute on the button tag or add an `eventlistener` in your JS. Test the button and functionality.
7. Extra: Customise `changeBackgroundColor()` or create a new function called `turnOnDarkMode()`. The function must change the styling of your page to dark mode.

# JS HTML DOM #2

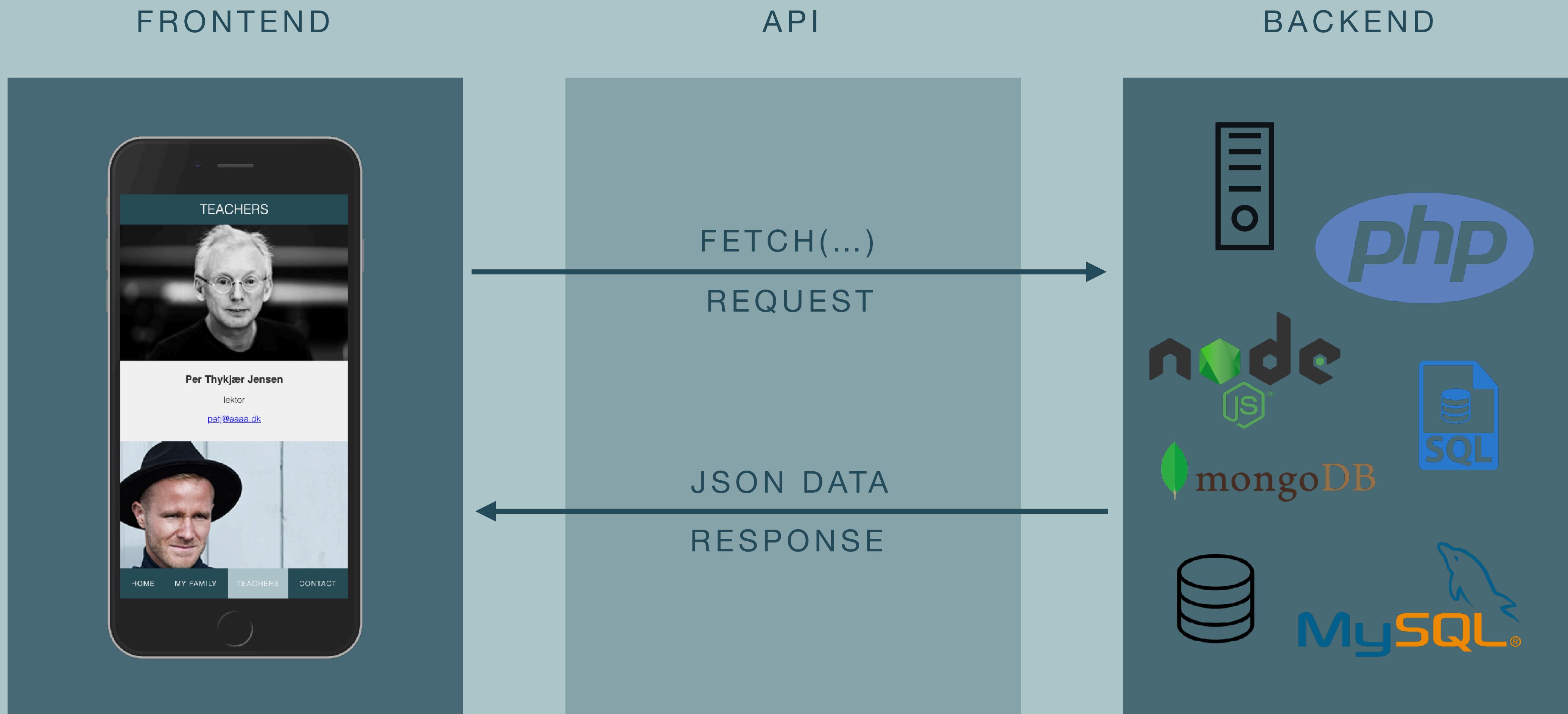
BACK

1. Define an input field and a new button in your HTML file.
2. Declare a new function called `alertName()` in your JS file.
3. Add an `onclick` event to the button executing `alertName()`.
4. `alertName()` should grab the value of the input field and `alert(...)` the value.
5. Test in browser.

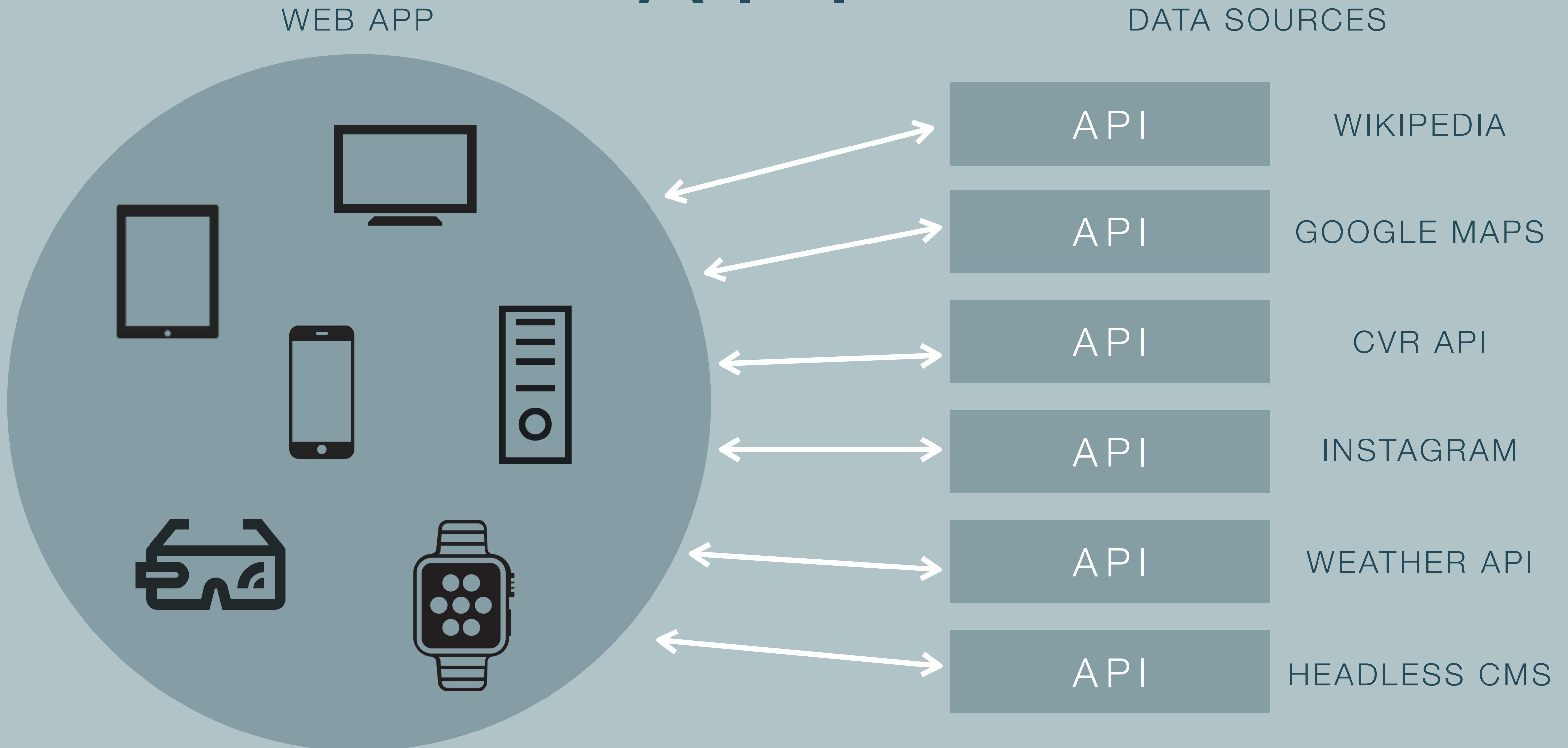
# CLIENT SERVER MODEL



# FETCH DATA



# API



# JAVASCRIPT CONCEPTS

## FRONTEND DEVELOPMENT

**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

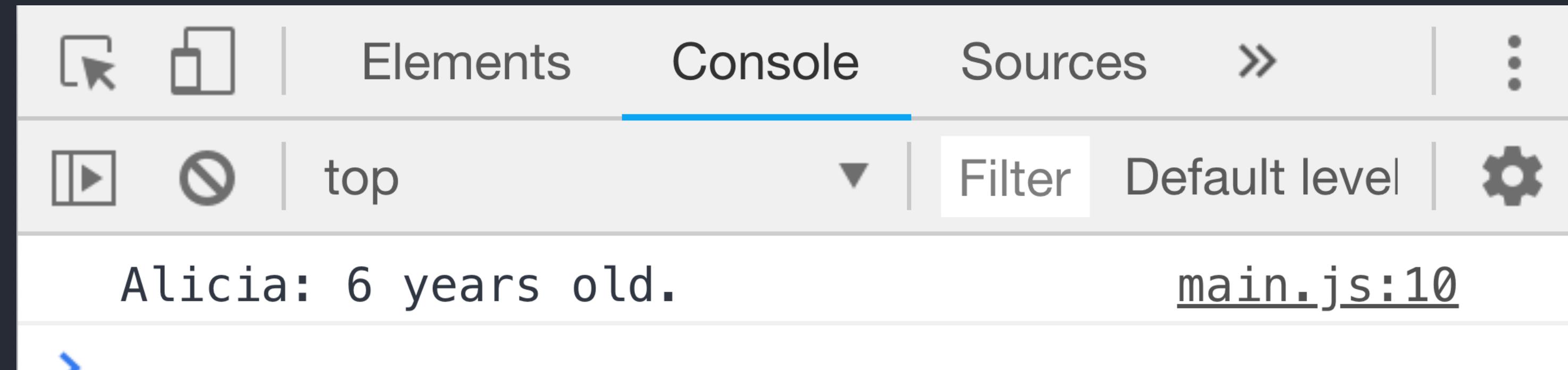
# VARIABLES

... STORE DATA IN THE MEMORY

```
let name = "Alicia";
```

```
let age = 6;
```

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

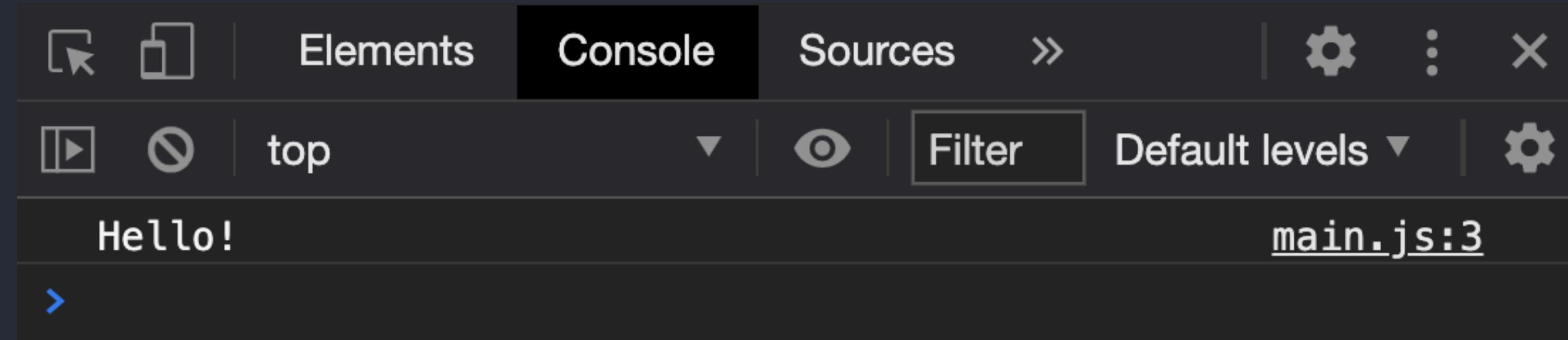


# VARIABLES

... STORE DATA IN THE MEMORY

```
let message = "Hello!";
```

```
console.log(message);
```

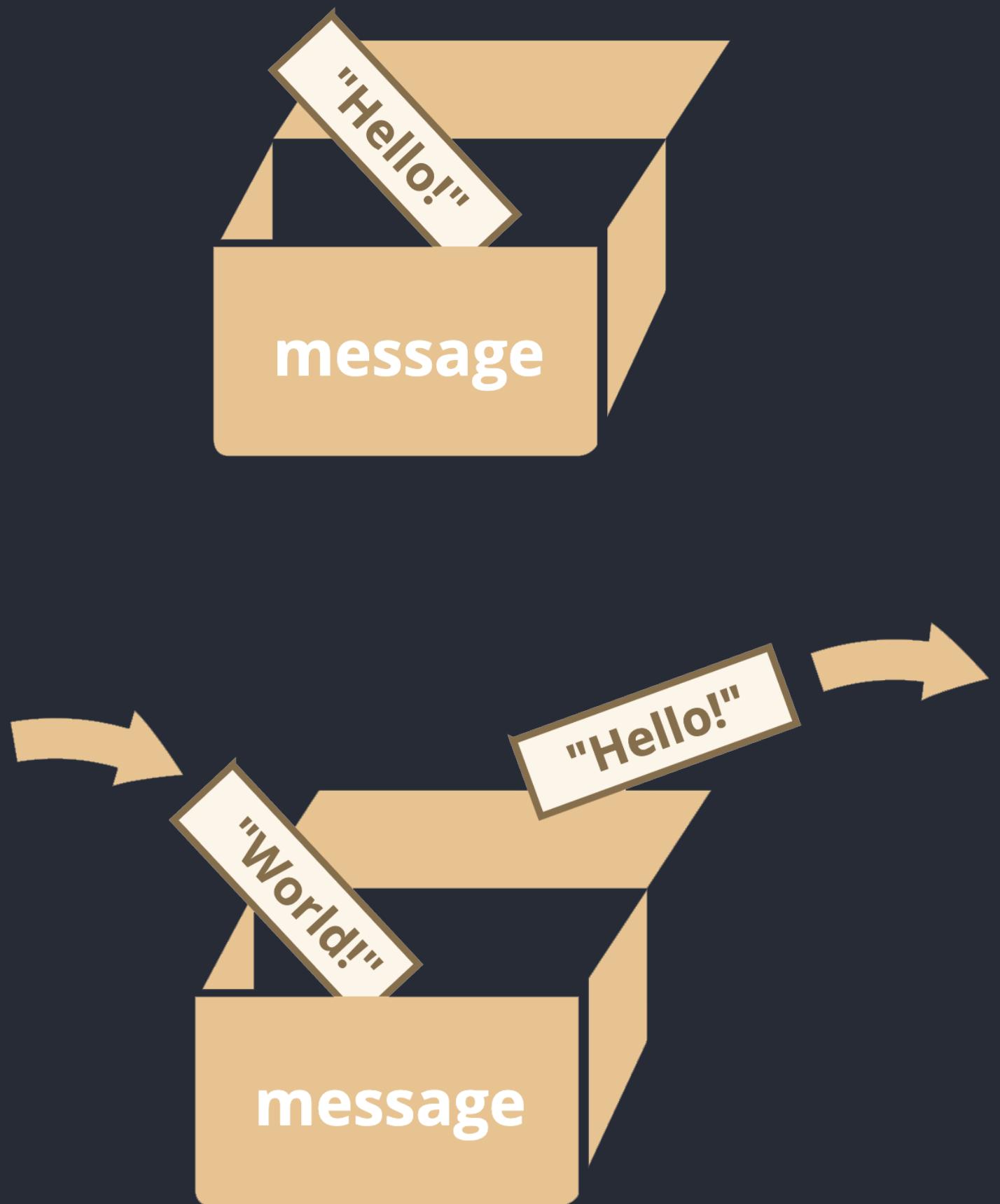


# VARIABLES

... STORE DATA IN THE MEMORY

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

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

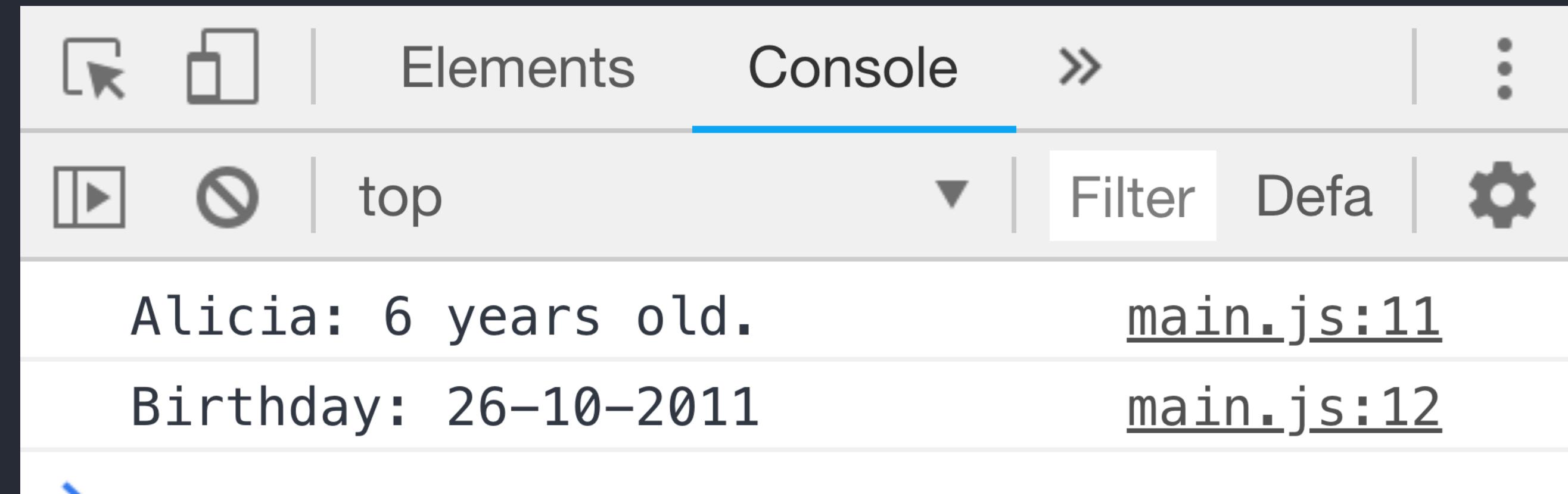


# VARIABLES

... STORE DATA IN THE MEMORY

```
let name = "Alicia";
let age = 6;
const birthday = '26-10-2011';

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

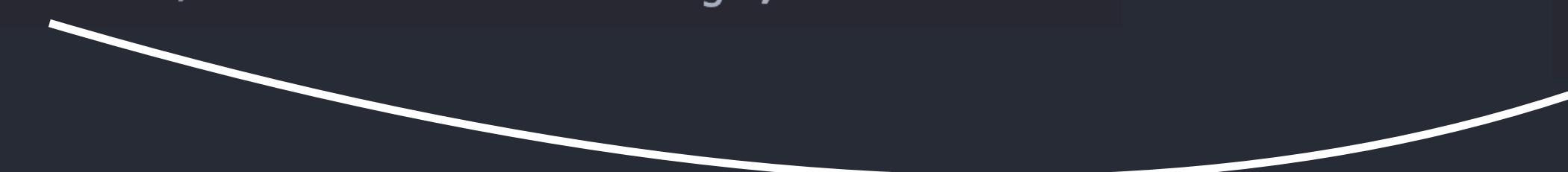


```
// declaring a variable with a value
let message = "Hi Frontenders!"

//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>
```

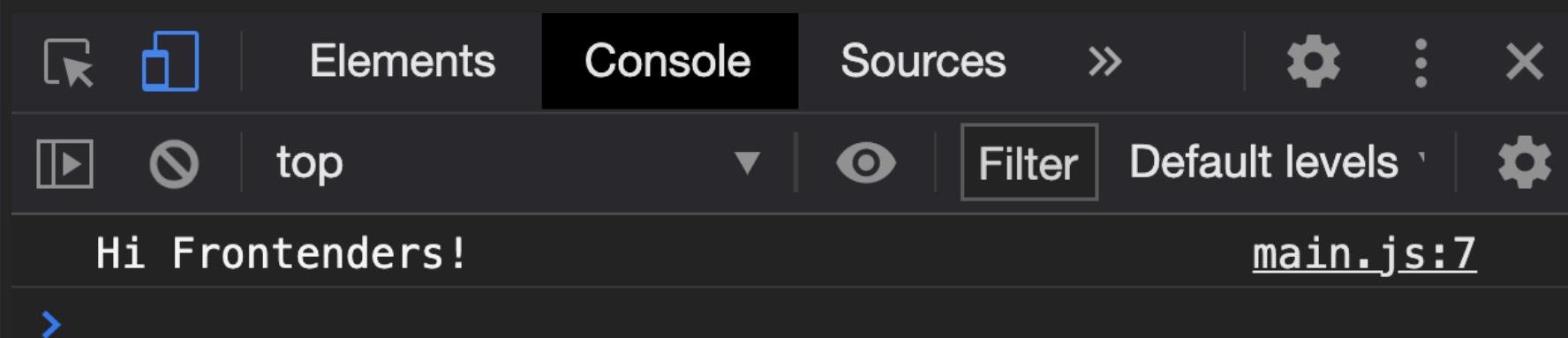


```
// declaring a variable with a value
let message = "Hi Frontenders!"

// 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:

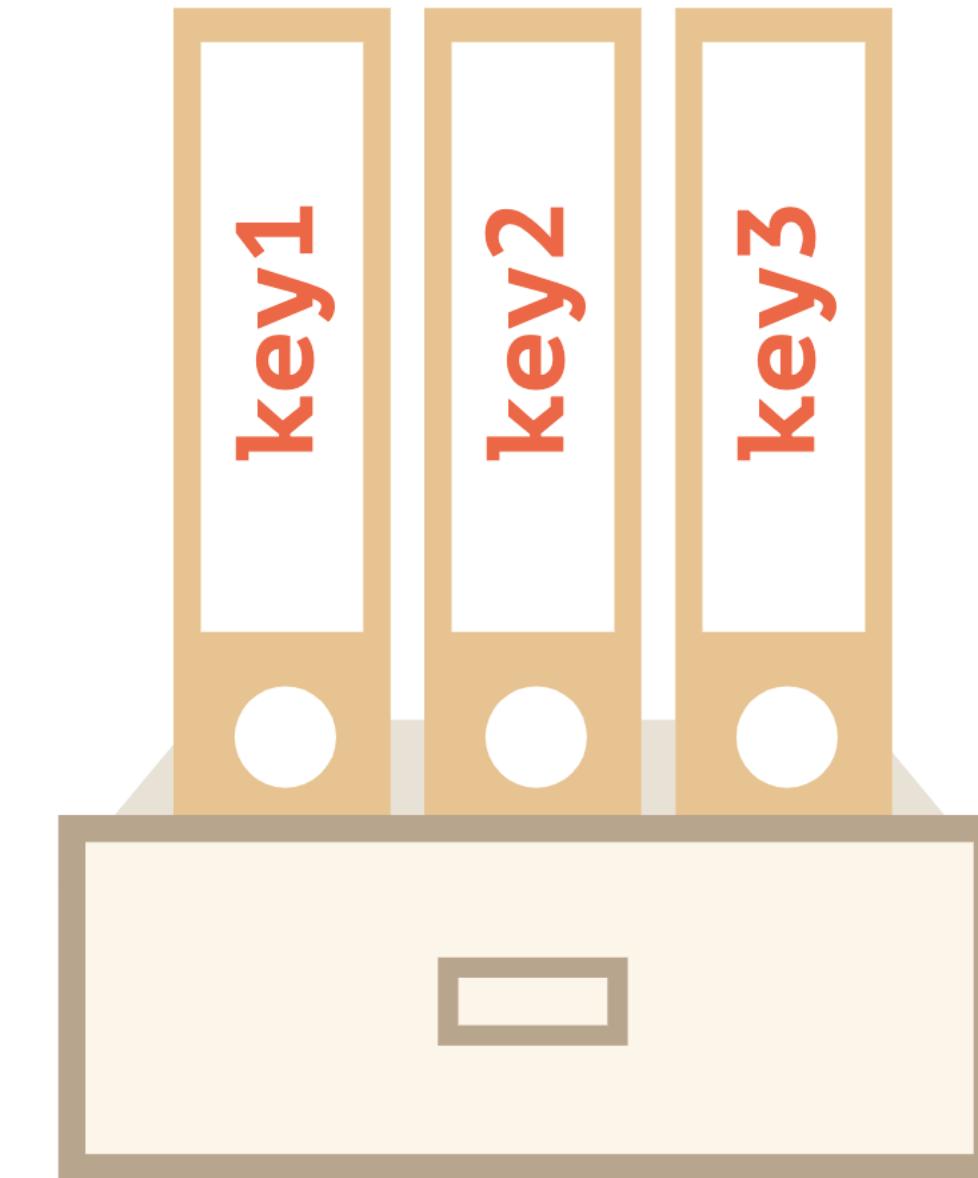
- Use human-readable names like `userName` or `shoppingCart`.
- Stay away from abbreviations or short names like `a`, `b`, `c`, unless you really know what you're doing.
- Make names maximally descriptive and concise. Examples of bad names are `data` and `value`. Such names say nothing. It's only okay 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`.

# O B J E C T S

... A SET OF NAMED VALUES

# O B J E C T S

Objects are used to store keyed collections of various data



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

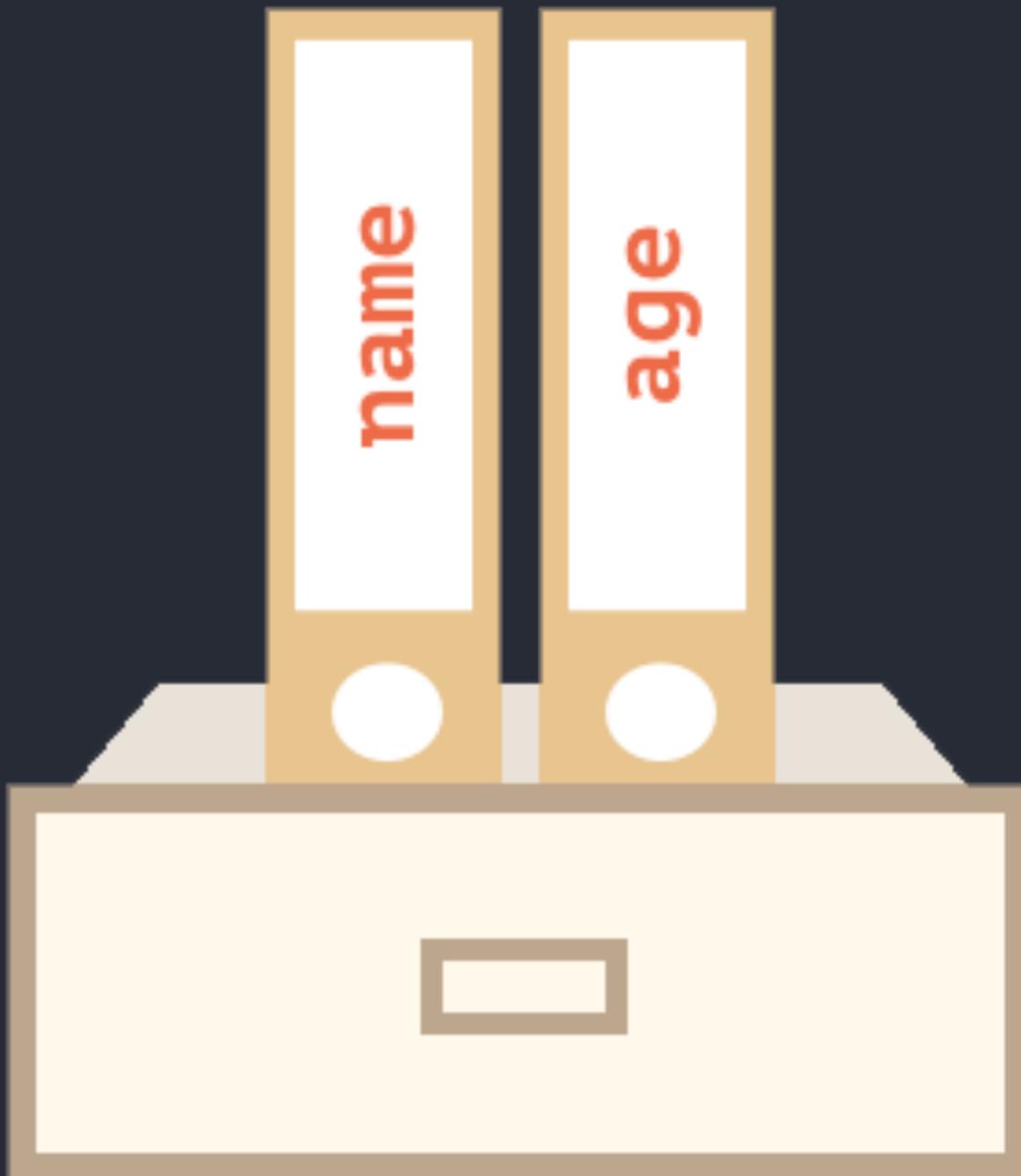
# OBJECTS

A SET OF NAMED VALUES

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

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

user



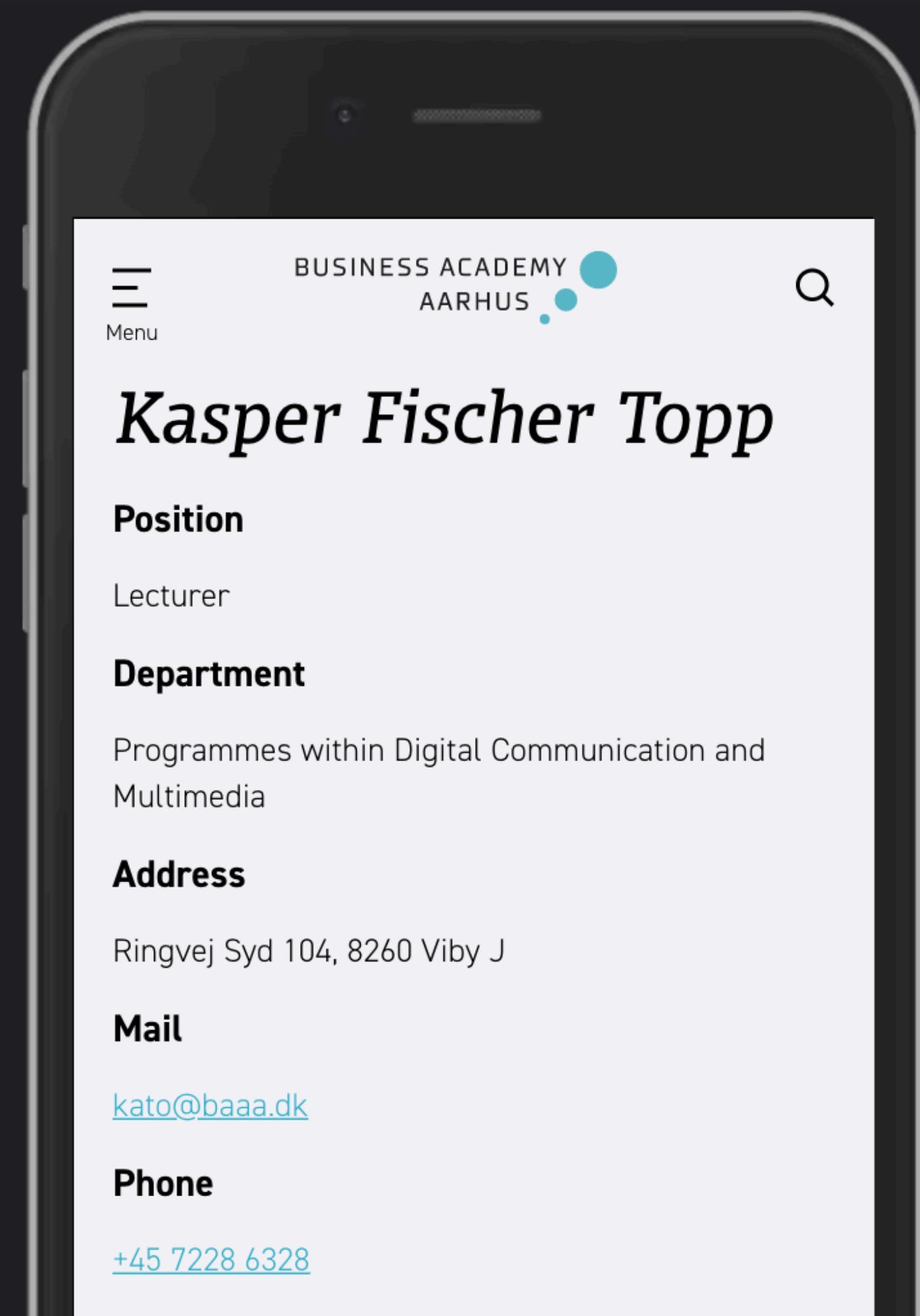
Alicia is 6 years old.

main.js:11

# OBJECTS

# A SET OF NAMED VALUES

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



● ○ ● netflix.com ↕

NETFLIX

# Hvem ser?



Personen der  
rent faktisk  
betaler for  
profilen

Nasser 1      Nasser 2      Nasser 3      Nasser 4 Khader

Administrer profiler

● ○ ● | □ | < > ⌂ +

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

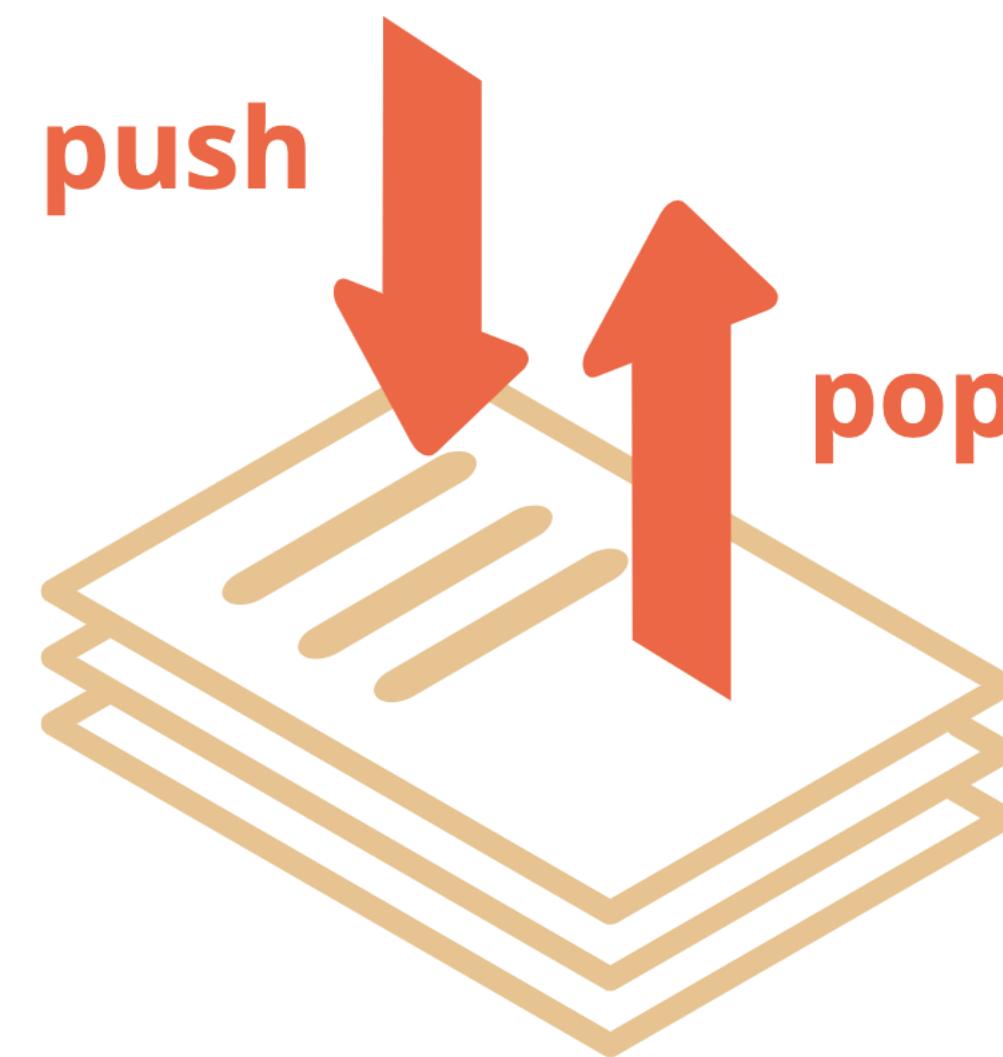
```
let movie = {  
  title: "Frozen 2",  
  description: "Elsa the Snow Queen has a",  
  trailer: "https://www.youtube.com/embed",  
  length: "1h 43m",  
  year: "2019"  
}
```

# ARRAYS

... A COLLECTION OF VALUES OR OBJECTS

# ARRAYS

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

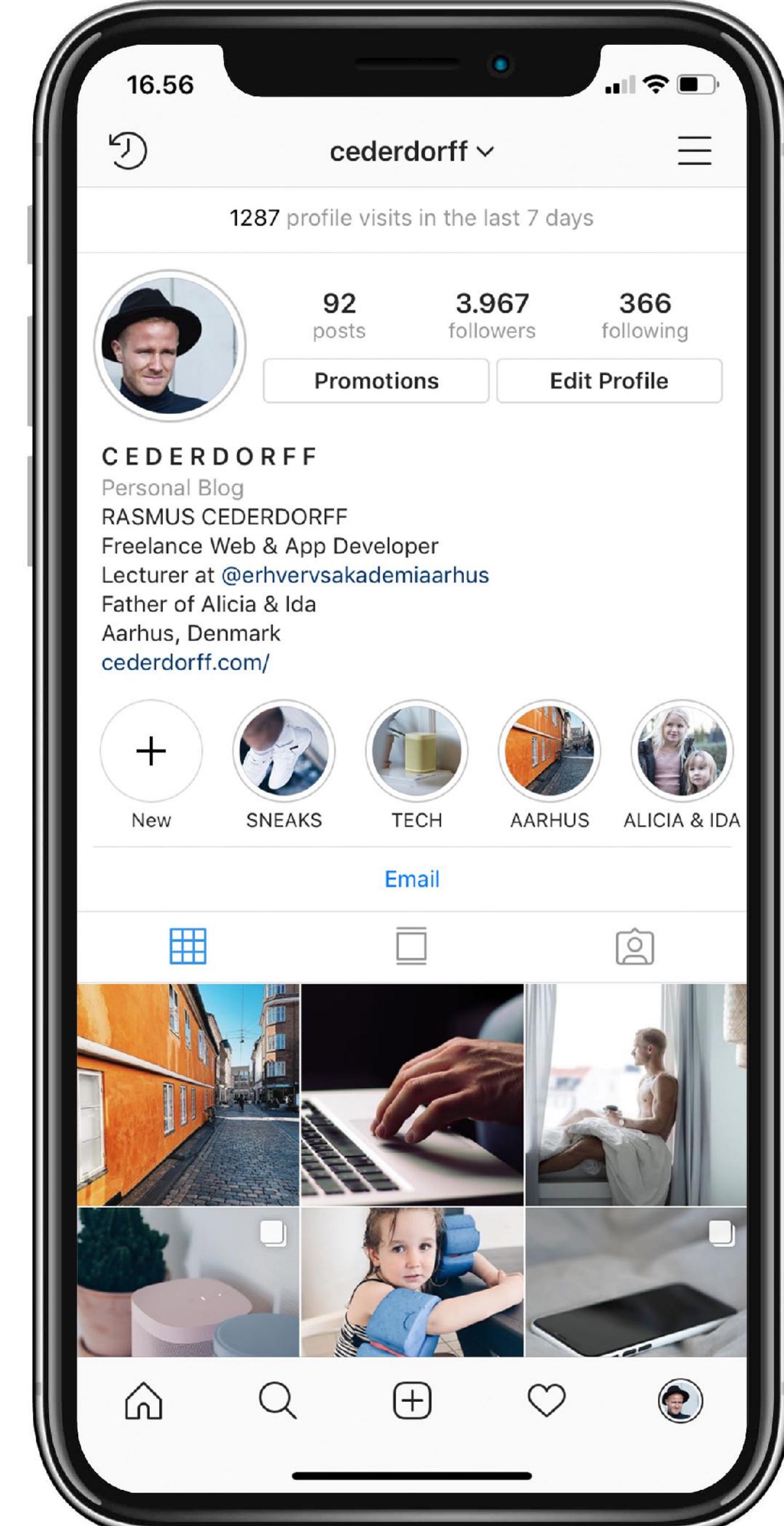
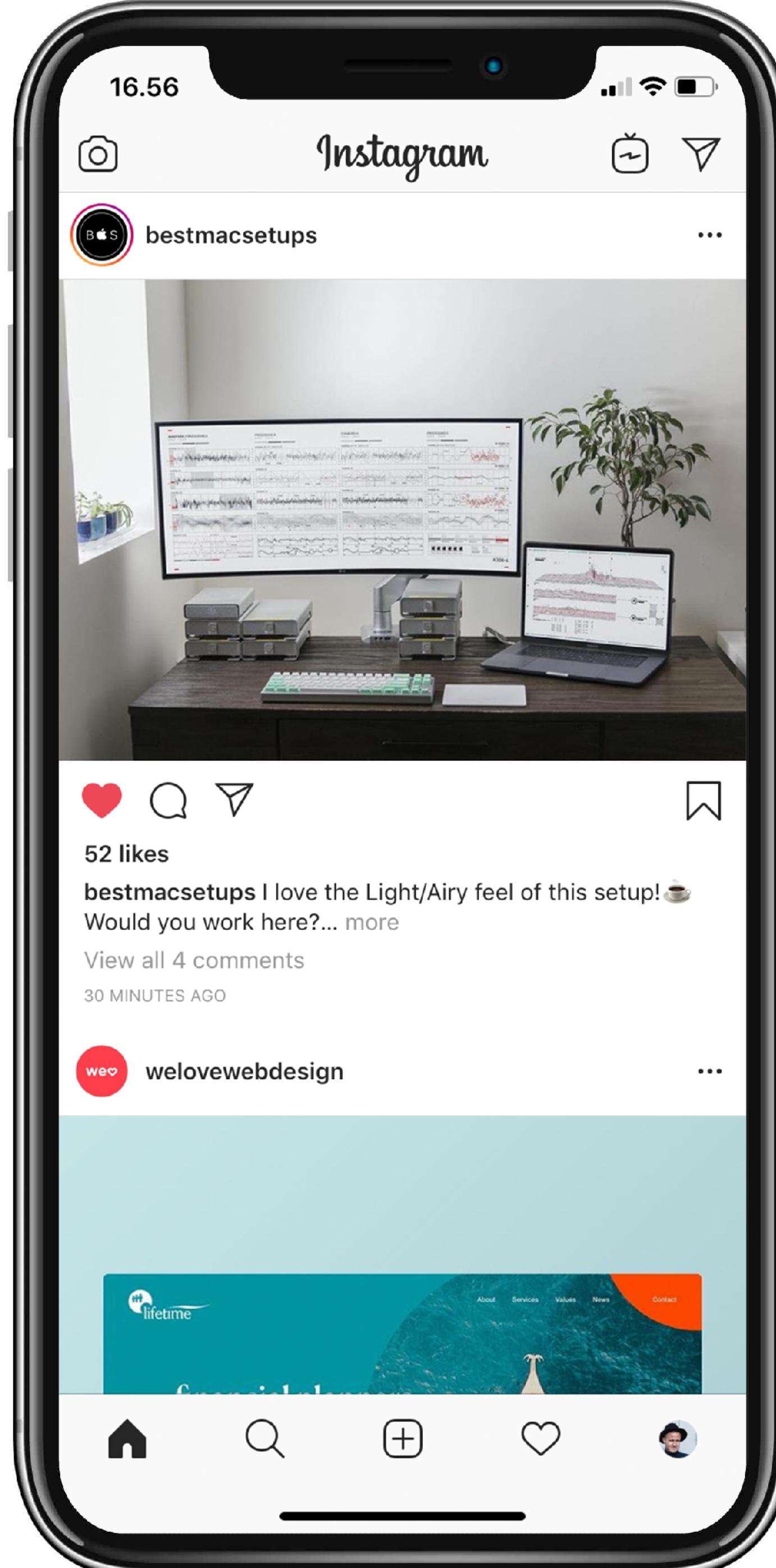
Second element

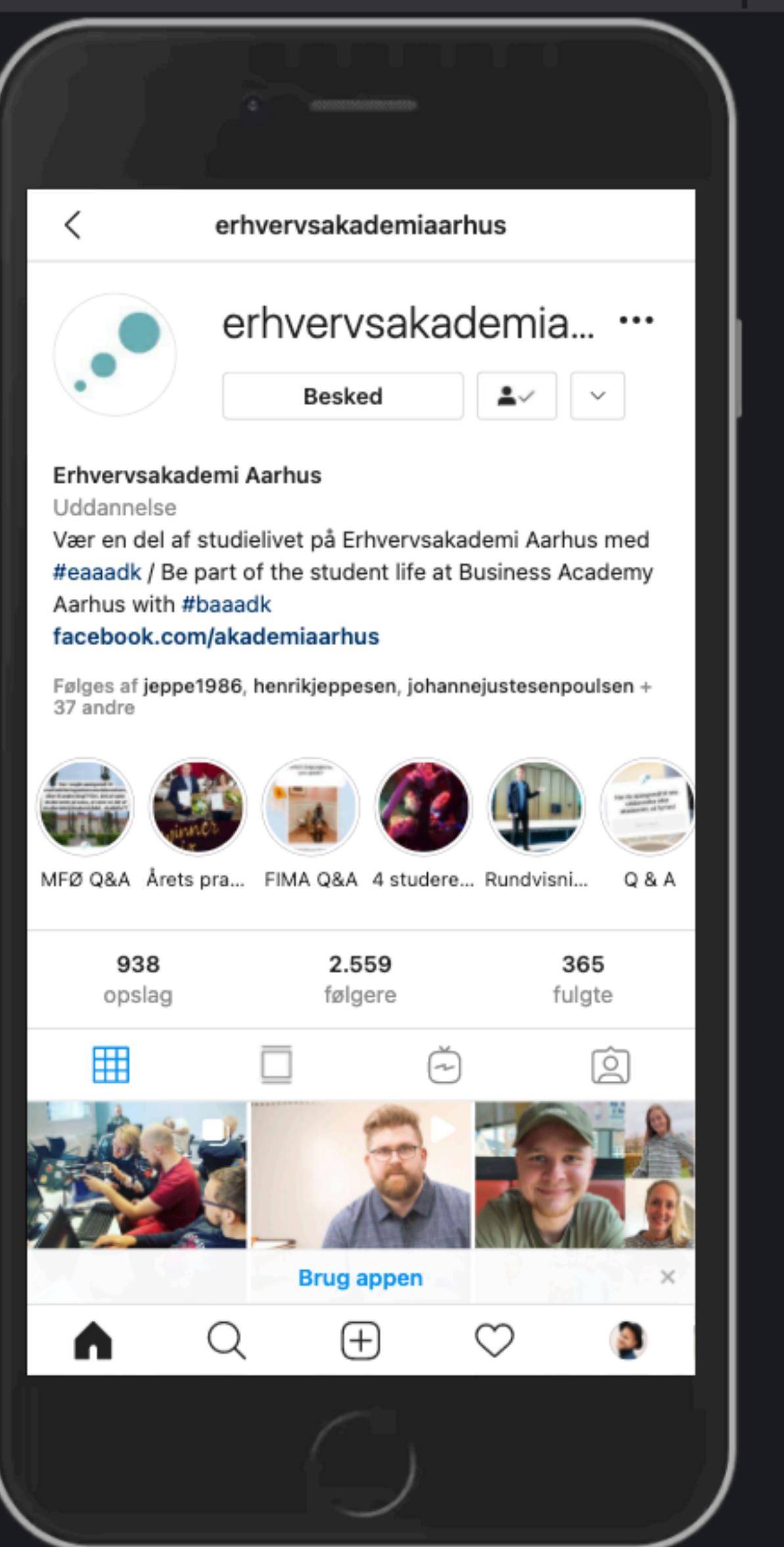
# ARRAYS

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



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





DR | Nyheder - Breaking - TV

dr.dk

NYHEDER DRTV DR LYD KONTAKT DR

DR1: Løvens Hule DR3: Nationens stærkste P1: LSD kælderen DR LYD: Annas Margrethe DR3: Du fucker med de forkerte A Very British Scandal

Seneste nyt

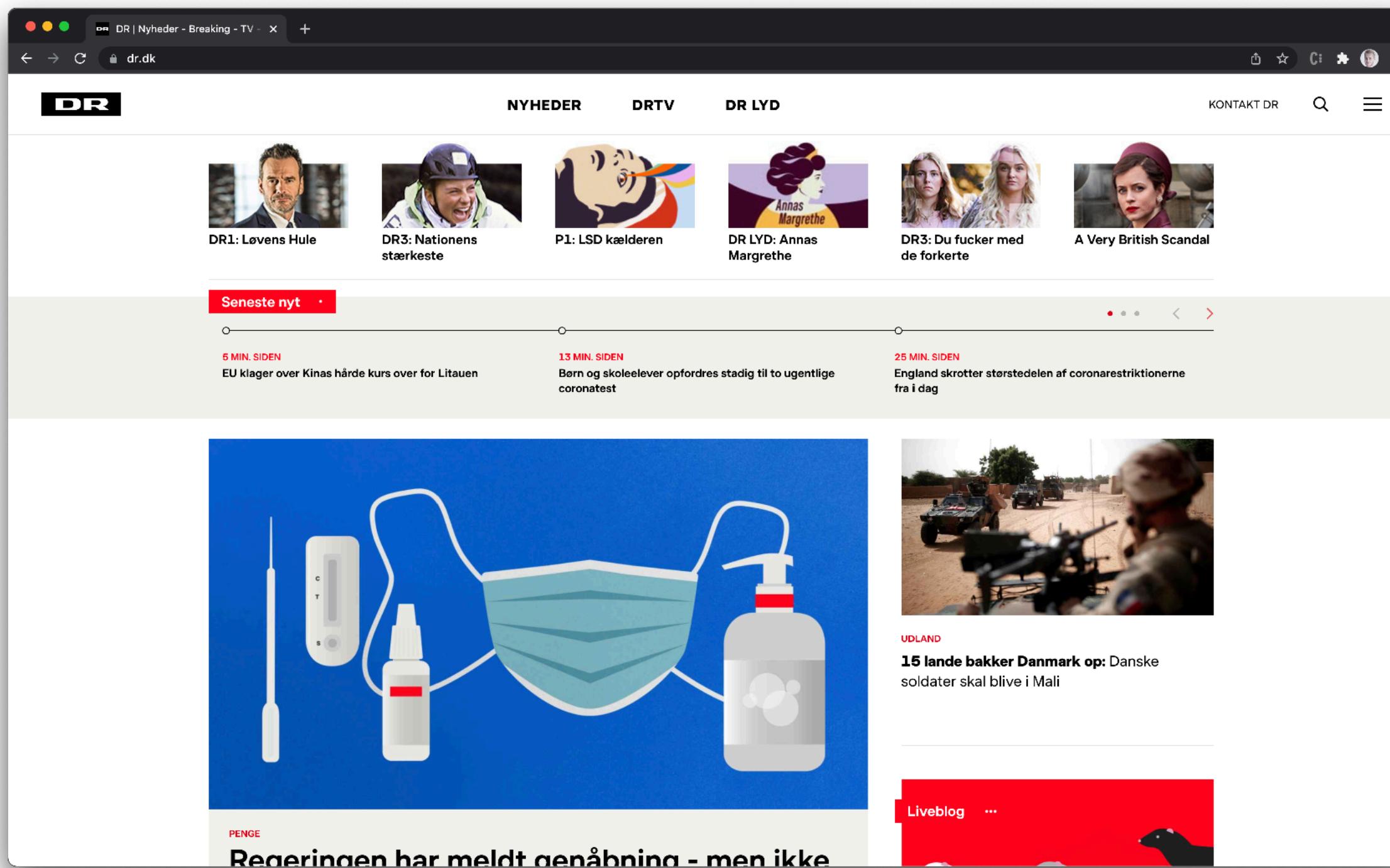
5 MIN. SIDEN EU klager over Kinas hårde kurs over for Litauen 13 MIN. SIDEN Barn og skoleelever opfordres stadig til ugentlige coronatest 25 MIN. SIDEN England skrætter størstedelen af coronarestriktionerne fra i dag

UDLAND 15 lande bakker Danmark op: Danske soldater skal blive i Mali

PENGE

Regeringen har meldt genåbning - men ikke

Liveblog



Erhvervsakademiuuddannelse

ERHVERVSAKADEMI AARHUS DA / INT Q

Byggekoordinator (lukket)

2-årig erhvervsakademiuuddannelse  
Uddannelsen optager ikke nye studerende fra og med 2022.

Damatiker

2-årig erhvervsakademiuuddannelse  
Få en bred viden inden for systemudvikling og programvare, og bliv kvalificeret på til et job med udvikling, implementering og drift af IT-systemer i virksomheden.

Financial controller

2-årig erhvervsakademiuuddannelse  
Bliv klar til et job i en økonomialdelse eller som revisor. Gør karriere inden for fx regnskab, likviditetsstyring, årsrapporter, skat, moms og husholdninger.

Finansøkonom

2-årig erhvervsakademiuuddannelse  
Bliv rustet til en karriere i investerings-, forsikrings- eller ejendomsbranchen - eller i kreditinstitutter. Du bliver en god rådgiver, der løser kundebehov med din virksomhedens konkrete

It-teknolog

2-årig erhvervsakademiuuddannelse  
Brænder du for at følge med i udviklingen af den nyeste teknologi? Så gør karriere inden for computer-, server- og netværksteknologi eller elektronik.

Jordbrugsteknolog

2-årig erhvervsakademiuuddannelse  
Bliv specialist inden for miljø og natur, jordbruksøkonomi og driftsledelse, husdyrproduktion, landskab og anlæg og planteproduktion.

Laborant

2½-årig erhvervsakademiuuddannelse  
Få job i et kontrol-, forsknings- eller udviklingslaboratorium og arbejd inden for fx fødevaresikkerhed, miljøbeskyttelse, medicinalindustrien eller bioteknologi.

Markedsføringsøkonom

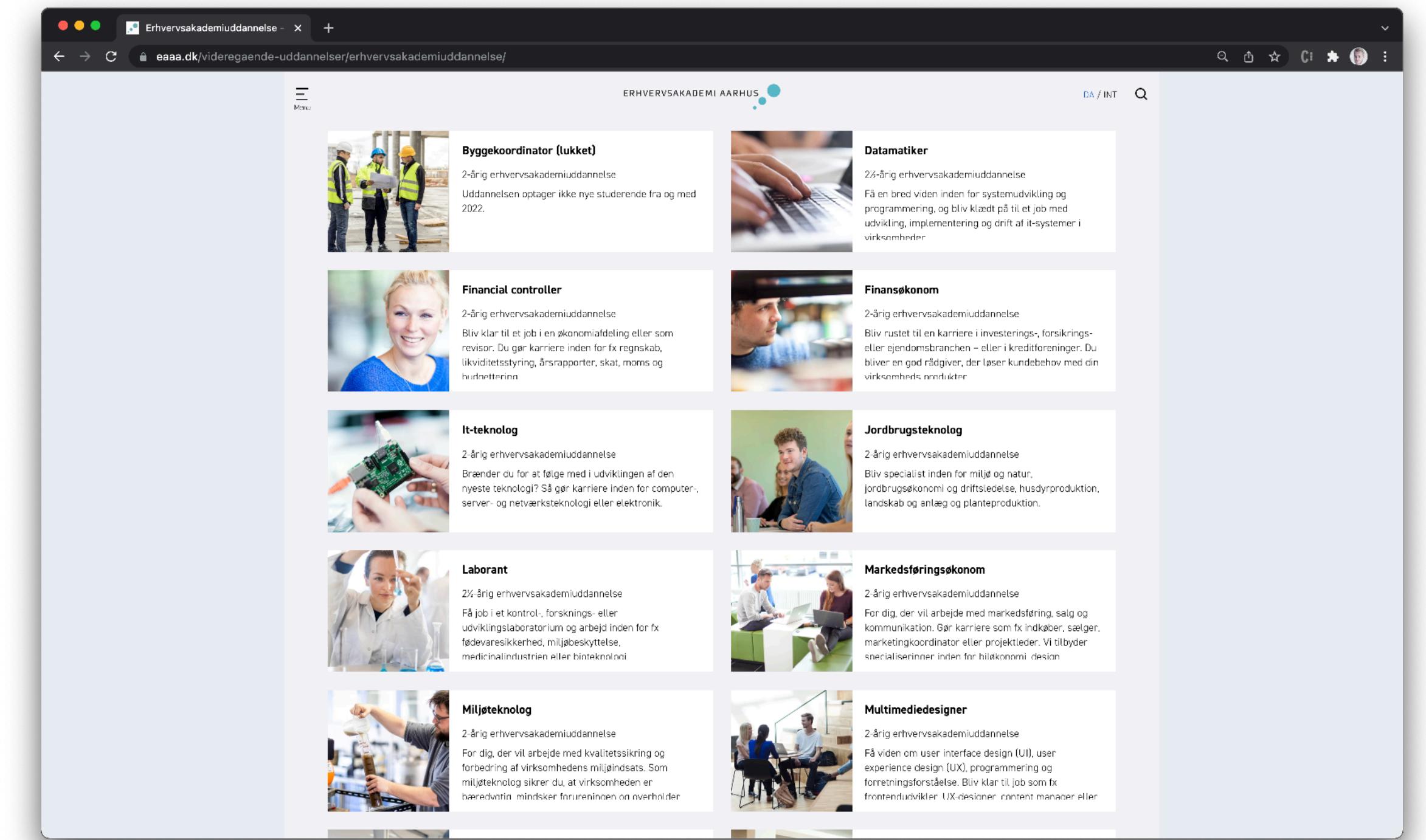
2-årig erhvervsakademiuuddannelse  
For dig, der vil arbejde med markedsføring, salg og kommunikation. Gør karriere som fx indkøber, sælger, marketingkoordinator eller projektleder. Vi tilbyder specialiseringer inden for mikroøkonomi, design

Miljøteknolog

2-årig erhvervsakademiuuddannelse  
For dig, der vil arbejde med kvalitetssikring og forbedring af virksomhedens miljøbelast. Som miljøteknolog sikrer du, at virksomheden er bæredygtigt mindsker forurenningen og overholder

Multimediedesigner

2-årig erhvervsakademiuuddannelse  
Få viden om user interface design (UI), user experience design (UX), programmering og formændsforståelse. Bliv klar til job som fx frontendenudvikler, UX-designer, content manager eller



Course roster: WU-F22v - 1. se

aaa.instructure.com/courses/14390/users

WU-F22v > People

Student view

Home Announcements People Modules BigBlueButton Zoom Assignments Discussions 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	To A
Diana Riis Myjak Andersen	eaadimy@students.eaaa.dk	WU-F22v - 2.	Student	Student	30 Jan at 20:14	40
Haya Barakat	eaahbar@students.eaaa.dk	WU-F22v - 2.	Student	Student	31 Jan at 15:05	18
Daniel Birkus	eaadabi@students.eaaa.dk	WU-F22v - 2.	Student	Student	26 Jan at 11:38	09
Rasmus Cederdorff	race@eaaa.dk	WU-F22v - 2.	Teacher	Teacher	31 Jan at 18:06	58
Nicklas Andié Christensen	eaanac@students.eaaa.dk	WU-F22v - 2.	Student	Student	31 Jan at 13:18	1
Andreas Bastian Dalsaa	eaaandd@students.eaaa.dk	WU-F22v - 2.	Student	Student		
Patrik Ádám Dienes	eaapad@students.eaaa.dk	WU-F22v - 2.	Student	Student		
Mohammad Hassan Bozorgi	eaamhbg@students.eaaa.dk	WU-F22v - 2.	Student	Student	31 Jan at 15:57	39

Filter Invert Hide data URLs

All Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other Has blocked cookies

Blocked Requests 3rd-party requests

500 ms 1000 ms 1500 ms 2000 ms 2500 ms 3000 ms 3500 ms 4000 ms 4500 ms 5000 ms

Name Headers Payload Preview Response Initiator Timing Cookies

[{id: "21628", name: "Diana Riis Myjak Andersen", created\_at: "2020-01-18T02:53:30+01:00", analytics\_url: "/courses/14390/analytics/users/21628", avatar\_url: "https://eaaa.instructure.com/imagesthumbnails/781064/21628", created\_at: "2020-01-18T02:53:30+01:00", custom\_links: [{text: "New Analytics", ...}], email: "eaadimy@students.eaaa.dk", enrollments: [{course\_id: "14390", id: "369475", user\_id: "21628", id: "21628"}, integration\_id: null, login\_id: "eaadimy@students.eaaa.dk", name: "Diana Riis Myjak Andersen", short\_name: "Diana Riis Myjak Andersen (EAADIMY)", sis\_user\_id: null, sortable\_name: "Andersen, Diana Riis Myjak"}]

1: {id: "33983", name: "Haya Barakat", created\_at: "2021-12-23T00:47:10+01:00"}  
2: {id: "33989", name: "Daniel Birkus", created\_at: "2021-12-23T00:47:10+01:00"}  
3: {id: "14427", name: "Rasmus Cederdorff", created\_at: "2018-09-06T08:45:00+02:00"}  
4: {id: "33982", name: "Nicklas Andié Christensen", created\_at: "2021-12-23T00:47:10+01:00"}  
5: {id: "34637", name: "Andreas Bastian Dalsaa", created\_at: "2022-01-10T08:45:00+01:00"}  
6: {id: "33993", name: "Patrik Ádám Dienes", created\_at: "2021-12-23T00:47:10+01:00"}  
7: {id: "33987", name: "Mohammad Hassan Bozorgi Gazani", created\_at: "2021-12-23T00:47:10+01:00"}  
8: {id: "19321", name: "Sebastian Bagger Hadi", created\_at: "2019-08-10T08:45:00+02:00"}  
9: {id: "33984", name: "Zaharia Horatau-Hostiuc", created\_at: "2021-12-23T00:47:10+01:00"}  
10: {id: "15576", name: "Nicolai Buus Palmkvist Johannessen", created\_at: "2021-12-23T00:47:10+01:00"}  
11: {id: "23531", name: "Søren Bo Jørgensen", created\_at: "2020-08-04T08:45:00+02:00"}  
12: {id: "33980", name: "Razvan Andrei Jumuga", created\_at: "2021-12-23T00:47:10+01:00"}  
13: {id: "117", name: "Anne Kirketerp", created\_at: "2017-05-10T14:09:00+02:00"}  
14: {id: "21858", name: "Jesper Kragh", created\_at: "2020-01-24T03:07:00+01:00"}  
15: {id: "21119", name: "Dina Heiberg Madsen", created\_at: "2019-12-10T08:45:00+01:00"}  
16: {id: "5410", name: "Delyan Antoanov Mitev", created\_at: "2017-08-10T08:45:00+02:00"}  
17: {id: "18548", name: "Marcus Olrik", created\_at: "2019-08-15T02:41:00+02:00"}  
18: {id: "18549", name: "Lars Christian Hansen", created\_at: "2019-08-15T02:41:00+02:00"}  
19: {id: "18550", name: "Morten Sørensen", created\_at: "2019-08-15T02:41:00+02:00"}  
20: {id: "18551", name: "Kasper Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
21: {id: "18552", name: "Thomas Rasmussen", created\_at: "2019-08-15T02:41:00+02:00"}  
22: {id: "18553", name: "Mikkel Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
23: {id: "18554", name: "Lars Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
24: {id: "18555", name: "Morten Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
25: {id: "18556", name: "Thomas Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
26: {id: "18557", name: "Mikkel Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
27: {id: "18558", name: "Lars Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
28: {id: "18559", name: "Morten Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
29: {id: "18560", name: "Thomas Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
30: {id: "18561", name: "Mikkel Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
31: {id: "18562", name: "Lars Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
32: {id: "18563", name: "Morten Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
33: {id: "18564", name: "Thomas Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
34: {id: "18565", name: "Mikkel Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
35: {id: "18566", name: "Lars Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
36: {id: "18567", name: "Morten Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
37: {id: "18568", name: "Thomas Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
38: {id: "18569", name: "Mikkel Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
39: {id: "18570", name: "Lars Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
40: {id: "18571", name: "Morten Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
41: {id: "18572", name: "Thomas Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
42: {id: "18573", name: "Mikkel Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
43: {id: "18574", name: "Lars Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
44: {id: "18575", name: "Morten Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
45: {id: "18576", name: "Thomas Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
46: {id: "18577", name: "Mikkel Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
47: {id: "18578", name: "Lars Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
48: {id: "18579", name: "Morten Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
49: {id: "18580", name: "Thomas Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
50: {id: "18581", name: "Mikkel Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
51: {id: "18582", name: "Lars Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
52: {id: "18583", name: "Morten Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
53: {id: "18584", name: "Thomas Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
54: {id: "18585", name: "Mikkel Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
55: {id: "18586", name: "Lars Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
56: {id: "18587", name: "Morten Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
57: {id: "18588", name: "Thomas Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
58: {id: "18589", name: "Mikkel Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
59: {id: "18590", name: "Lars Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
60: {id: "18591", name: "Morten Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
61: {id: "18592", name: "Thomas Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
62: {id: "18593", name: "Mikkel Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
63: {id: "18594", name: "Lars Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
64: {id: "18595", name: "Morten Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
65: {id: "18596", name: "Thomas Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
66: {id: "18597", name: "Mikkel Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
67: {id: "18598", name: "Lars Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
68: {id: "18599", name: "Morten Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
69: {id: "18600", name: "Thomas Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
70: {id: "18601", name: "Mikkel Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
71: {id: "18602", name: "Lars Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
72: {id: "18603", name: "Morten Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
73: {id: "18604", name: "Thomas Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
74: {id: "18605", name: "Mikkel Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
75: {id: "18606", name: "Lars Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
76: {id: "18607", name: "Morten Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
77: {id: "18608", name: "Thomas Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
78: {id: "18609", name: "Mikkel Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
79: {id: "18610", name: "Lars Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
80: {id: "18611", name: "Morten Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
81: {id: "18612", name: "Thomas Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
82: {id: "18613", name: "Mikkel Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
83: {id: "18614", name: "Lars Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
84: {id: "18615", name: "Morten Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
85: {id: "18616", name: "Thomas Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
86: {id: "18617", name: "Mikkel Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
87: {id: "18618", name: "Lars Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
88: {id: "18619", name: "Morten Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
89: {id: "18620", name: "Thomas Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
90: {id: "18621", name: "Mikkel Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
91: {id: "18622", name: "Lars Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
92: {id: "18623", name: "Morten Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
93: {id: "18624", name: "Thomas Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
94: {id: "18625", name: "Mikkel Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
95: {id: "18626", name: "Lars Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
96: {id: "18627", name: "Morten Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
97: {id: "18628", name: "Thomas Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
98: {id: "18629", name: "Mikkel Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
99: {id: "18630", name: "Lars Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
100: {id: "18631", name: "Morten Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
101: {id: "18632", name: "Thomas Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
102: {id: "18633", name: "Mikkel Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
103: {id: "18634", name: "Lars Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
104: {id: "18635", name: "Morten Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
105: {id: "18636", name: "Thomas Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
106: {id: "18637", name: "Mikkel Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
107: {id: "18638", name: "Lars Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
108: {id: "18639", name: "Morten Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
109: {id: "18640", name: "Thomas Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
110: {id: "18641", name: "Mikkel Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
111: {id: "18642", name: "Lars Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
112: {id: "18643", name: "Morten Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
113: {id: "18644", name: "Thomas Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
114: {id: "18645", name: "Mikkel Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
115: {id: "18646", name: "Lars Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
116: {id: "18647", name: "Morten Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
117: {id: "18648", name: "Thomas Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
118: {id: "18649", name: "Mikkel Mikkelsen", created\_at: "2019-08-15T02:41:00+02:00"}  
119: {id: "18650", name: "Lars Mikkelsen", created\_at: "2019-08-15T02:41:00+

IT'S ALL  
OBJECTS &  
ARRAYS

# DATA TYPES (& 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
```



# 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>

# ARRAY

.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](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)

# 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](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>`;  
  }  
}
```

# `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>;
  }
}
```

# CANVAS USERS CASE #1

BACK

## CREATE A GRID DISPLAYING USERS FROM CANVAS

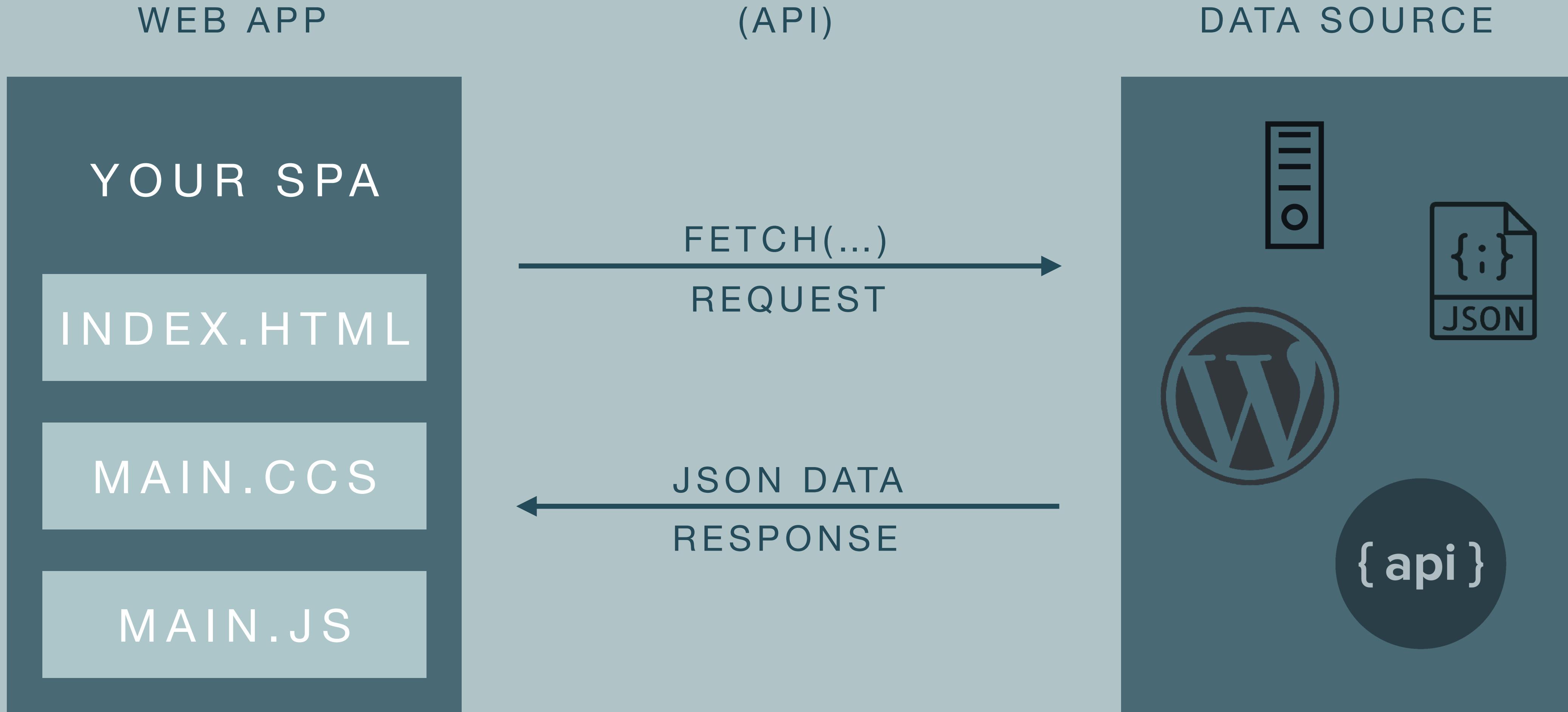
1. Make a copy of your previous project.
2. In your `app.js` define a variable called `users` holding an array.
3. Inside of the array, define at least 4 user objects with properties similar to the data structure from <https://cederdorff.github.io/web-frontend/canvas-users/data.json> (no use of `fetch` yet).
4. Use `console.log()` to test the array.
5. Create a function called `appendUsers(...)`. The function must loop through the `users` array and append each user to your grid container in the DOM.
  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 the static defined from the previous exercise.
  2. Delete the static define HTML content in your grid.
  3. Make sure to execute the function and that all users are appended to the DOM.

# FETCH( ... )

HTTP REQUEST IN JAVASCRIPT

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

# F E T C H



# 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();
```

# CANVAS USERS CASE #2

BACK

## FETCH USERS FROM JSON & DISPLAY IN GRID

1. Make a copy of your previous project.
2. In your app.js create a function called `fetchUsers()`.
3. In `fetchUsers()` implement the logic to fetch users from the following url: <https://cederdorff.github.io/web-frontend/canvas-users/data.json>
4. Use `console.log` to test the result (the fetched data).
5. Remove the *in script* defined `users` array (from previous exercise) and save the fetched result in the `users` variable.
6. Execute and use `appendUsers(...)` to append the newly fetched users from the json file.

# CANVAS USERS CASE #3

## EXTRA

1. Implement search functionality searching on `name.includes(searchValue)`.
2. Implement sort functionality in order to sort by `name, sortableName, etc.`
3. Make it possible to filter users on `enrollment_type` (students and teachers)
4. Implement create functionality using `array.push()`. Remember generating 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.

BACK

# NEXT UP

MORE JAVASCRIPT

WEB APP DEVELOPMENT

SINGLE PAGE APPLICATIONS

MORE FETCH, JSON & ASYNC JS

CODE EVERY DAY!