

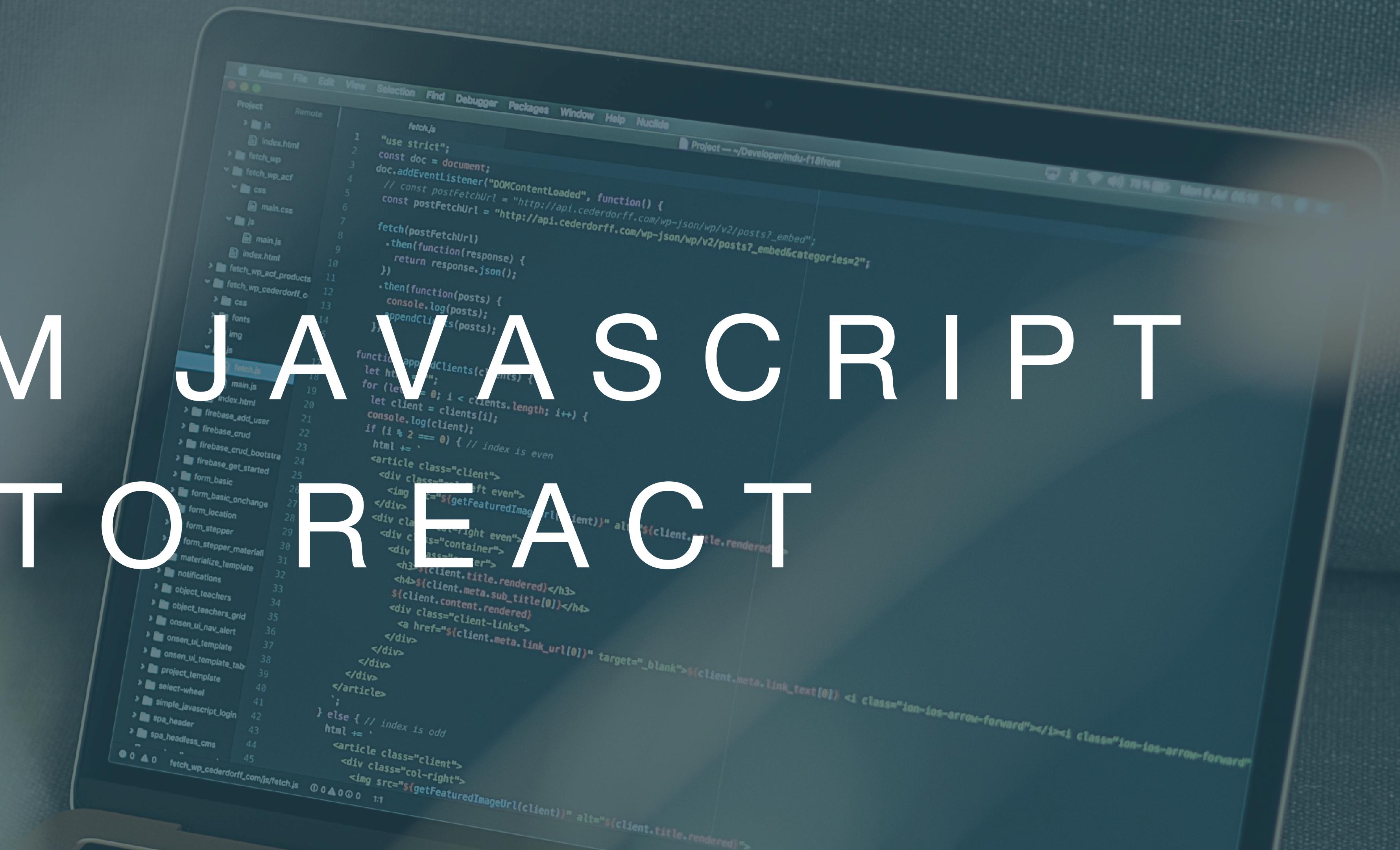
race.js is loading...



Check out: [spa-canvas-users](#)

FRONTEND DEVELOPER

# FROM JAVASCRIPT TO REACT



# A G E N D A

Wrap up: Canvas  
Users Case & Fetch

Function-based JS &  
JS Basics

Web App, SPA &  
Routing

Canvas Users SPA  
Case

# EXERCISES

Wrap up: Canvas Users Case

Canvas Users SPA Case

Product SPA

# WRAP UP: CANVAS USERS CASE

Canvas Users

127.0.0.1:5500/canvas-users/

Barbora Byrtusová <a href="mailto:eaababy@students.eaaa.dk">eaababy@students.eaaa.dk</a> Role: StudentEnrollment 	Rasmus Cederdorff <a href="mailto:race@eaaa.dk">race@eaaa.dk</a> Role: TeacherEnrollment 	Sarah Øster Dybvad <a href="mailto:eaasody@students.eaaa.dk">eaasody@students.eaaa.dk</a> Role: StudentEnrollment 
Kristine Dzumakajeva <a href="mailto:eaakrdz@students.eaaa.dk">eaakrdz@students.eaaa.dk</a> Role: StudentEnrollment 	Wojciech Arkadiusz Dzwonczyk <a href="mailto:eaawodz@students.eaaa.dk">eaawodz@students.eaaa.dk</a> Role: StudentEnrollment 	Jesson Alsaybar Getapal <a href="mailto:eaajage@students.eaaa.dk">eaajage@students.eaaa.dk</a> Role: StudentEnrollment 
Marian Alexandru Hanghiuc <a href="mailto:eaamahang@students.eaaa.dk">eaamahang@students.eaaa.dk</a> Role: StudentEnrollment 	Lasse Bundsgaard Hansen <a href="mailto:eaalasbh@students.eaaa.dk">eaalasbh@students.eaaa.dk</a> Role: StudentEnrollment 	Mads Villads Hoe <a href="mailto:eaamvho@students.eaaa.dk">eaamvho@students.eaaa.dk</a> Role: StudentEnrollment 
		

# 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.

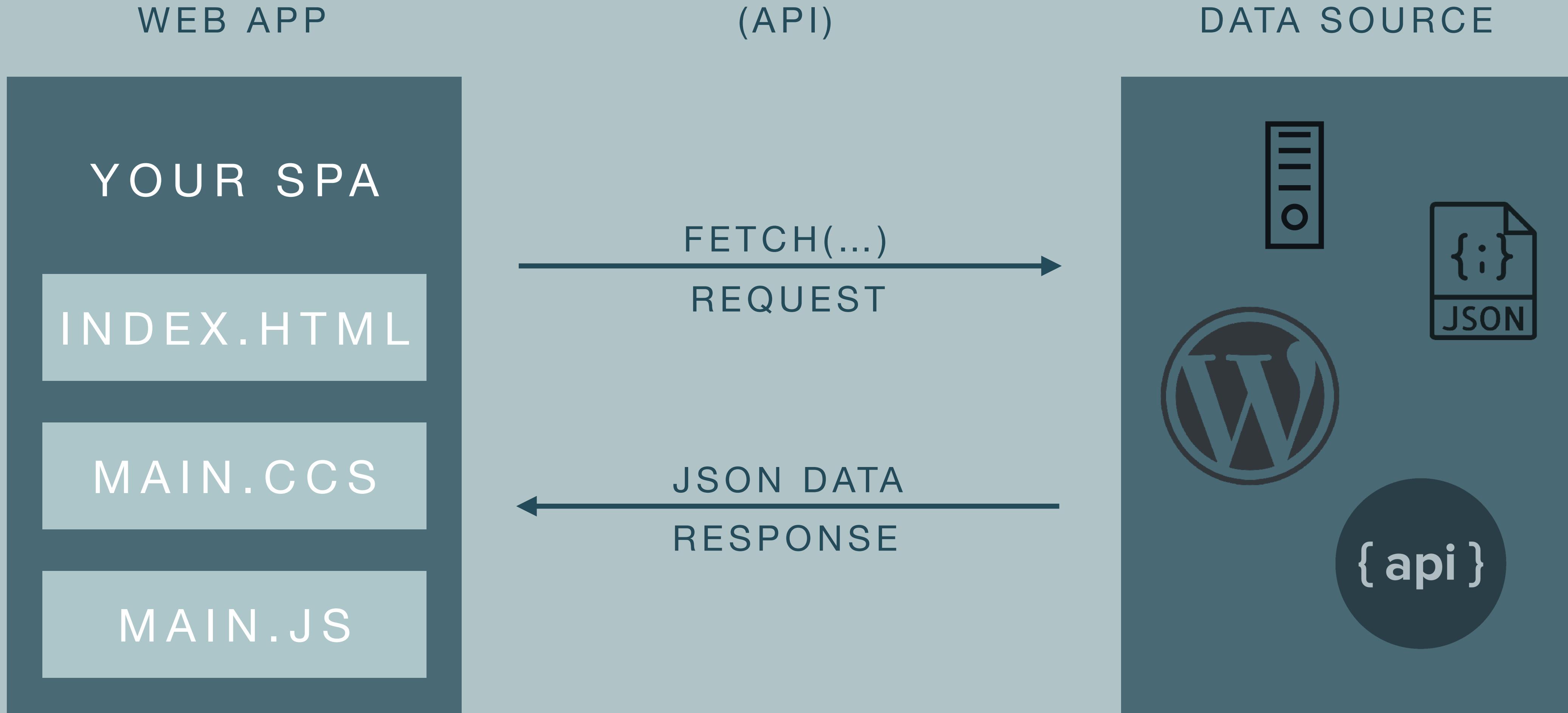
# FETCH( ... )

HTTP REQUEST IN JAVASCRIPT

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

```
fetch('json/persons.json')
  .then(function(response) {
    return response.json();
  })
  .then(function(json) {
    console.log(json);
    appendPersons(json);
  });
}
```

# 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"
},
{
```

# JSON

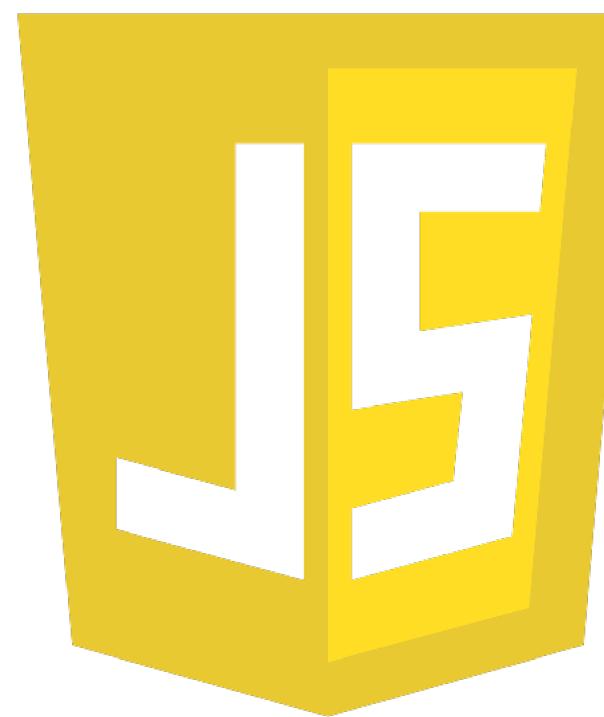
... A SYNTAX FOR STORING AND EXCHANGING DATA  
OVER THE WEB

# SEPARATION OF CONCERNS

**HTML**



**JS**



**CSS**



**JSON**



STRUCTURE  
CONTENT

FUNCTIONALITY  
BEHAVIOR

LAYOUT  
PRESENTATION

DATA  
PERSISTENCE

# JAVASCRIPT OBJECT NOTATION

- Collection of key-value pair: "key": "value"
- List of values, collections or objects
- Lightweight data-interchange format
- Syntax for storing and exchanging data over the web
- Human and machine readable **text**: small, fast and simple
- Language independent
- Can be parsed directly to JavaScript Object
- JavaScript Objects can be converted directly to JSON

```
[{  
  "name": "Peter Hansen",  
  "age": 52,  
  "hairColor": "brown",  
  "relation": "dad"  
}, {  
  "name": "Alicia Hansen",  
  "age": 51,  
  "hairColor": "blonde",  
  "relation": "mom"  
}, {  
  "name": "Martin Hansen",  
  "age": 22,  
  "hairColor": "dark",  
  "relation": "brother"  
}, {  
  "name": "Jesper Petersen",  
  "age": 35,  
  "hairColor": "brown",  
  "relation": "uncle"  
}]
```

# JSON

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

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

JSON OBJECT

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

LIST OF JSON OBJECTS

# JSON

FORMATTER & VALIDATOR

[HTTPS://JSONFORMATTER.CURIOSCONCEPT.COM](https://jsonformatter.curiousconcept.com)

# FETCH

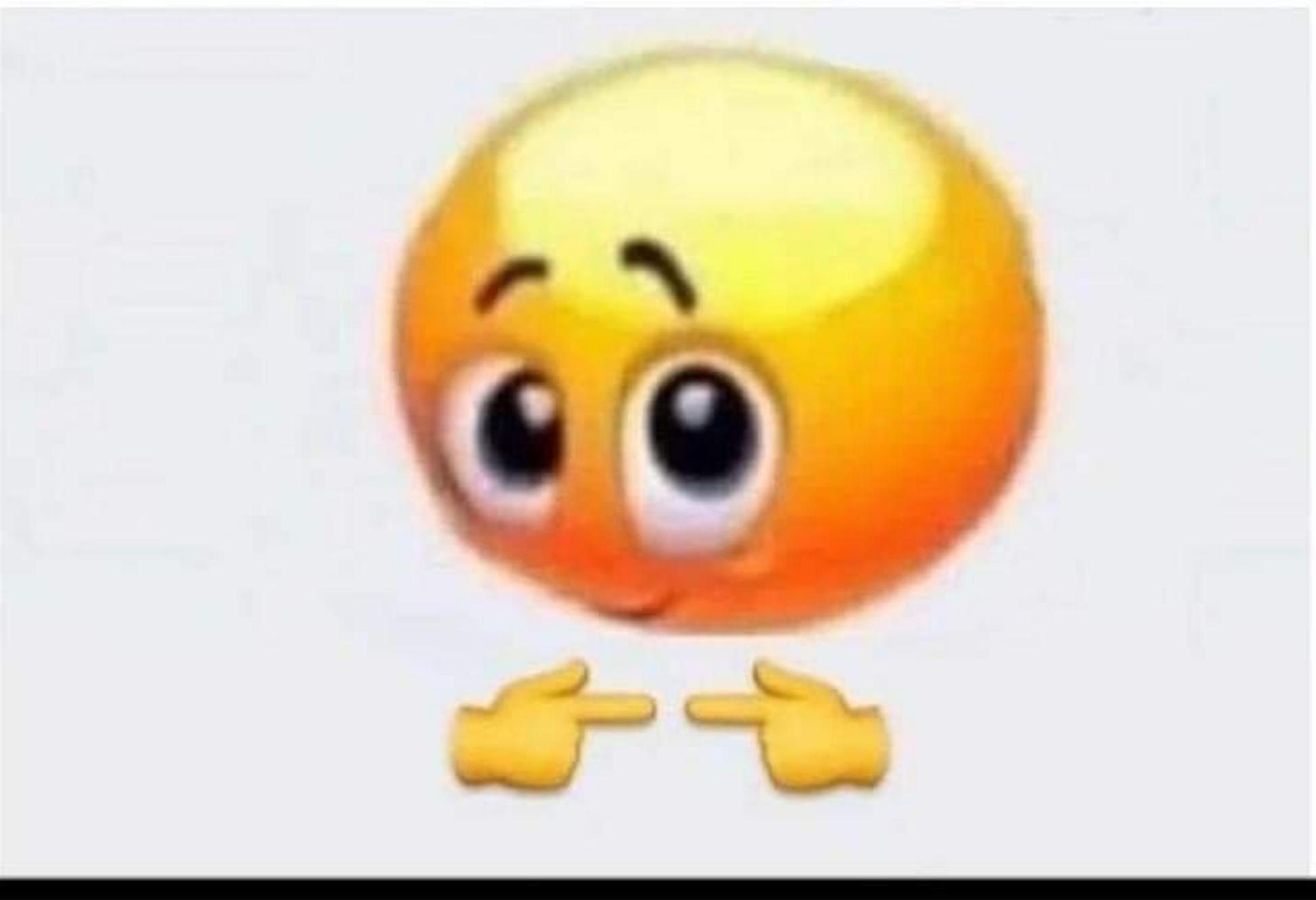
## JSON, DATA SOURCES & API'S

```
// 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();
```

# FETCH WITH CALLBACKS OR ASYNC/AWAIT

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

will you be async to my await?



# ASYNC / AWAIT

WORKING WITH PROMISES

WAIT FOR A PROMISE TO FINISH

# ASYNC / AWAIT

MAKING ASYNCHRONOUS PROGRAMMING EASIER  
WITH ASYNC AND AWAIT

"ASYNC AND AWAIT MAKE  
PROMISES EASIER TO WRITE"

**ASYNC** MAKES A FUNCTION RETURN A PROMISE

**AWAIT** MAKES A FUNCTION WAIT FOR A PROMISE

[https://www.w3schools.com/js/js\\_async.asp](https://www.w3schools.com/js/js_async.asp)

# **FETCH RETURNS A PROMISE**

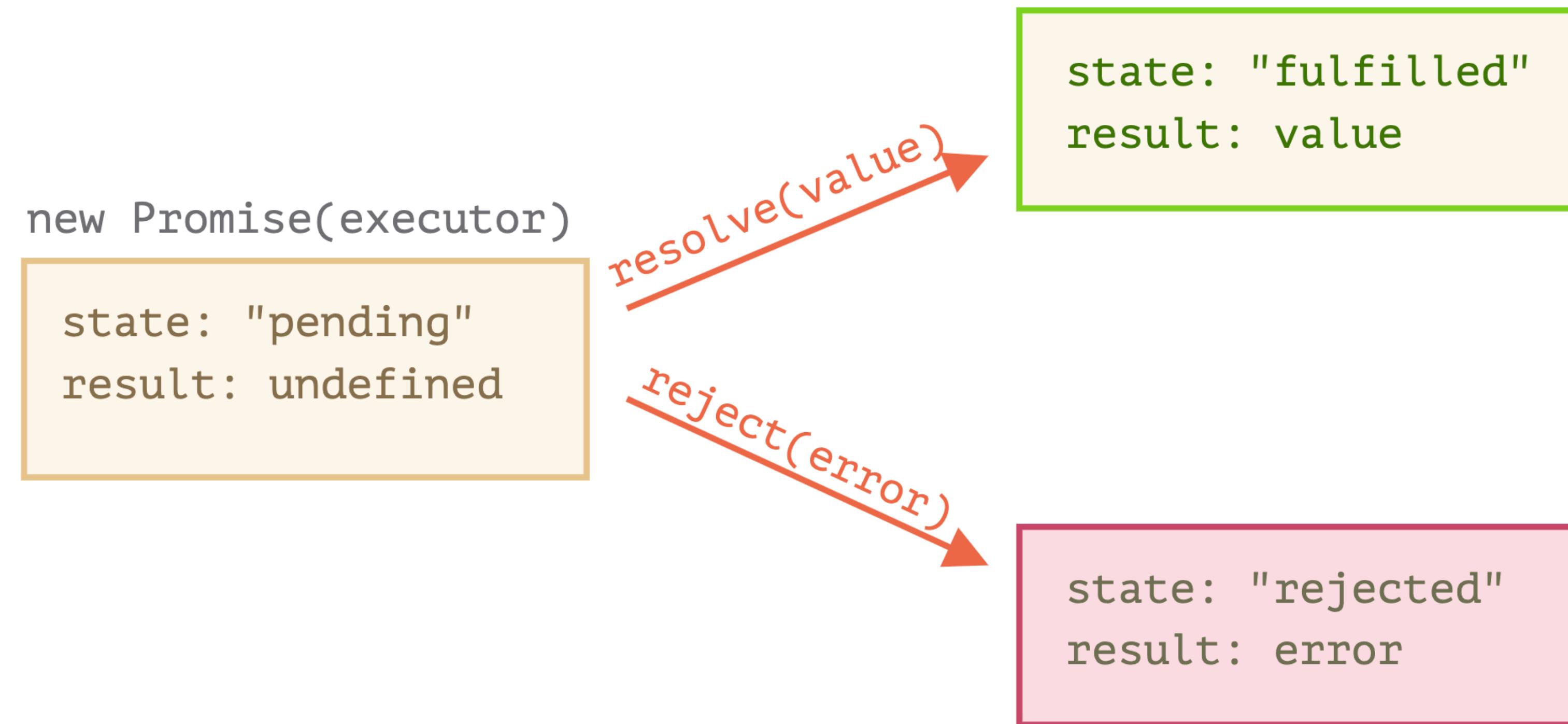
**ASYNC MAKES A FUNCTION RETURN A PROMISE**

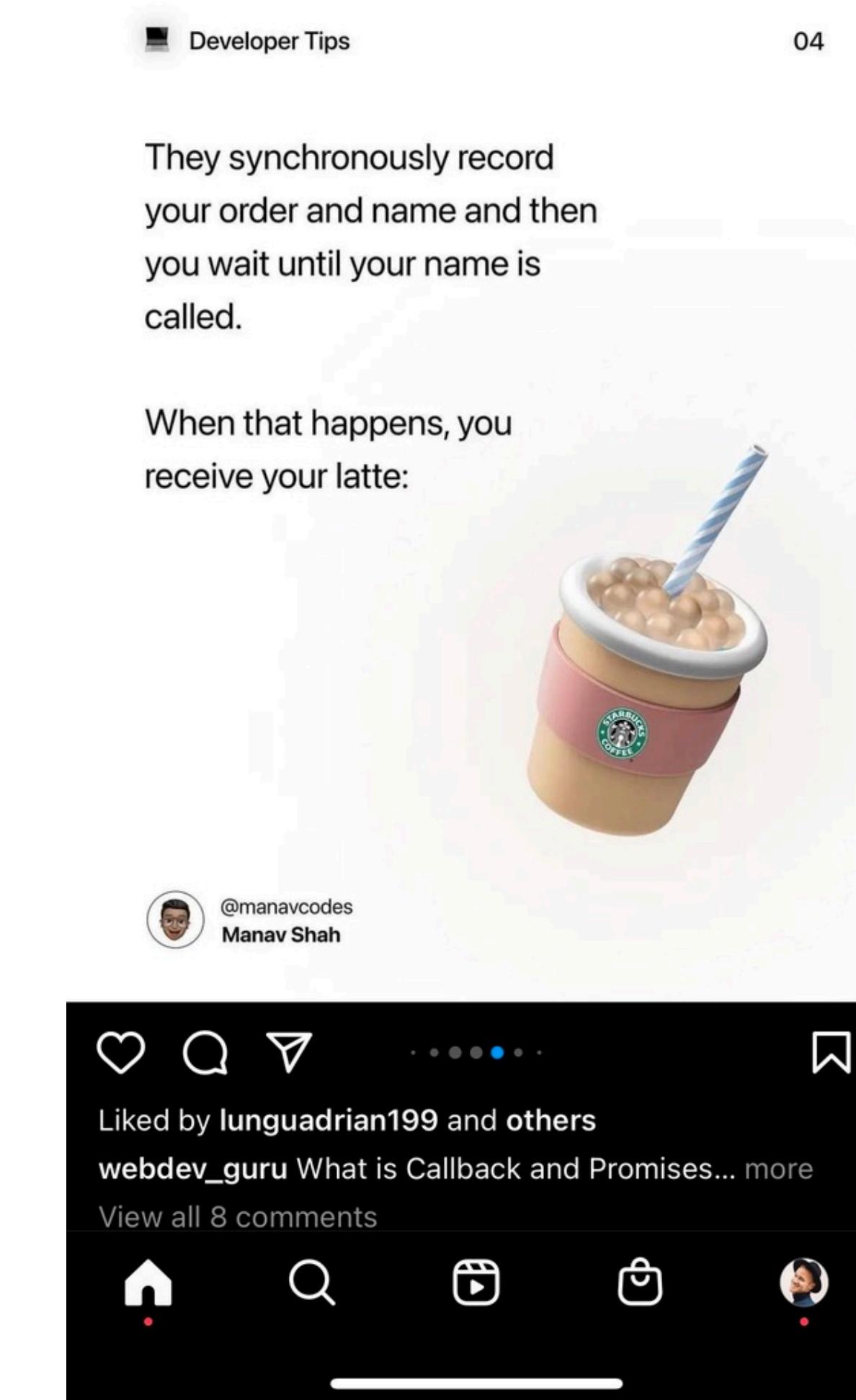
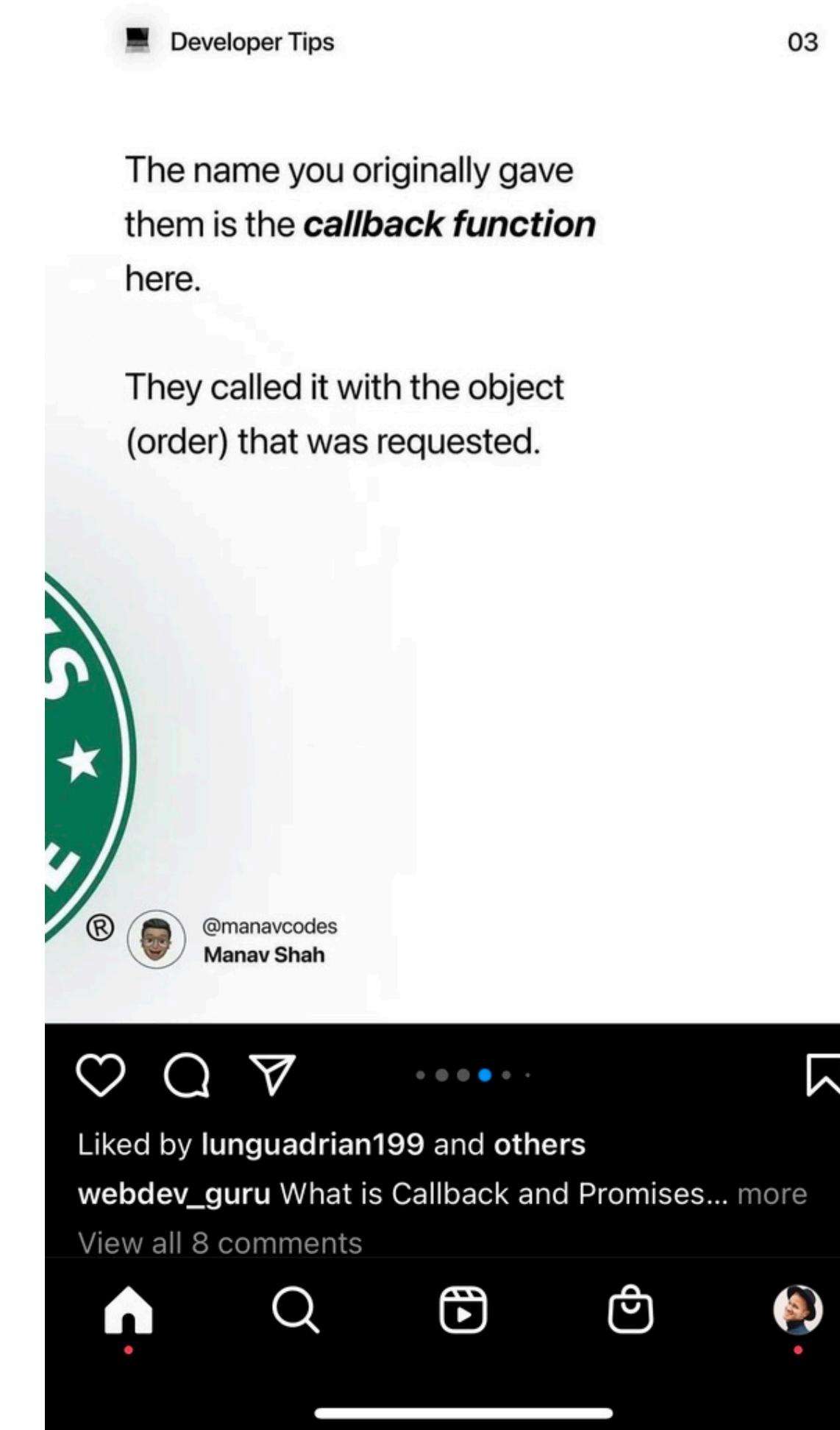
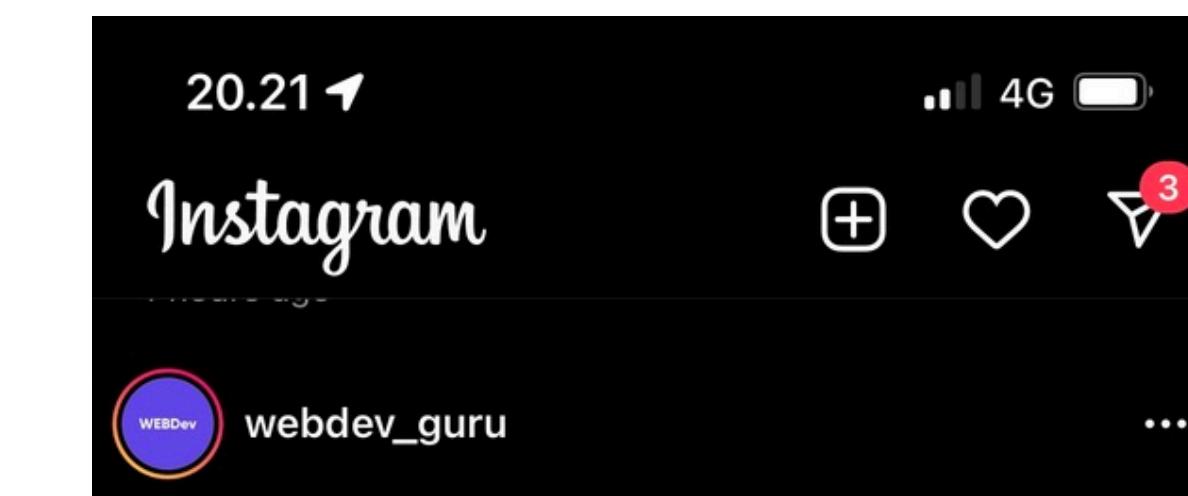
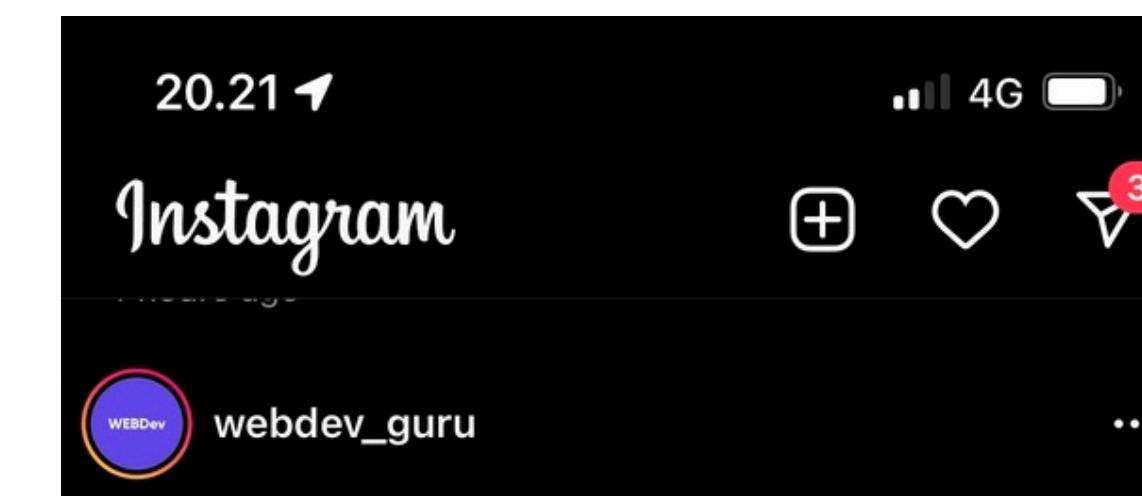
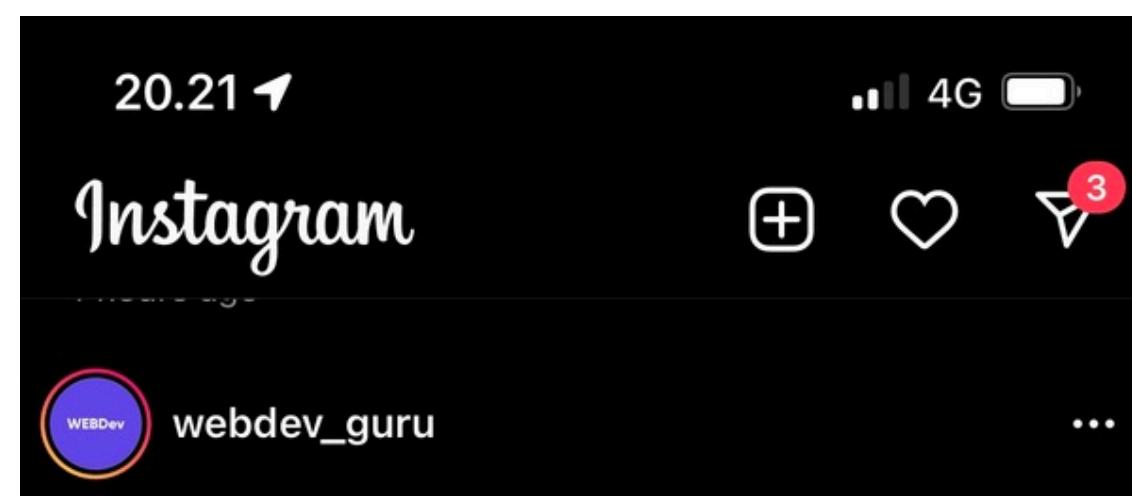
**AWAIT MAKES A FUNCTION WAIT FOR A PROMISE**

**WE CAN USE AWAIT TO WAIT FOR FETCH TO FINISH**

[https://www.w3schools.com/js/js\\_async.asp](https://www.w3schools.com/js/js_async.asp)

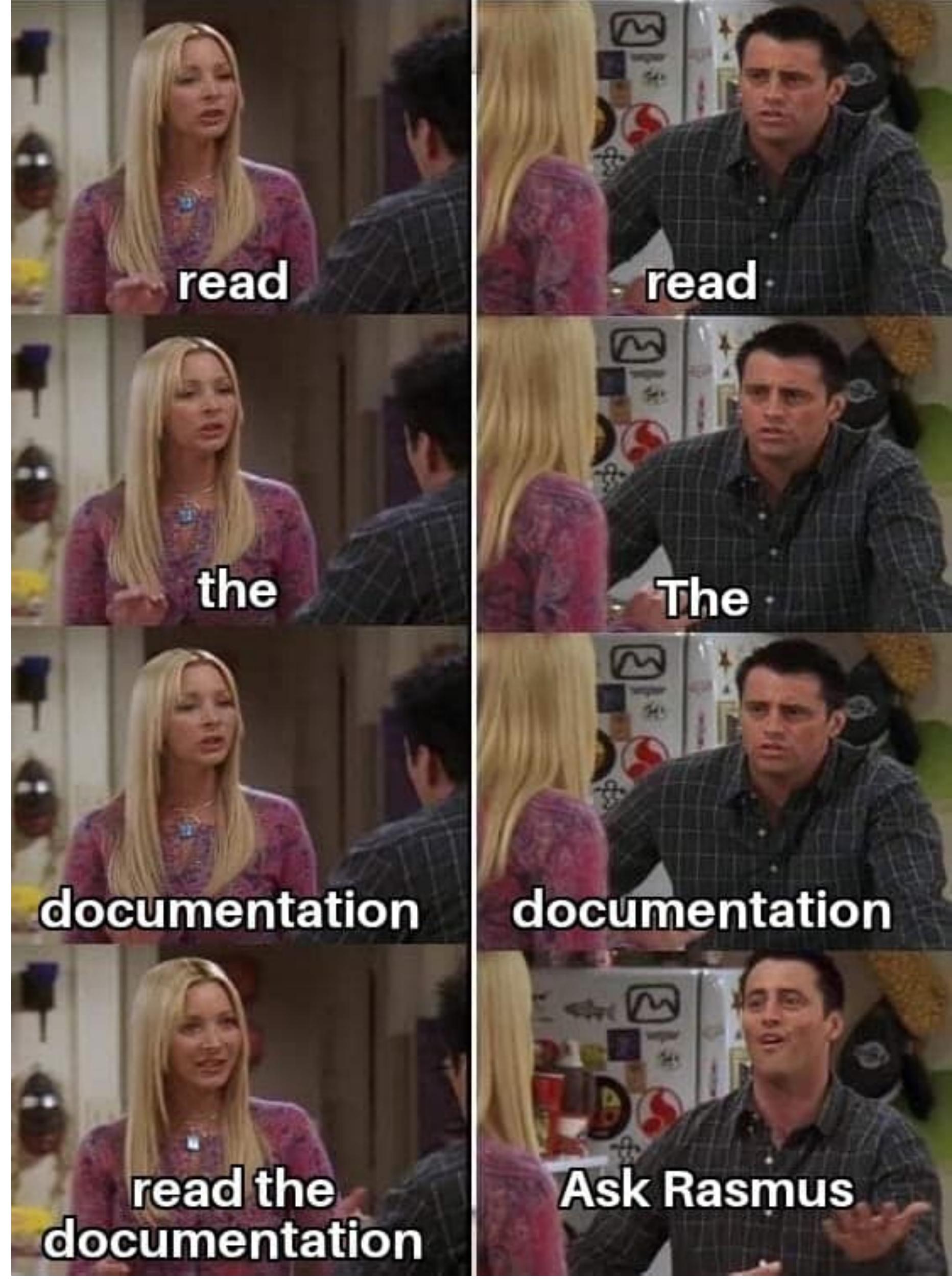
# THE TWO STATES OF PROMISE





```
let myPromise = new Promise(function (resolve, reject) {
    // "Producing Code" (May take some time)
    // Making coffee ...
    resolve("Yaaay, here's your coffee"); // when successful
    reject("Ohh, f***. Something went wrong"); // when error
});

// "Consuming Code" (Must wait for a fulfilled Promise)
myPromise.then(
    function (value) {
        /* code if successful */
        console.log(value); // Yaaay, here's your coffee
    },
    function (error) {
        /* code if some error */
        console.log(error); // Ohh, f***. Something went wrong
    }
);
```



# GLOBAL VARIABLES

VARIABLES OUTSIDE A FUNCTION (AND SCOPES)  
ARE GLOBAL VARIABLES

## Local variables

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

For example:

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

## Outer variables

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

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

The function has full access to the outer variable. It can modify it as well.

```
let userName = 'John';

function showMessage() {
    userName = "Bob"; // (1) changed the outer variable

    let message = 'Hello, ' + userName;
    alert(message);
}

alert( userName ); // John before the function call

showMessage();

alert( userName ); // Bob, the value was modified by the function
```

## Global variables

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

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

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

Global  
Variable

```
let _movies = [];

// fetch all movies from WP
async function getMovies() {
    let response = await fetch("https://movie-api.cederdorff.com/wp-json/wp/v2/posts");
    let data = await response.json();
    console.log(data);
    _movies = data;
    appendMovies(data);
    showLoader(false);
}

getMovies();
```

ARRAY  
movies

```
// append movies to the DOM
function appendMovies(movies) {
  let htmlTemplate = '';
  for (let movie of movies) {
    htmlTemplate += `
      <article>
        <h2>${movie.title.rendered} (${movie.acf.year})</h2>
        
        <p>${movie.acf.description}</p>
        <iframe src="${movie.acf.trailer}"></iframe>
      </article>
    `;
  }
  document.querySelector('#movies-container').innerHTML = htmlTemplate;
}
```

Inline variable

Store the current object  
from the array

function  
argument

variable inside  
the function

```
let _movies = [];

// fetch all movies from WP
async function getMovies() {
  let response = await fetch("https://movie-api.cederdorff.com/wp-json/wp/v2/posts");
  let data = await response.json();
  console.log(data);
  _movies = data;
  appendMovies(data);
  showLoader(false);
}

getMovies();
```

```
// append movies to the DOM
function appendMovies(movies) {
  let htmlTemplate = "";
  for (let movie of movies) {
    htmlTemplate += `
      <article>
        <h2>${movie.title.rendered} (${movie.acf.year})</h2>
        
        <p>${movie.excerpt.rendered}</p>
      </article>
    `;
  }
  document.querySelector("#content").innerHTML = htmlTemplate;
}
```

function argument

variable inside the function

Inline variable

Store the current object from the array

```
// ===== Product functionality ===== //
/*
global variables: _products, _selectedProductId
*/
let _products = [];
let _selectedProductId;

/*
Fetches json data from the file products.json
*/
async function fetchData() {
    const response = await fetch('json/products.json');
    const data = await response.json();
    _products = data;
    console.log(_products);
    appendProducts(_products);
    showLoader(false);
}

fetchData();

function appendProducts(products) {
    let htmlTemplate = "";
    for (let product of products) {
        htmlTemplate += /*html*/
            <article class="${product.status}">
                <article onclick="showDetailView(${product.id})">
```

FUNCTION - BASED JS  
OR FUNCTIONAL PROGRAMMING

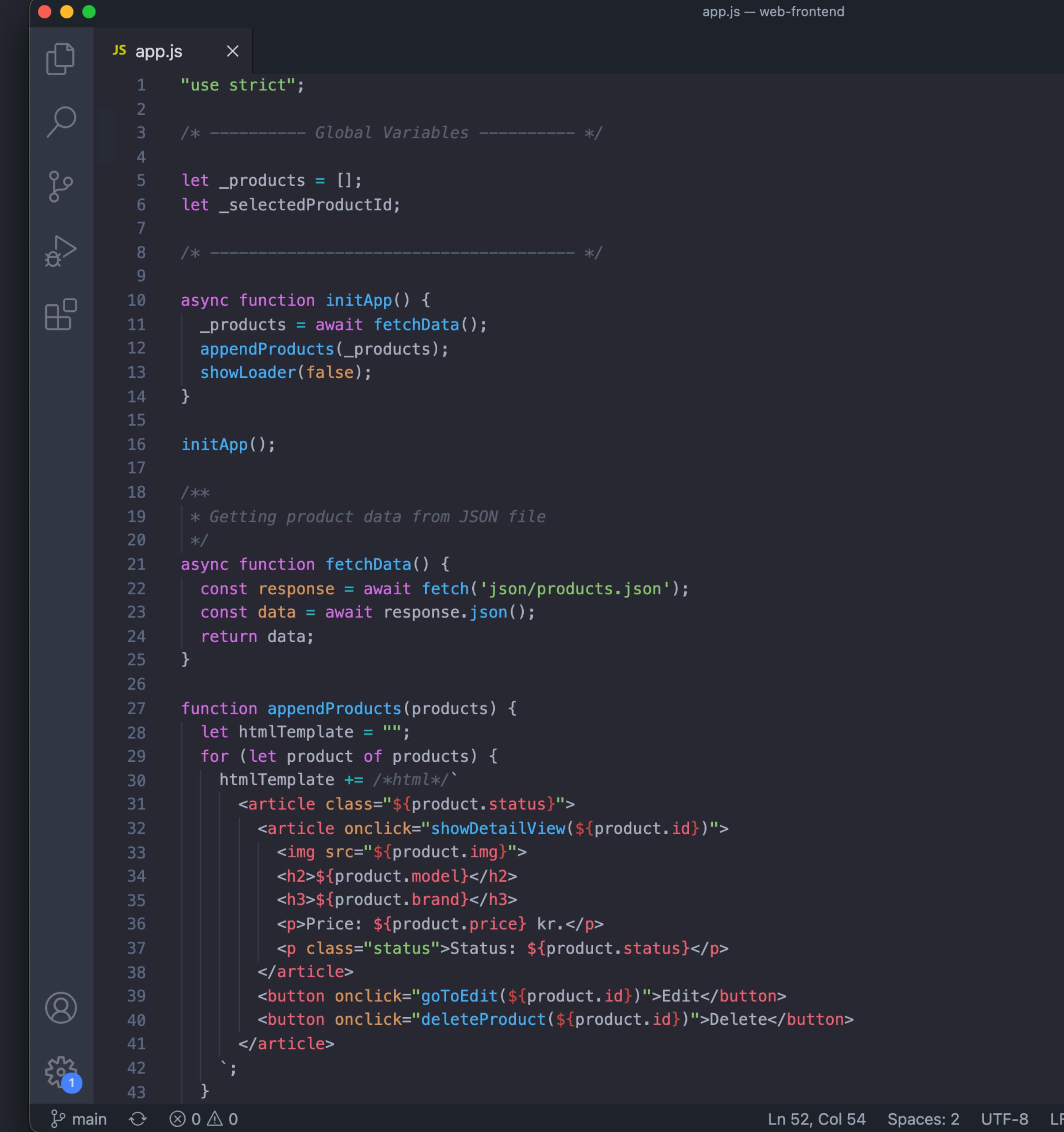
# FUNCTION-BASED JS

Functions, functions & functions

Your code and scripts are  
(mostly) structured using  
functions ... a bunch of functions

Action and event based.

Functions take input which is  
called arguments.

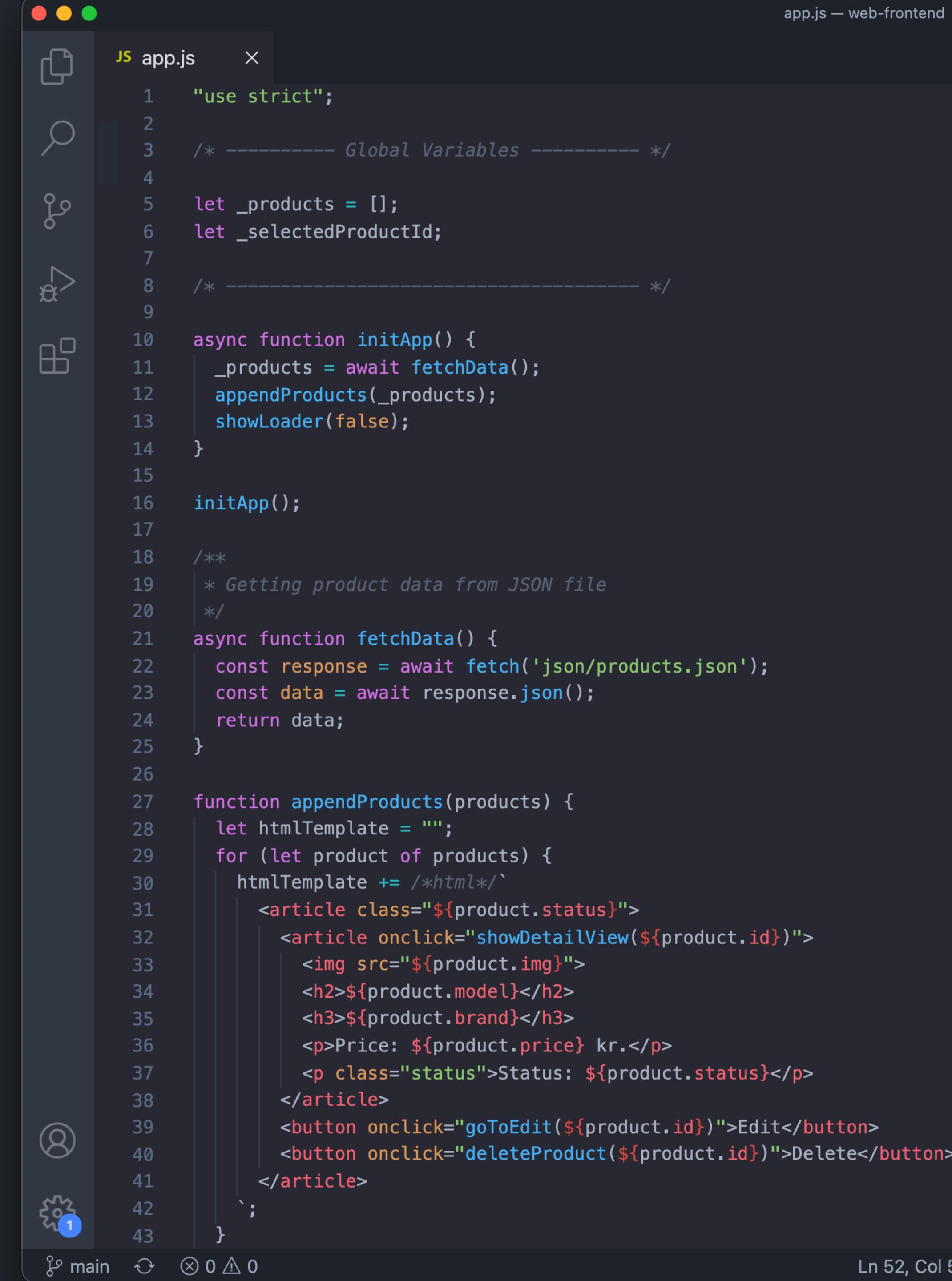


```
JS app.js ×
1 "use strict";
2
3 /* ----- Global Variables ----- */
4
5 let _products = [];
6 let _selectedProductId;
7
8 /*
9 */
10 async function initApp() {
11   _products = await fetchData();
12   appendProducts(_products);
13   showLoader(false);
14 }
15
16 initApp();
17
18 /**
19 * Getting product data from JSON file
20 */
21 async function fetchData() {
22   const response = await fetch('json/products.json');
23   const data = await response.json();
24   return data;
25 }
26
27 function appendProducts(products) {
28   let htmlTemplate = "";
29   for (let product of products) {
30     htmlTemplate += /*html*/
31       <article class="${product.status}">
32         <article onclick="showDetailView(${product.id})">
33           
34           <h2>${product.model}</h2>
35           <h3>${product.brand}</h3>
36           <p>Price: ${product.price} kr.</p>
37           <p class="status">Status: ${product.status}</p>
38         </article>
39         <button onclick="goToEdit(${product.id})">Edit</button>
40         <button onclick="deleteProduct(${product.id})">Delete</button>
41       </article>
42   ;
43 }
```

It's clean and straightforward.

Some would say it's easier.

It depends 🤷



The screenshot shows a dark-themed code editor window titled "JS app.js". The code is written in JavaScript and includes comments and some CSS-like styling within the HTML output. The code defines a global variable `\_products` and a function `initApp()` that fetches data from a JSON file and appends products to the DOM. It also defines a `fetchData()` function to get the data and an `appendProducts()` function to generate HTML based on the fetched data. The code uses template literals and conditional classes.

```
1  "use strict";
2
3  /* ----- Global Variables ----- */
4
5  let _products = [];
6  let _selectedProductId;
7
8  /*
9  */
10 async function initApp() {
11   _products = await fetchData();
12   appendProducts(_products);
13   showLoader(false);
14 }
15
16 initApp();
17
18 /**
19  * Getting product data from JSON file
20  */
21 async function fetchData() {
22   const response = await fetch('json/products.json');
23   const data = await response.json();
24   return data;
25 }
26
27 function appendProducts(products) {
28   let htmlTemplate = "";
29   for (let product of products) {
30     htmlTemplate += /*html*/
31       <article class="${product.status}">
32         <article onclick="showDetailView(${product.id})">
33           
34           <h2>${product.model}</h2>
35           <h3>${product.brand}</h3>
36           <p>Price: ${product.price} kr.</p>
37           <p class="status">Status: ${product.status}</p>
38         </article>
39         <button onclick="goUpEdit(${product.id})">Edit</button>
40         <button onclick="deleteProduct(${product.id})">Delete</button>
41       </article>
42   ;
43 }
```

# 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

# 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

```
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");
```

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)

# FUNCTION CALLED BY ANOTHER FUNCTION

Use the name of  
the function to  
call / execute

```
let users = [...  
];  
  
function appendUsers(users) {  
  let htmlTemplate = "";  
  for (const user of users) {  
    console.log(user);  
    htmlTemplate += /*html*/`  
      <article>  
          
        <h2>${user.name}</h2>  
        <a href="mailto:${user.email}">${user.email}</a>  
        <p>Role: ${user.enrollment_type}</p>  
      </article>  
    `;  
  }  
  document.querySelector("#users").innerHTML = htmlTemplate;  
}  
  
function initApp() {  
  appendUsers(users);  
}  
  
initApp();
```

users refers to  
our declared  
variable, users,  
(global variable)

# ARROW FUNCTIONS

## SHORTER & SIMPLIFIED FUNCTION SYNTAX

# ARROW FUNCTIONS

```
fetch('json/persons.json')
  .then(function (response) {
    return response.json();
  })
  .then(function (data) {
    appendPersons(data)
  });

```

FUNCTION DECLARATION

```
fetch('json/persons.json')
  .then(response => response.json())
  .then(data => appendPersons(data));

```

ARROW FUNCTION

```
function search(value) {  
    value = value.toLowerCase();  
    let filteredTeachers = [];  
    for (let teacher of teachers) {  
        let name = teacher.name.toLowerCase();  
        if (name.includes(value)) {  
            filteredTeachers.push(teacher);  
        }  
    }  
    appendTeachers(filteredTeachers);  
}
```

FOR OF LOOP

## .FILTER & ARROW FUNCTION

```
function search(value) {  
    let searchValue = value.toLowerCase();  
    let filteredTeachers = _teachers.filter(teacher => teacher.title.rendered.toLowerCase().includes(searchValue));  
    appendTeachers(filteredTeachers);  
}
```

```
function orderByBrand() {
  _products.sort((product1, product2) => {
    return product1.brand.localeCompare(product2.brand);
  });
  appendProducts(_products);
}

function orderByModel() {
  _products.sort((product1, product2) => {
    return product1.model.localeCompare(product2.model);
  });
  appendProducts(_products);
}

function orderByPrice() {
  _products.sort((product1, product2) => {
    return product1.price - product2.price;
  });
  appendProducts(_products);
}
```

. SORT & ARROW FUNCTIONS  
INSIDE FUNCTIONS DECLARATIONS

# ARROW FUNCTIONS

```
function orderByBrand() {  
  _products.sort(function (product1, product2) {  
    return product1.brand.localeCompare(product2.brand);  
  });  
  appendProducts(_products);  
}
```

FUNCTION DECLARATION

```
function orderByBrand() {  
  _products.sort((product1, product2) => {  
    return product1.brand.localeCompare(product2.brand);  
  });  
  appendProducts(_products);  
}
```

ARROW FUNCTION

```
function orderByBrand() {  
  _products.sort((product1, product2) => product1.brand.localeCompare(product2.brand));  
  appendProducts(_products);  
}
```

ONE LINE ARROW FUNCTION

# WHEN TO USE?

IT'S UP TO YOU



# CANVAS USERS SPA CASE

BACK

ALL USERS

Order by: Choose here ▾ Filter by: All Courses: All Search

Photo	Name	Email	Role	Course	Action
	Clara Juul Birk	<a href="#">eaaclb1@students.eaaa.dk</a>	Student	Course: MDU-E20FRONT1	<a href="#">UPDATE</a> <a href="#">DELETE</a>
	Mikkel Due Hørup Bjørnsholm	<a href="#">mikkel.13@hotmail.com</a>	Student	Course: MDU-E20FRONT1	<a href="#">UPDATE</a> <a href="#">DELETE</a>
	Morten Algy Bonderup	<a href="#">moab@eaaa.dk</a>	Teacher	Course: admin	<a href="#">UPDATE</a> <a href="#">DELETE</a>

USERS CREATE

Template: spa-canvas-users-template

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:  
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: "2019-08-02T02:26:25+02:00"}  
11: {id: "17478", name: "Lasse Bickmann", created\_at: "2021-08-05T00:58:06+02:00"}  
12: {id: "17479", name: "Thomas Hyllegaard Busk", created\_at: "2021-08-05T00:58:06+02:00"}  
13: {id: "17480", name: "Jesper Nissen Byg", created\_at: "2018-08-08T02:10:00+02:00"}  
14: {id: "17481", name: "Barbora Byrtusová", created\_at: "2019-08-07T02:34:00+02:00"}  
15: {id: "17482", name: "Rasmus Cederdorff", created\_at: "2018-09-06T02:23:57+02:00"}  
16: {id: "17483", name: "Lasse Aakjær", created\_at: "2019-08-03T02:26:25+02:00"}  
17: {id: "17484", name: "Nicklas Eibye Andersen", created\_at: "2019-08-07T00:00:00+02:00"}  
18: {id: "17485", name: "Piotr Andrzej Pospiech", created\_at: "2019-08-02T00:00:00+02:00"}  
19: {id: "17486", name: "Jeffrey David Serio", created\_at: "2017-05-10T14:09:27+02:00"}  
20: {id: "17487", name: "Sarah Øster Dybvad", created\_at: "2021-08-05T00:57:00+02:00"}  
21: {id: "17488", name: "Kristine Dzumakaijeva", created\_at: "2019-08-02T02:26:25+02:00"}  
22: {id: "17489", name: "Lasse Bickmann", created\_at: "2021-08-05T00:58:06+02:00"}  
23: {id: "17490", name: "Thomas Hyllegaard Busk", created\_at: "2021-08-05T00:58:06+02:00"}  
24: {id: "17491", name: "Jesper Nissen Byg", created\_at: "2018-08-08T02:10:00+02:00"}  
25: {id: "17492", name: "Barbora Byrtusová", created\_at: "2019-08-07T02:34:00+02:00"}  
26: {id: "17493", name: "Rasmus Cederdorff", created\_at: "2018-09-06T02:23:57+02:00"}  
27: {id: "17494", name: "Lasse Aakjær", created\_at: "2019-08-03T02:26:25+02:00"}  
28: {id: "17495", name: "Nicklas Eibye Andersen", created\_at: "2019-08-07T00:00:00+02:00"}  
29: {id: "17496", name: "Piotr Andrzej Pospiech", created\_at: "2019-08-02T00:00:00+02:00"}  
30: {id: "17497", name: "Jeffrey David Serio", created\_at: "2017-05-10T14:09:27+02:00"}  
31: {id: "17498", name: "Sarah Øster Dybvad", created\_at: "2021-08-05T00:57:00+02:00"}  
32: {id: "17499", name: "Kristine Dzumakaijeva", created\_at: "2019-08-02T02:26:25+02:00"}  
33: {id: "17500", name: "Lasse Bickmann", created\_at: "2021-08-05T00:58:06+02:00"}  
34: {id: "17501", name: "Thomas Hyllegaard Busk", created\_at: "2021-08-05T00:58:06+02:00"}  
35: {id: "17502", name: "Jesper Nissen Byg", created\_at: "2018-08-08T02:10:00+02:00"}  
36: {id: "17503", name: "Barbora Byrtusová", created\_at: "2019-08-07T02:34:00+02:00"}  
37: {id: "17504", name: "Rasmus Cederdorff", created\_at: "2018-09-06T02:23:57+02:00"}  
38: {id: "17505", name: "Lasse Aakjær", created\_at: "2019-08-03T02:26:25+02:00"}  
39: {id: "17506", name: "Nicklas Eibye Andersen", created\_at: "2019-08-07T00:00:00+02:00"}  
40: {id: "17507", name: "Piotr Andrzej Pospiech", created\_at: "2019-08-02T00:00:00+02:00"}  
41: {id: "17508", name: "Jeffrey David Serio", created\_at: "2017-05-10T14:09:27+02:00"}  
42: {id: "17509", name: "Sarah Øster Dybvad", created\_at: "2021-08-05T00:57:00+02:00"}  
43: {id: "17510", name: "Kristine Dzumakaijeva", created\_at: "2019-08-02T02:26:25+02:00"}  
44: {id: "17511", name: "Lasse Bickmann", created\_at: "2021-08-05T00:58:06+02:00"}  
45: {id: "17512", name: "Thomas Hyllegaard Busk", created\_at: "2021-08-05T00:58:06+02:00"}  
46: {id: "17513", name: "Jesper Nissen Byg", created\_at: "2018-08-08T02:10:00+02:00"}  
47: {id: "17514", name: "Barbora Byrtusová", created\_at: "2019-08-07T02:34:00+02:00"}  
48: {id: "17515", name: "Rasmus Cederdorff", created\_at: "2018-09-06T02:23:57+02:00"}  
49: {id: "17516", name: "Lasse Aakjær", created\_at: "2019-08-03T02:26:25+02:00"}  
50: {id: "17517", name: "Nicklas Eibye Andersen", created\_at: "2019-08-07T00:00:00+02:00"}  
51: {id: "17518", name: "Piotr Andrzej Pospiech", created\_at: "2019-08-02T00:00:00+02:00"}  
52: {id: "17519", name: "Jeffrey David Serio", created\_at: "2017-05-10T14:09:27+02:00"}  
53: {id: "17520", name: "Sarah Øster Dybvad", created\_at: "2021-08-05T00:57:00+02:00"}  
54: {id: "17521", name: "Kristine Dzumakaijeva", created\_at: "2019-08-02T02:26:25+02:00"}  
55: {id: "17522", name: "Lasse Bickmann", created\_at: "2021-08-05T00:58:06+02:00"}  
56: {id: "17523", name: "Thomas Hyllegaard Busk", created\_at: "2021-08-05T00:58:06+02:00"}  
57: {id: "17524", name: "Jesper Nissen Byg", created\_at: "2018-08-08T02:10:00+02:00"}  
58: {id: "17525", name: "Barbora Byrtusová", created\_at: "2019-08-07T02:34:00+02:00"}  
59: {id: "17526", name: "Rasmus Cederdorff", created\_at: "2018-09-06T02:23:57+02:00"}  
60: {id: "17527", name: "Lasse Aakjær", created\_at: "2019-08-03T02:26:25+02:00"}  
61: {id: "17528", name: "Nicklas Eibye Andersen", created\_at: "2019-08-07T00:00:00+02:00"}  
62: {id: "17529", name: "Piotr Andrzej Pospiech", created\_at: "2019-08-02T00:00:00+02:00"}  
63: {id: "17530", name: "Jeffrey David Serio", created\_at: "2017-05-10T14:09:27+02:00"}  
64: {id: "17531", name: "Sarah Øster Dybvad", created\_at: "2021-08-05T00:57:00+02:00"}  
65: {id: "17532", name: "Kristine Dzumakaijeva", created\_at: "2019-08-02T02:26:25+02:00"}  
66: {id: "17533", name: "Lasse Bickmann", created\_at: "2021-08-05T00:58:06+02:00"}  
67: {id: "17534", name: "Thomas Hyllegaard Busk", created\_at: "2021-08-05T00:58:06+02:00"}  
68: {id: "17535", name: "Jesper Nissen Byg", created\_at: "2018-08-08T02:10:00+02:00"}  
69: {id: "17536", name: "Barbora Byrtusová", created\_at: "2019-08-07T02:34:00+02:00"}  
70: {id: "17537", name: "Rasmus Cederdorff", created\_at: "2018-09-06T02:23:57+02:00"}  
71: {id: "17538", name: "Lasse Aakjær", created\_at: "2019-08-03T02:26:25+02:00"}  
72: {id: "17539", name: "Nicklas Eibye Andersen", created\_at: "2019-08-07T00:00:00+02:00"}  
73: {id: "17540", name: "Piotr Andrzej Pospiech", created\_at: "2019-08-02T00:00:00+02:00"}  
74: {id: "17541", name: "Jeffrey David Serio", created\_at: "2017-05-10T14:09:27+02:00"}  
75: {id: "17542", name: "Sarah Øster Dybvad", created\_at: "2021-08-05T00:57:00+02:00"}  
76: {id: "17543", name: "Kristine Dzumakaijeva", created\_at: "2019-08-02T02:26:25+02:00"}  
77: {id: "17544", name: "Lasse Bickmann", created\_at: "2021-08-05T00:58:06+02:00"}  
78: {id: "17545", name: "Thomas Hyllegaard Busk", created\_at: "2021-08-05T00:58:06+02:00"}  
79: {id: "17546", name: "Jesper Nissen Byg", created\_at: "2018-08-08T02:10:00+02:00"}  
80: {id: "17547", name: "Barbora Byrtusová", created\_at: "2019-08-07T02:34:00+02:00"}  
81: {id: "17548", name: "Rasmus Cederdorff", created\_at: "2018-09-06T02:23:57+02:00"}  
82: {id: "17549", name: "Lasse Aakjær", created\_at: "2019-08-03T02:26:25+02:00"}  
83: {id: "17550", name: "Nicklas Eibye Andersen", created\_at: "2019-08-07T00:00:00+02:00"}  
84: {id: "17551", name: "Piotr Andrzej Pospiech", created\_at: "2019-08-02T00:00:00+02:00"}  
85: {id: "17552", name: "Jeffrey David Serio", created\_at: "2017-05-10T14:09:27+02:00"}  
86: {id: "17553", name: "Sarah Øster Dybvad", created\_at: "2021-08-05T00:57:00+02:00"}  
87: {id: "17554", name: "Kristine Dzumakaijeva", created\_at: "2019-08-02T02:26:25+02:00"}  
88: {id: "17555", name: "Lasse Bickmann", created\_at: "2021-08-05T00:58:06+02:00"}  
89: {id: "17556", name: "Thomas Hyllegaard Busk", created\_at: "2021-08-05T00:58:06+02:00"}  
90: {id: "17557", name: "Jesper Nissen Byg", created\_at: "2018-08-08T02:10:00+02:00"}  
91: {id: "17558", name: "Barbora Byrtusová", created\_at: "2019-08-07T02:34:00+02:00"}  
92: {id: "17559", name: "Rasmus Cederdorff", created\_at: "2018-09-06T02:23:57+02:00"}  
93: {id: "17560", name: "Lasse Aakjær", created\_at: "2019-08-03T02:26:25+02:00"}  
94: {id: "17561", name: "Nicklas Eibye Andersen", created\_at: "2019-08-07T00:00:00+02:00"}  
95: {id: "17562", name: "Piotr Andrzej Pospiech", created\_at: "2019-08-02T00:00:00+02:00"}  
96: {id: "17563", name: "Jeffrey David Serio", created\_at: "2017-05-10T14:09:27+02:00"}  
97: {id: "17564", name: "Sarah Øster Dybvad", created\_at: "2021-08-05T00:57:00+02:00"}  
98: {id: "17565", name: "Kristine Dzumakaijeva", created\_at: "2019-08-02T02:26:25+02:00"}  
99: {id: "17566", name: "Lasse Bickmann", created\_at: "2021-08-05T00:58:06+02:00"}  
100: {id: "17567", name: "Thomas Hyllegaard Busk", created\_at: "2021-08-05T00:58:06+02:00"}  
101: {id: "17568", name: "Jesper Nissen Byg", created\_at: "2018-08-08T02:10:00+02:00"}  
102: {id: "17569", name: "Barbora Byrtusová", created\_at: "2019-08-07T02:34:00+02:00"}  
103: {id: "17570", name: "Rasmus Cederdorff", created\_at: "2018-09-06T02:23:57+02:00"}  
104: {id: "17571", name: "Lasse Aakjær", created\_at: "2019-08-03T02:26:25+02:00"}  
105: {id: "17572", name: "Nicklas Eibye Andersen", created\_at: "2019-08-07T00:00:00+02:00"}  
106: {id: "17573", name: "Piotr Andrzej Pospiech", created\_at: "2019-08-02T00:00:00+02:00"}  
107: {id: "17574", name: "Jeffrey David Serio", created\_at: "2017-05-10T14:09:27+02:00"}  
108: {id: "17575", name: "Sarah Øster Dybvad", created\_at: "2021-08-05T00:57:00+02:00"}  
109: {id: "17576", name: "Kristine Dzumakaijeva", created\_at: "2019-08-02T02:26:25+02:00"}  
110: {id: "17577", name: "Lasse Bickmann", created\_at: "2021-08-05T00:58:06+02:00"}  
111: {id: "17578", name: "Thomas Hyllegaard Busk", created\_at: "2021-08-05T00:58:06+02:00"}  
112: {id: "17579", name: "Jesper Nissen Byg", created\_at: "2018-08-08T02:10:00+02:00"}  
113: {id: "17580", name: "Barbora Byrtusová", created\_at: "2019-08-07T02:34:00+02:00"}  
114: {id: "17581", name: "Rasmus Cederdorff", created\_at: "2018-09-06T02:23:57+0

SINGLE PAGE WEB APP TEMP | Products | +

127.0.0.1:5501/spa-products-fetch-json-enhanced/index.html

# PRODUCTS

Order by: Choose here ▾

Show out of stock

Search



**MacBook Pro 13"**

Apple

Price: 11799 kr.

Status: outOfStock

[EDIT](#) [DELETE](#)



**MacBook Pro 15"**

Apple

Price: 21499 kr.

Status: inStock

[EDIT](#) [DELETE](#)



**Zenbook 14"**

ASUS

Price: 8099 kr.

Status: outOfStock

[EDIT](#) [DELETE](#)



**Unknown laptop**

ASUS

Price: 8099 kr.

Status: inStock

[EDIT](#) [DELETE](#)



**ASUS ZenBook**

ASUS

Price: 8099 kr.

Status: inStock

[EDIT](#) [DELETE](#)



**Unknown tablet**

ASUS

Price: 8099 kr.

Status: inStock

[EDIT](#) [DELETE](#)

PRODUCTS ADD PRODUCT

# CANVAS USER SPA #1

BACK

Template: spa-canvas-users-template

1. Create a single page application based on user data fetched from: <https://cederdorff.github.io/web-frontend/canvas-users/data.json>
2. Append and display all users in a view in your SPA. Use a grid view or flex box to display all users, and make sure it adjusts to the width of the screen.
3. Display at least the following for every user: name, avatar\_url, email & enrollment\_type
4. Implement search functionality searching on name and/or other properties. You might have to use a for of or .filter
5. Implement a create view and implement the create functionally using array.push(...). Remember generating a dummy id when adding a new user.

# CANVAS USER SPA #2

BACK

Template: spa-canvas-users-template

1. Implement sort functionality in order to sort by `name` (first name) and `sortableName` (last name). Use a select-option (dropdown).
2. Make it possible to filter users on `enrollment` type. You could use select and option, checkbox, and/or create different pages.
3. Create and implement an update view using the `id` of the user to `.find` and update properties of the user in the users array.
4. Implement delete functionality using the `id` to filter the users array. You must display a delete confirm dialog before deleting.
5. Create a detail view displaying all properties of the selected user.
6. Implement spinner/ loader to tell the user every time the script is loading or executing functions. We must give the user some feedback.
7. Reimplement the styling. Create a full page layout with grid and/or flexbox. Please, style it different than a RACE.js SPA.

# VARIABLE

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



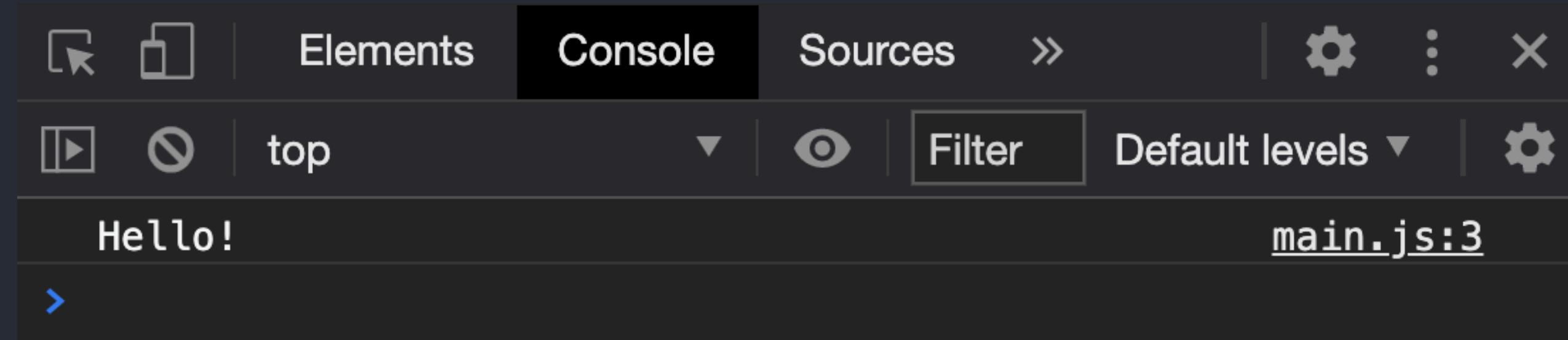
The variable `message` can be imagined as a box labeled "message" with the value "Hello!" in it. We can put any value in the box.

# VARIABLES

... STORE DATA IN THE MEMORY

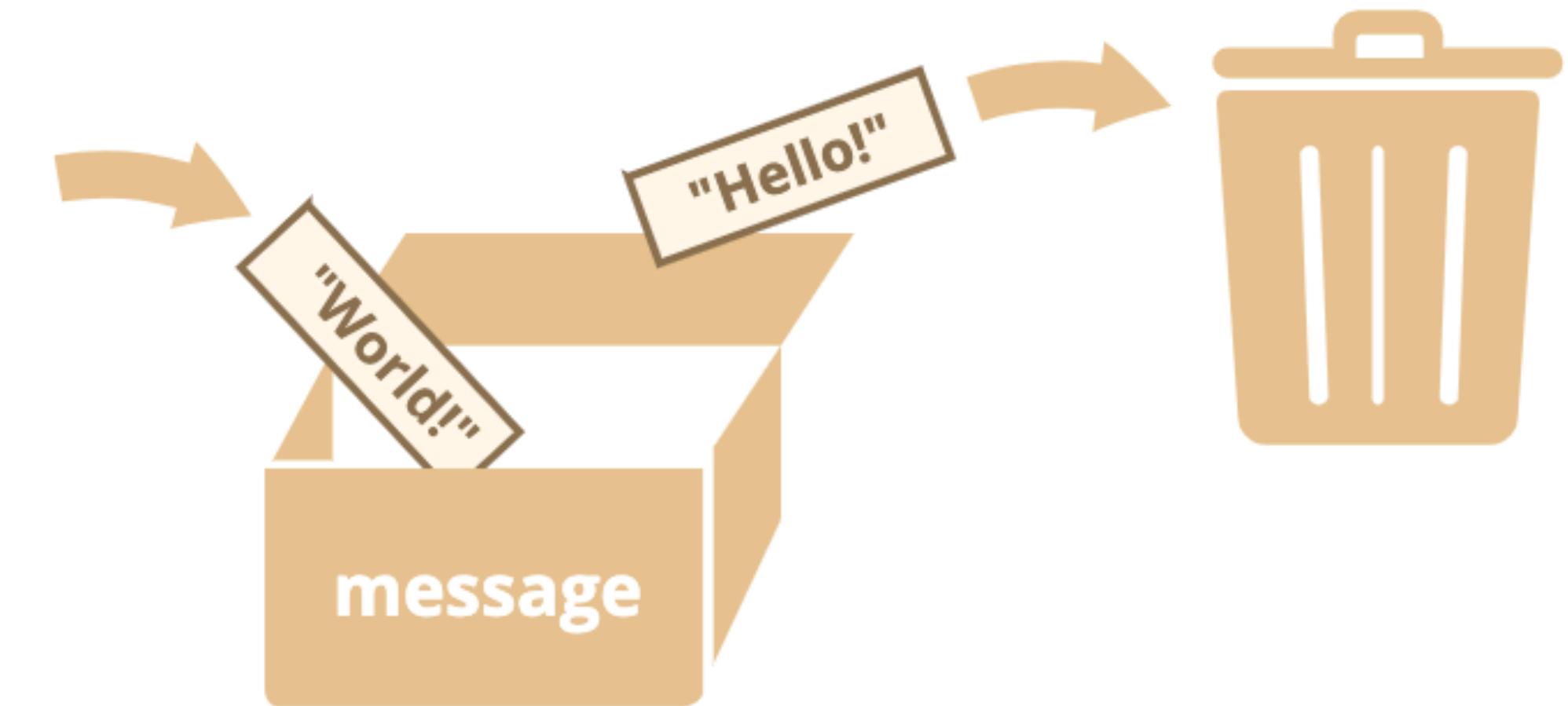
```
let message = "Hello!";
```

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



# VARIABLE

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



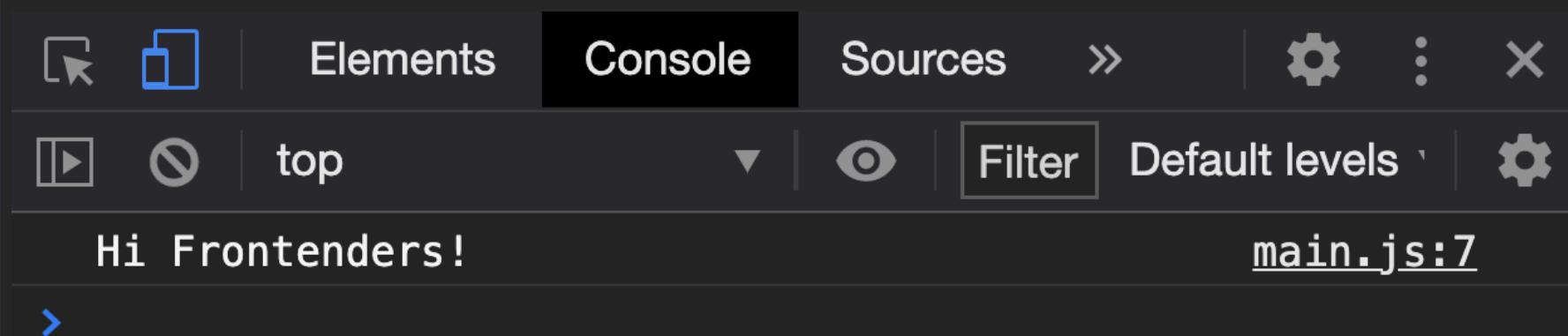
We can change the value of the variables as many times as we want.

```
// 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/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-vs-let

```
// "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`.

# DATA TYPES

In variables we store values. Values are of a certain data type.

In JavaScript we have 8 basic data types.

`number` for numbers of any kind: integer or floating-point, integers are limited by  $\pm(2^{53}-1)$ .

`bigint` is for integer numbers of arbitrary length.

`string` for strings. A string may have zero or more characters, there's no separate single-character type.

`boolean` for `true`/`false`.

`null` for unknown values – a standalone type that has a single value `null`.

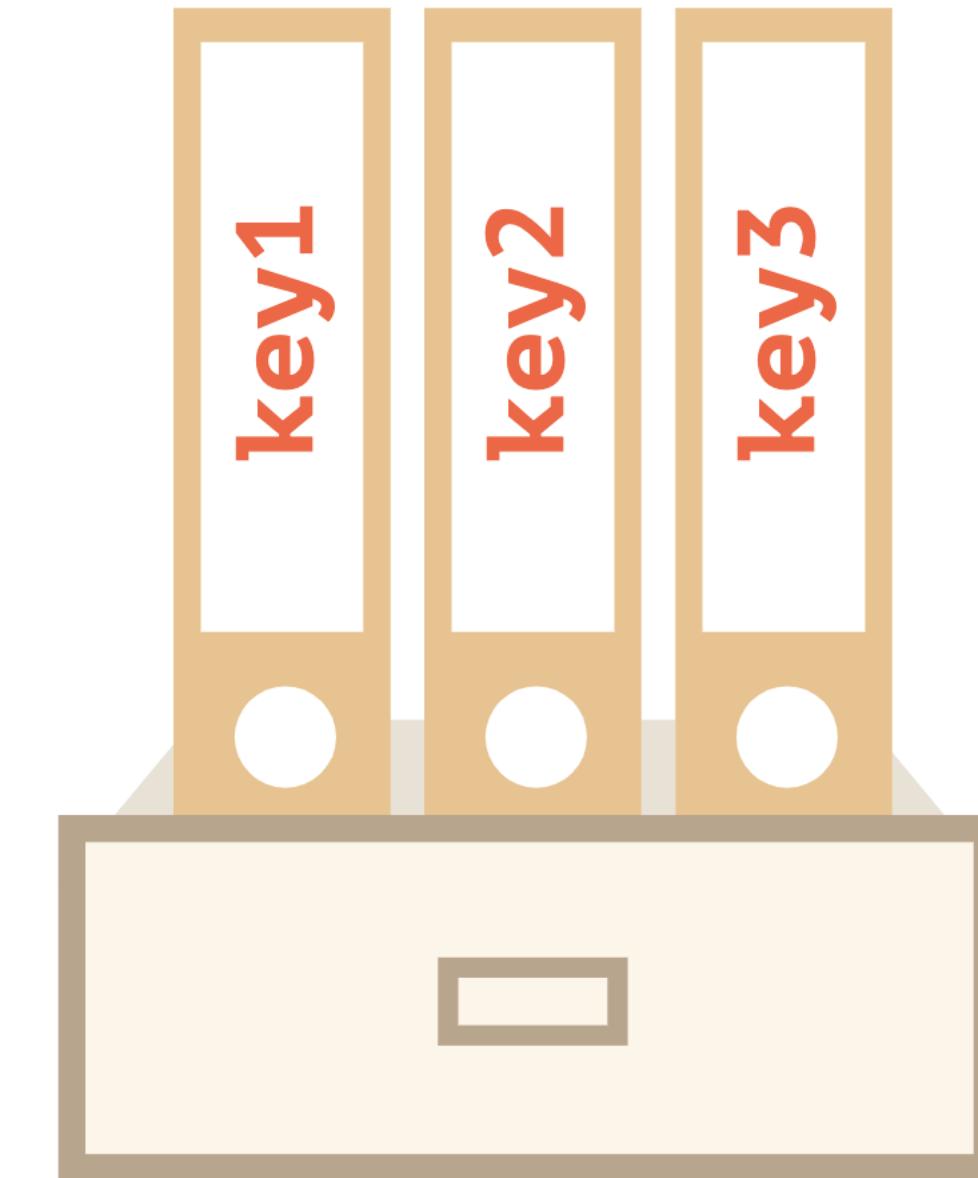
`undefined` for unassigned values – a standalone type that has a single value `undefined`.

`object` for more complex data structures.

`symbol` for unique identifiers.

# 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

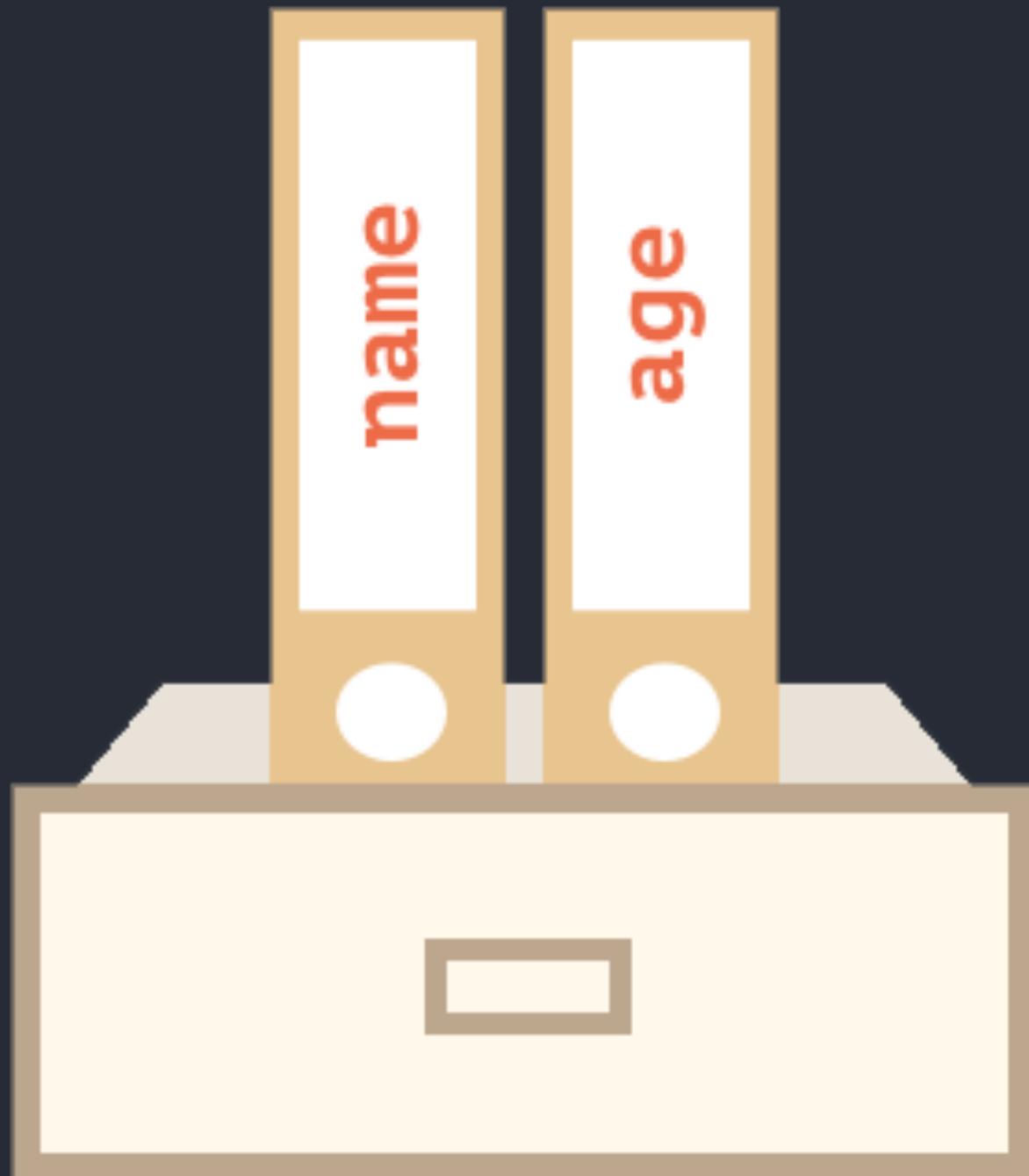
# OBJECTS

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.

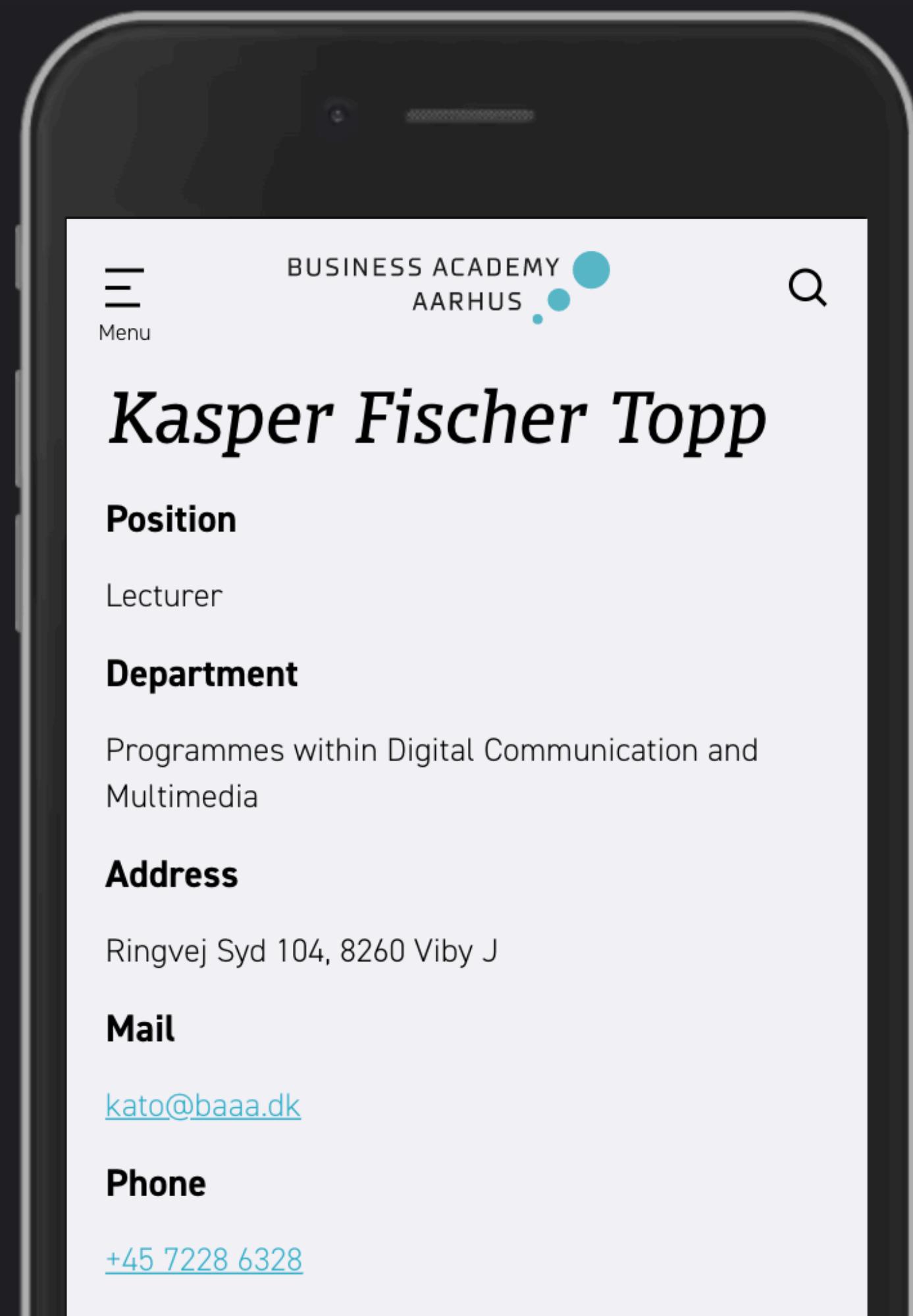
main.js:11

# OBJECTS

# A SET OF NAMED VALUES

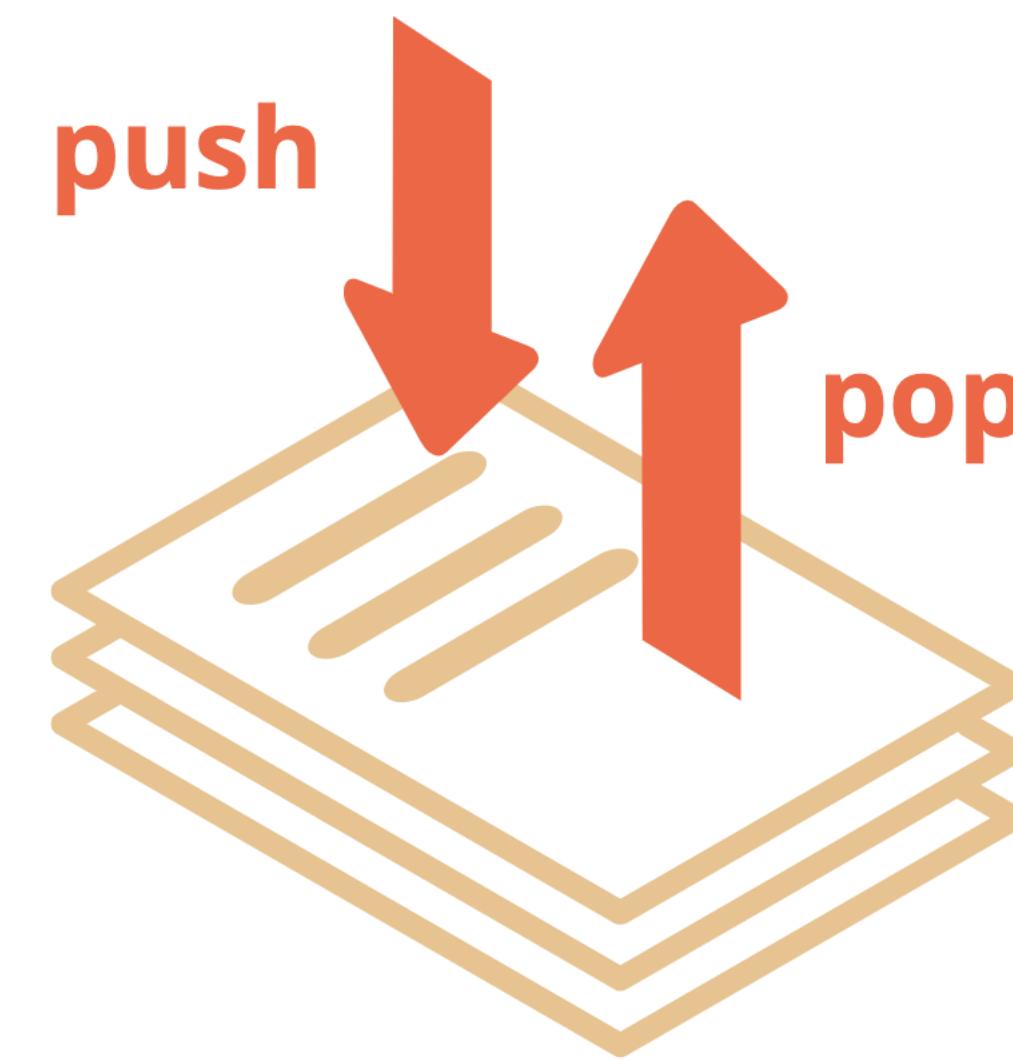
property value

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



# ARRAYS

Arrays are ordered collections



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

Course roster: WU-E21s - 1. se

aaaa.instructure.com/courses/13351/users

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 Ac

Name	Login ID	SIS ID	Section	Role	Last Activity	To Ac
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

property value

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

[{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: "2019-08-07T01:22:27+02:00"}  
2: {id: "17476", name: "Piotr Andrzej Pospiech", created\_at: "2019-08-02T01:22:27+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: "2018-08-08T02:10:09+02:00"}  
6: {id: "17960", name: "Barbora Byrtusová", created\_at: "2019-08-07T02:34:09+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:  
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: "2021-08-05T00:57:09+02:00"}  
10: {id: "17477", name: "Kristine Dzumakaijeva", created\_at: "2019-08-02T02:26:25+02:00"}  
11: {id: "17478", name: "Ksenia Dzumakaijeva", created\_at: "2019-08-02T02:26:25+02:00"}  
12: {id: "17479", name: "Dmitry Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
13: {id: "17480", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
14: {id: "17481", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
15: {id: "17482", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
16: {id: "17483", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
17: {id: "17484", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
18: {id: "17485", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
19: {id: "17486", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
20: {id: "17487", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
21: {id: "17488", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
22: {id: "17489", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
23: {id: "17490", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
24: {id: "17491", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
25: {id: "17492", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
26: {id: "17493", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
27: {id: "17494", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
28: {id: "17495", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
29: {id: "17496", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
30: {id: "17497", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
31: {id: "17498", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
32: {id: "17499", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
33: {id: "17500", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
34: {id: "17501", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
35: {id: "17502", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
36: {id: "17503", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
37: {id: "17504", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
38: {id: "17505", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
39: {id: "17506", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
40: {id: "17507", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
41: {id: "17508", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
42: {id: "17509", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
43: {id: "17510", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
44: {id: "17511", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
45: {id: "17512", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
46: {id: "17513", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
47: {id: "17514", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
48: {id: "17515", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
49: {id: "17516", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
50: {id: "17517", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
51: {id: "17518", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
52: {id: "17519", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
53: {id: "17520", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
54: {id: "17521", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
55: {id: "17522", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
56: {id: "17523", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
57: {id: "17524", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
58: {id: "17525", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
59: {id: "17526", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
60: {id: "17527", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
61: {id: "17528", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
62: {id: "17529", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
63: {id: "17530", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
64: {id: "17531", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
65: {id: "17532", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
66: {id: "17533", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
67: {id: "17534", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
68: {id: "17535", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
69: {id: "17536", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
70: {id: "17537", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
71: {id: "17538", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
72: {id: "17539", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
73: {id: "17540", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
74: {id: "17541", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
75: {id: "17542", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
76: {id: "17543", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
77: {id: "17544", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
78: {id: "17545", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
79: {id: "17546", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
80: {id: "17547", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
81: {id: "17548", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
82: {id: "17549", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
83: {id: "17550", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
84: {id: "17551", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
85: {id: "17552", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
86: {id: "17553", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
87: {id: "17554", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
88: {id: "17555", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
89: {id: "17556", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
90: {id: "17557", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
91: {id: "17558", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
92: {id: "17559", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
93: {id: "17560", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
94: {id: "17561", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
95: {id: "17562", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
96: {id: "17563", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
97: {id: "17564", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
98: {id: "17565", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
99: {id: "17566", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
100: {id: "17567", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
101: {id: "17568", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
102: {id: "17569", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
103: {id: "17570", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
104: {id: "17571", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
105: {id: "17572", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
106: {id: "17573", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
107: {id: "17574", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26:25+02:00"}  
108: {id: "17575", name: "Dmitriy Dzumakaijev", created\_at: "2019-08-02T02:26

# DATA TYPES (& STRUCTURES)

## OBJECTS & ARRAYS

# FOR OF LOOP

ITERATE OVER ARRAYS OR OTHER ITERABLE  
OBJECTS

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

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

# LOOPS

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

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

[https://www.w3schools.com/js/js\\_loop\\_for.asp](https://www.w3schools.com/js/js_loop_for.asp)

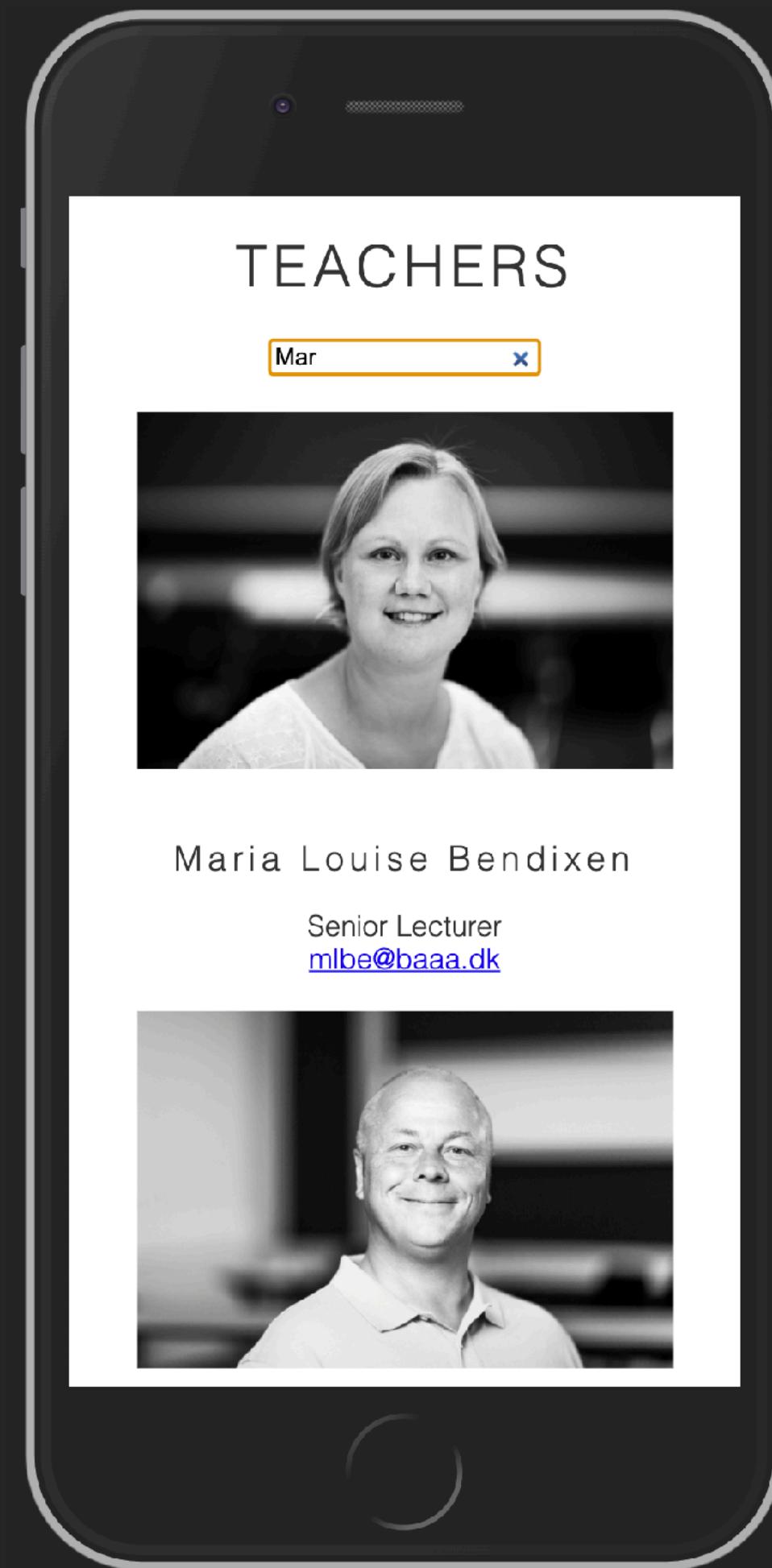
<https://javascript.info/array#loops>

<https://javascript.info/while-for>

# LOOPS

... SEARCH

```
function search(value) {  
    console.log(value);  
    let filteredTeachers = [];  
    for (let teacher of teachers) {  
        let name = teacher.name.toLowerCase();  
        if (name.includes(value.toLowerCase())) {  
            filteredTeachers.push(teacher);  
        }  
    }  
    console.log(filteredTeachers);  
    appendTeachers(filteredTeachers);  
}
```



array-teachers-search

# ARRAYS

## .FILTER(...)

```
searchValue = searchValue.toLowerCase();
let filteredfamilyMembers = familyMembers.filter(function(familyMember) {
  let name = familyMember.name.toLowerCase();
  return name.includes(searchValue);
});

console.log(filteredfamilyMembers);
```

# 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

# ARRAY METHODS

.PUSH()

.FIND()

.MAP()

.REDUCE()

.FILTER()

.SORT()

.CONCAT()

...

Computer science student



Senior developer, 10+ years experience



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

100 seconds of

JS

# ARRAY MAP



# ARRAY.MAP(...)

...iterate over an array and modify each element.

Array.map(...) calls a callback function for each element in the array.

```
const persons = [
  { firstname: "Birgitte", lastname: "Iversen" },
  { firstname: "Lykke", lastname: "Dahlen" },
  { firstname: "Rasmus", lastname: "Cederdorff" }
];

const mapped = persons.map(person => {
  return {
    name: `${person.firstname} ${person.lastname}`
  };
});

console.log(mapped);
```

▼ (3) [{...}, {...}, {...}] *i*

► 0: {name: 'Birgitte Iversen'}

► 1: {name: 'Lykke Dahlen'}

► 2: {name: 'Rasmus Cederdorff'}

length: 3

■ ■ ■ ■	.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.MAP(...)

BACK

- Project template: array-map
- Use .map to map over the persons array and concatenate firstName and lastName to a new property called name.
- console.log() the result
- Add a property called birthYear for each person object. Use map to map over the array and add the property birthYear dynamically based on birthDate (hint: use String.split(...) or .slice(...)).

# `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`

... 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>;
  }
}
```

WEB APP, SPA &  
ROUTING

# WEB APP

.. IS ANY PROGRAM EXECUTED IN THE BROWSER

# WEB APP

... IS ANY PROGRAM THAT PERFORMS A SPECIFIC  
FUNCTION BY USING A WEB BROWSER AS ITS CLIENT

WEB AS ACTION

THE USER WANTS TO COMPLETE A TASK

NOT JUST ABOUT MOBILE DEVICES

# WEBSITE

THE USER WANTS TO GET SOMETHING

# WEB APP

THE USER WANTS TO DO SOMETHING

# WEB APP

- Stored on a web server
- Launched by the browser
- Accessible anywhere, anytime - on any device through the browser
- JavaScript, HTML & CSS
- Single Page Application
- Fetches data from (different) data sources.



ROUTER & SPA

```
✓ vanilla-js-spa-router
  > css
  > img
  ✓ js
    JS app.js
    JS router.js
  <> index.html

✓ vanilla-js-spa-router-hash
  > css
  > img
  ✓ js
    JS app.js
    JS router.js
  <> index.html
```

```
> spa-user-crud-jsonbin
> spa-user-crud-jsonbin-module-based
> spa-user-crud-jsonbin-module-pages
> spa-user-crud-jsonbin-template
```

# SPA & ROUTER PROJECTS

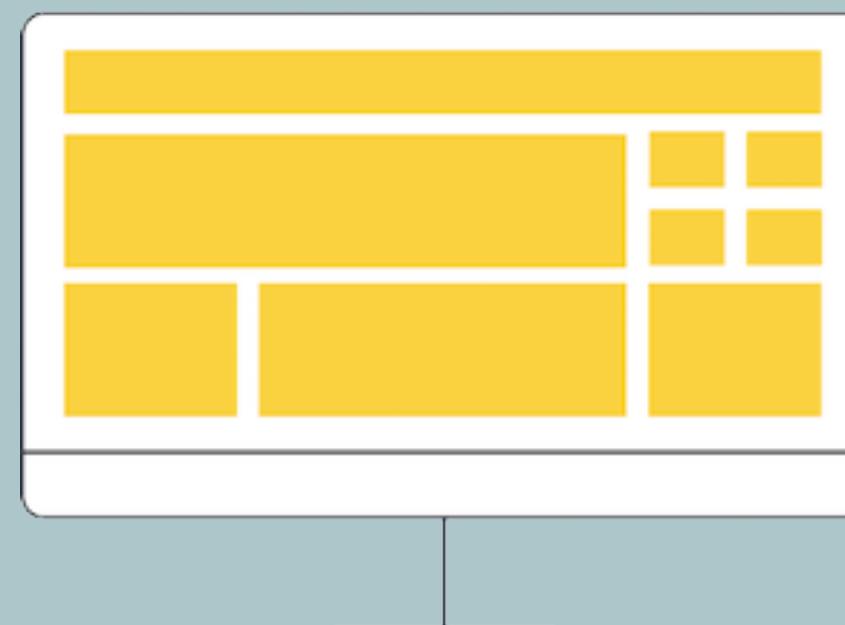
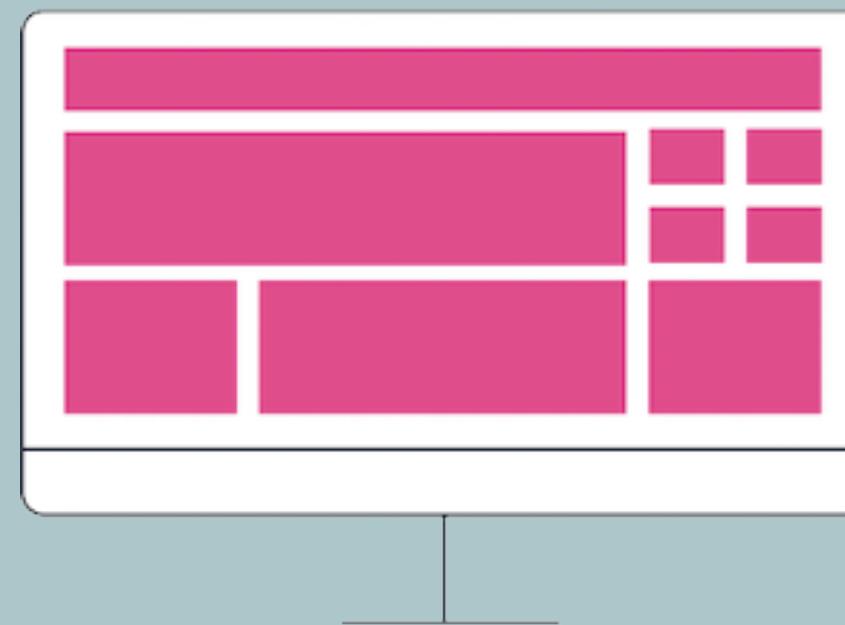
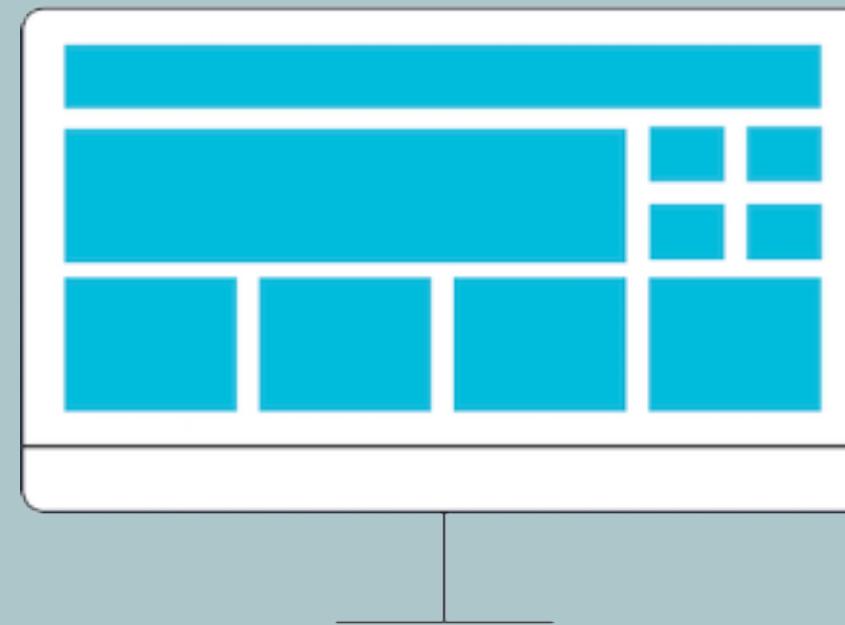
# SINGLE PAGE APPLICATION

*“A single-page application is an application that loads a single HTML page and all the necessary assets (such as JavaScript and CSS) required for the application to run. Any interactions with the page or subsequent pages do not require a round trip to the server which means the page is not reloaded.”*

<https://reactjs.org/docs/glossary.html#single-page-application>

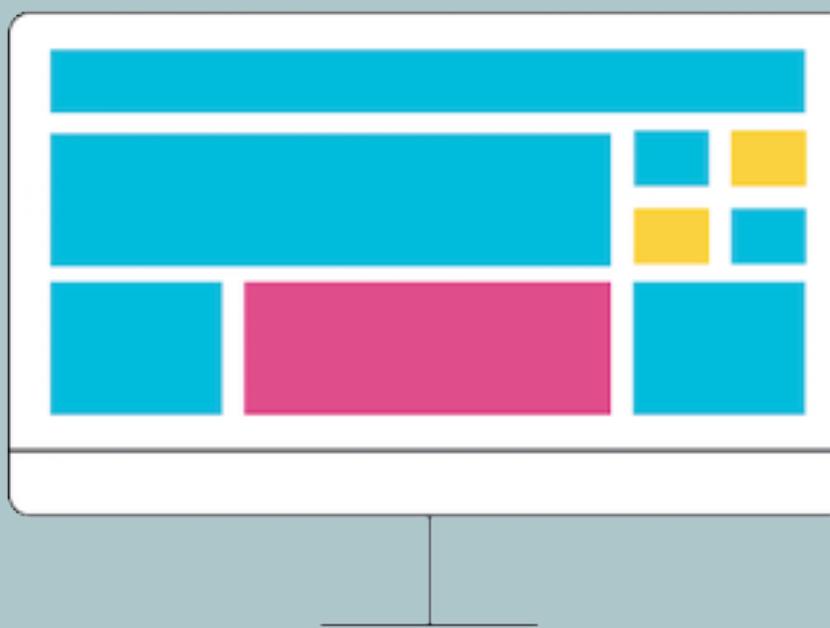
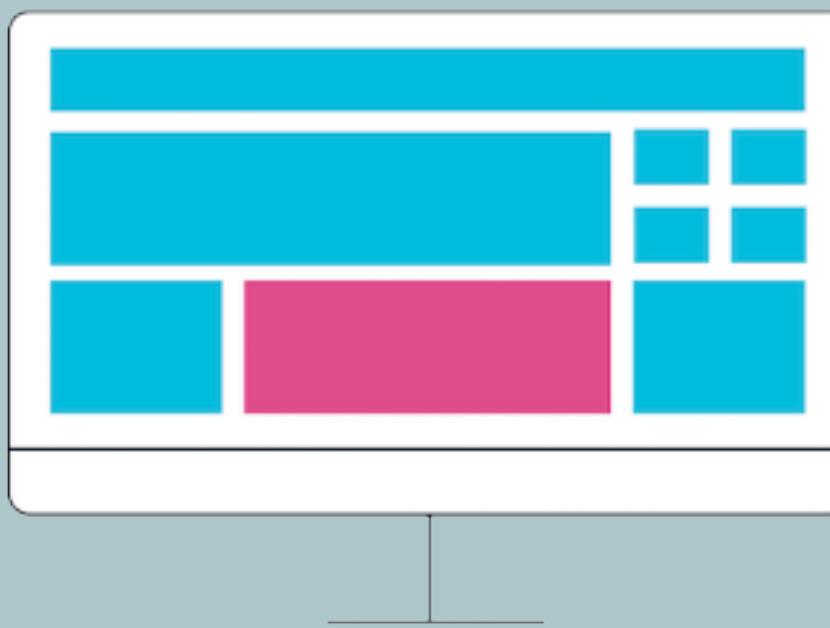
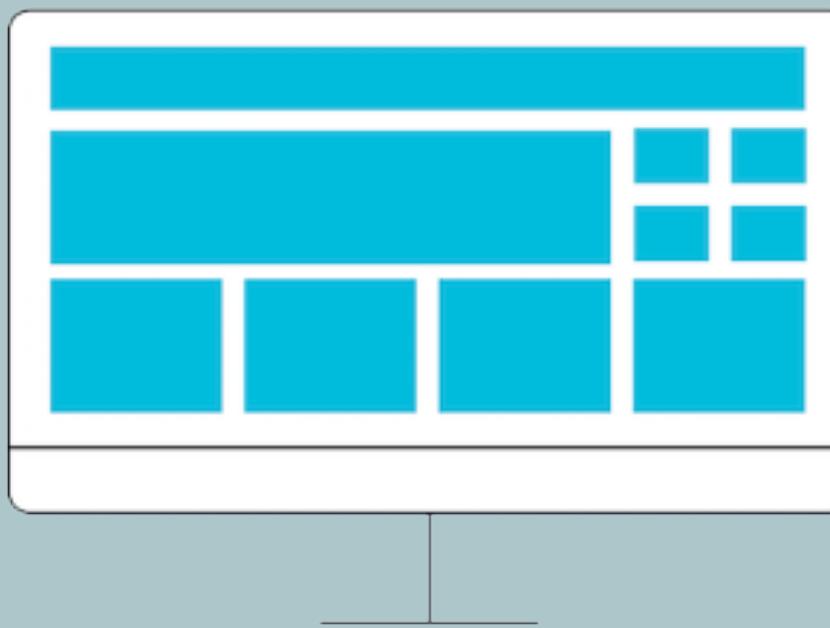
# TRADITIONAL

- Every request for new information gives you a new version of the whole page.
- We have to load it all over again every time we want to change data and get new data.
- The browser has to render the page again.

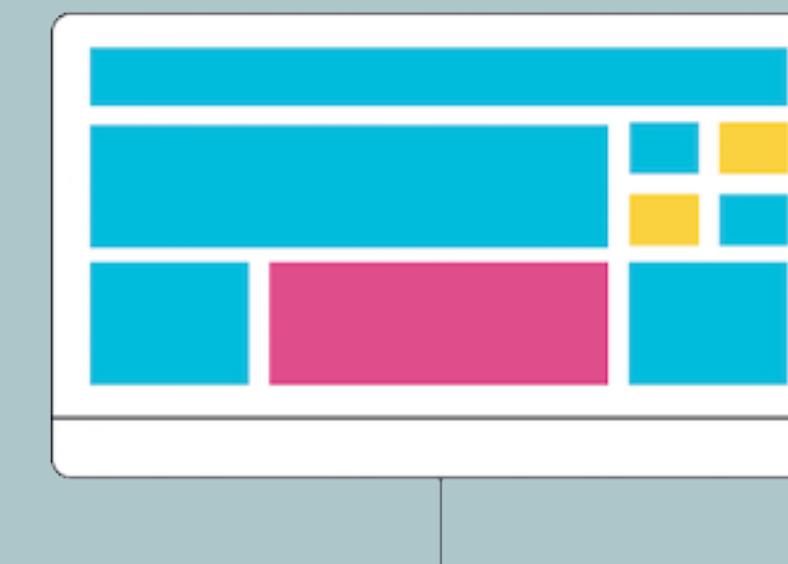
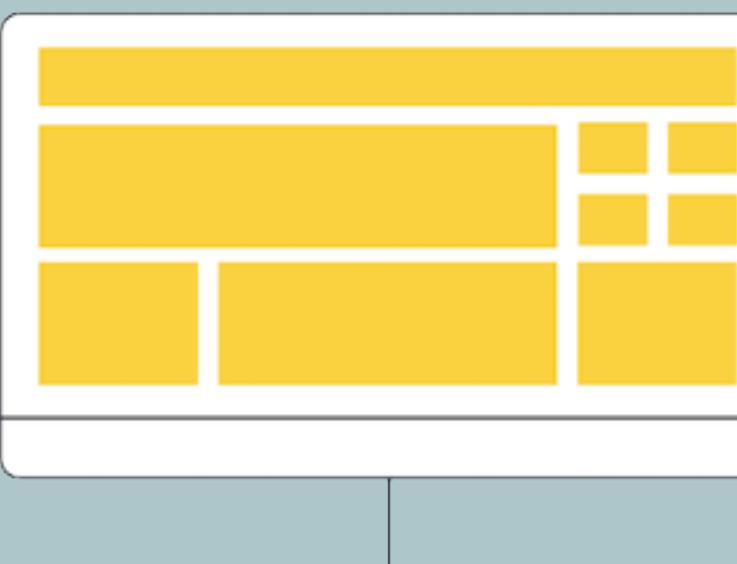
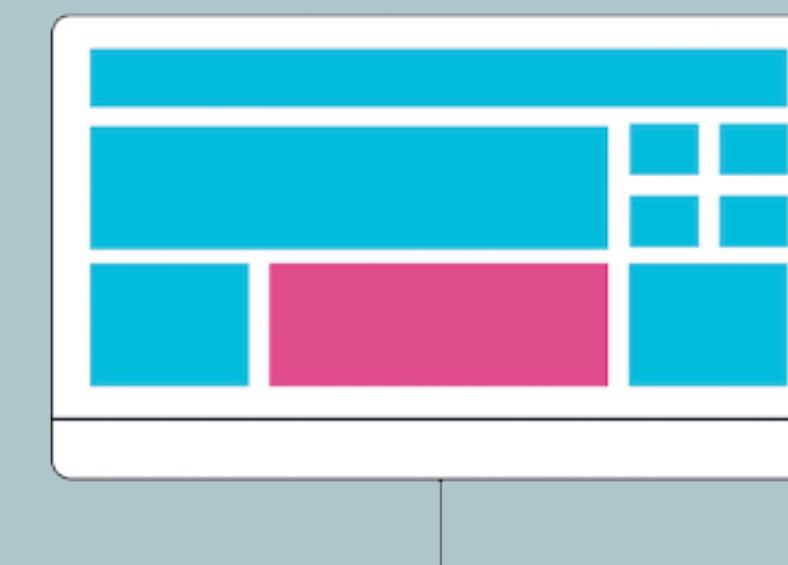
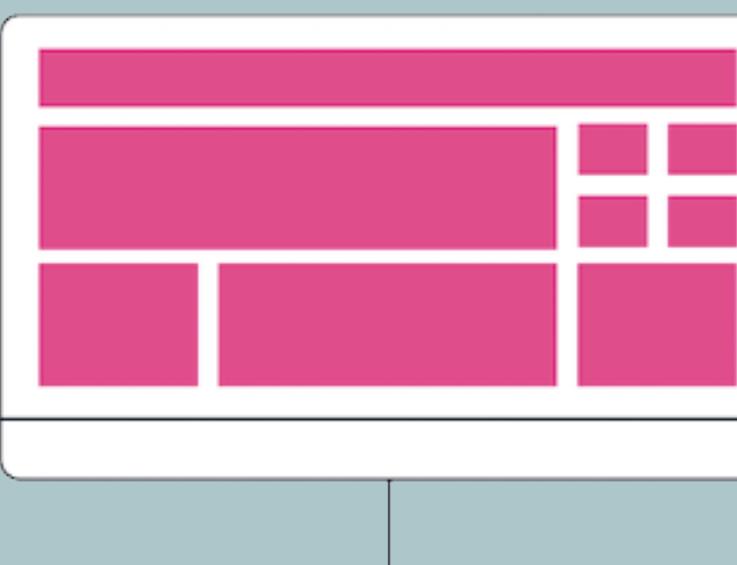
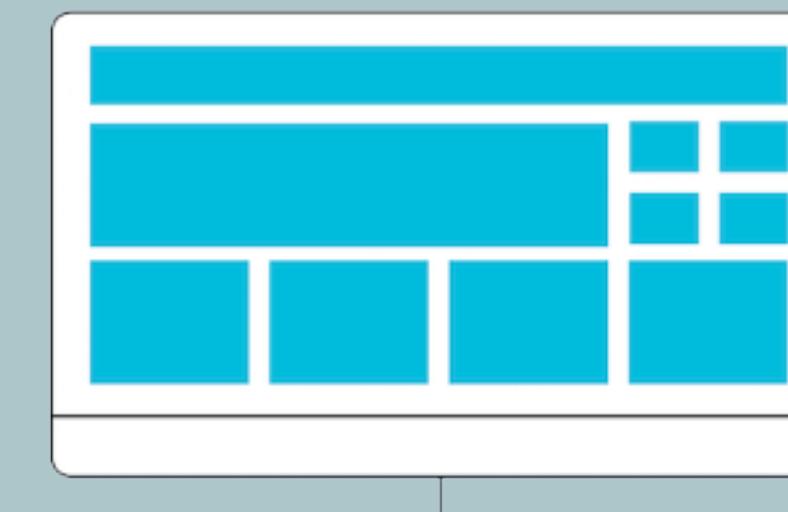
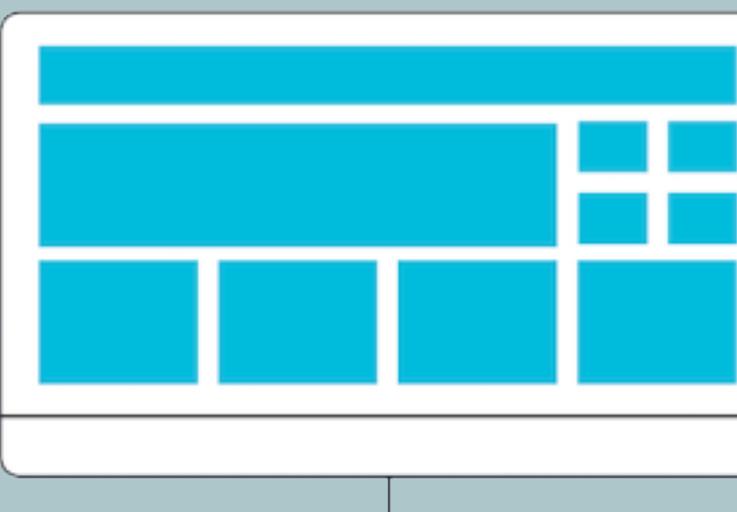


# SPA

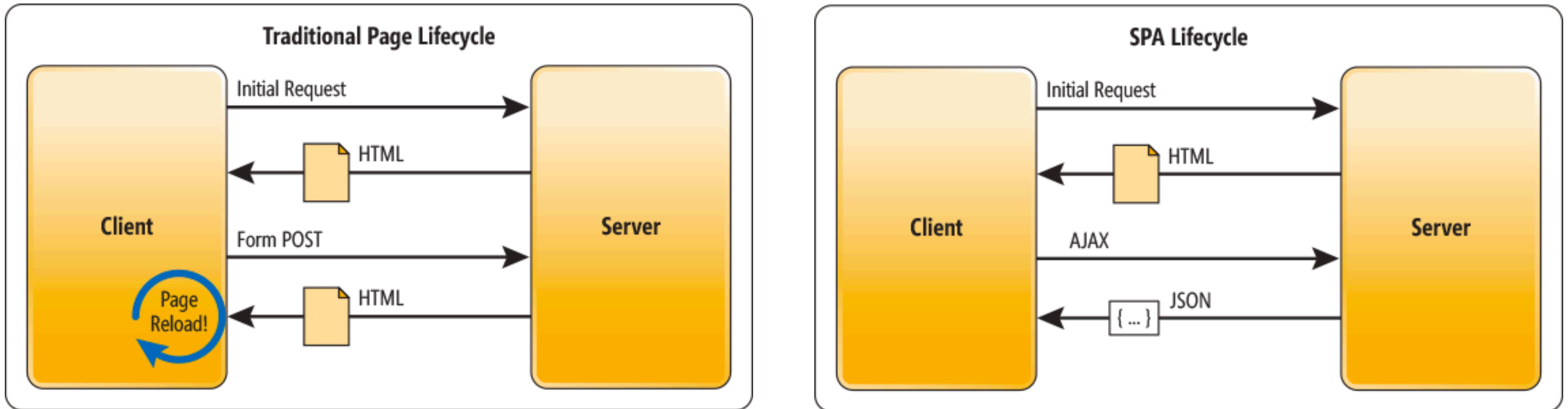
- We only request and load the data we want.
- No page reload/ refresh.
- Client side & JavaScript.
- Less server work required.
- Advanced UI & Animations on page change.
- Better performance.



# M P A   V S   S P A



# TRADITIONAL VS SPA LIFECYCLE



# ROUTER

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

It's a key component in Single Page Applications.

[https://medium.com/@fro\\_g/routing-in-javascript-d552ff4d2921](https://medium.com/@fro_g/routing-in-javascript-d552ff4d2921)

# SPA & ROUTER

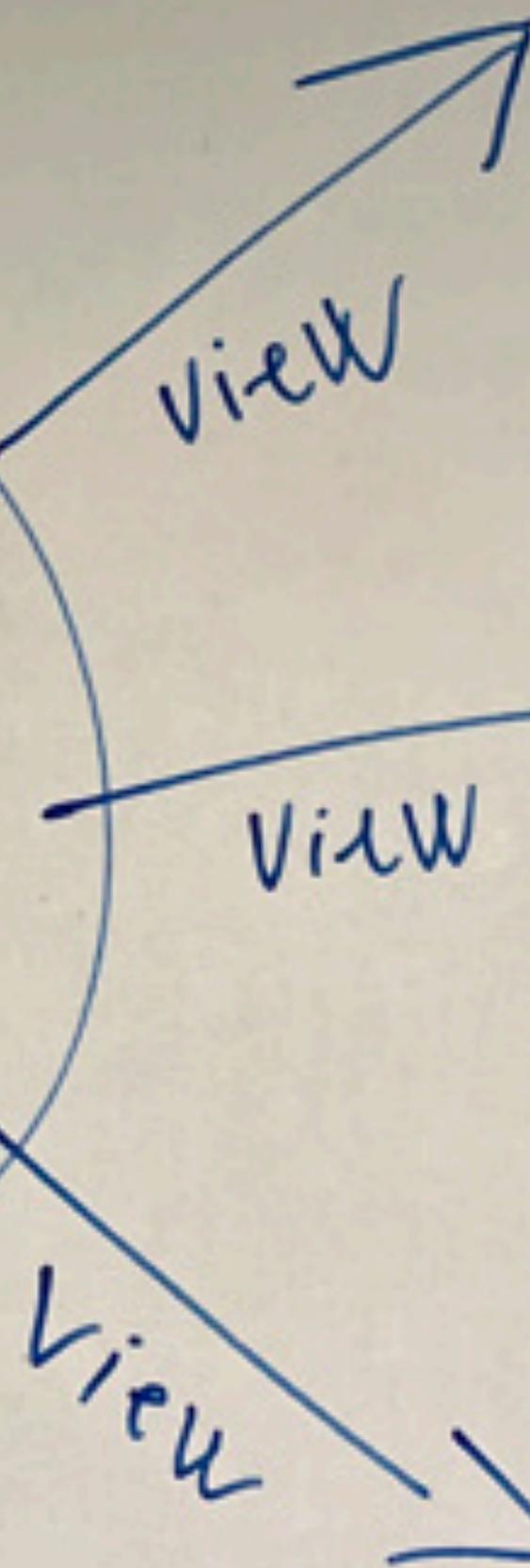
In an SPA the router is in charge of displaying the right view by watching changes on the URL.

The router makes it look like the web app has multiple pages and HTML files. But it's just changing the view, what is displayed, inside of the single `index.html` file. The page transition and change of content, is done dynamically by JavaScript (`router.js`).



/create  
route

Router  
(router.js)



update  
page

users  
page

create  
page

```
✓ vanilla-js-spa-router-hash
  > css
  > img
  < JS app.js
  < JS router.js
```

```
<> index.html
```

```
* All routes of the SPA
* "path": "id of page in DOM"
*/
const _routes = {
  "#/": "home",
  "#/about": "about",
  "#/clients": "clients",
  "#/contact": "contact"
};
const _pages = document.querySelectorAll(".page");
const _basePath = location.pathname.replace("index.html", "");
const _navLinks = document.querySelectorAll("nav a");

/**
 * Changing display to none for all pages
 */
function hideAllPages() {
  for (const page of _pages) {
    page.style.display = "none";
  }
}

/**
 * Navigating SPA to specific page by given path
 */
function navigateTo(path) {
  window.history.pushState({}, path, _basePath + path);
  showPage(path);
}

/**
 * Displaying page by given path
*/
function showPage(path) {
```

## ✓ vanilla-js-spa-router-hash

```
const routes = {  
  "/": "home",  
  "/about": "about",  
  "/clients": "clients",  
  "/contact": "contact"  
};
```

route

id of view

```
<body>  
  <!----- Navigation ----->  
  <nav>  
    <a class="nav-link" href="#/">Home</a>  
    <a class="nav-link" href="#/about">About</a>  
    <a class="nav-link" href="#/clients">Clients</a>  
    <a class="nav-link" href="#/contact">Contact</a>  
  </nav>  
  
  <main id="appRoot">  
    <!----- Pages ----->  
    <!-- home page -->  
    <section id="home" class="page">...</section>  
    <!-- about page -->  
    <section id="about" class="page">...</section>  
    <!-- clients page -->  
    <section id="clients" class="page">...</section>  
    <!-- contact page -->  
    <section id="contact" class="page">...</section>  
  </main>
```



# change default event for all .nav-link items

```
<!------- Navigation ----->
<nav>
  <a class="nav-link" href="#/">Home</a>
  <a class="nav-link" href="#/about">About</a>
  <a class="nav-link" href="#/clients">Clients</a>
  <a class="nav-link" href="#/contact">Contact</a>
</nav>
```

```
/** 
 * Attaching event to nav links and preventing default anchor
 */
function attachNavLinkEvents() {
  const navLinks = document.querySelectorAll(".nav-link");
  for (const link of navLinks) {
    link.addEventListener("click", function (event) {
      const path = link.getAttribute("href");
      navigateTo(path);
      event.preventDefault();
    });
  }
}
```

when a `.nav-link` item is clicked,  
`navigateTo(...)` is called with the value of `href`

```
<!------- Navigation ----->
<nav>
  <a class="nav-link" href="#/">Home</a>
  <a class="nav-link" href="#/about">About</a>
  <a class="nav-link" href="#/clients">Clients</a>
  <a class="nav-link" href="#/contact">Contact</a>
</nav>
```

```
/** 
 * Navigating SPA to specific page by given path
 */
function navigateTo(path) {
  window.history.pushState({}, path, _basePath + path);
  showPage(path);
}

/** 
 * Displaying page by given path
 */
function showPage(path) {
  hideAllPages(); // hide all pages
  document.querySelector(`#${_routes[path]}`).style.display = "block";
  setActiveTab(path);
}
```

# PRODUCT SPA

[BACK](#)

SINGLE PAGE WEB APP TEMP X Products 127.0.0.1:spa-products-fetch-json-enhanced/index.html

## PRODUCTS

Order by: Choose here ▾ Show out of stock  Search

Product	Brand	Price	Status	Action
MacBook Pro 13"	Apple	11799 kr.	outOfStock	<a href="#">EDIT</a> <a href="#">DELETE</a>
MacBook Pro 15"	Apple	21499 kr.	inStock	<a href="#">EDIT</a> <a href="#">DELETE</a>
Zenbook 14"	ASUS	8099 kr.	outOfStock	<a href="#">EDIT</a> <a href="#">DELETE</a>
Surface Laptop 3	Microsoft	1299 kr.	inStock	<a href="#">EDIT</a> <a href="#">DELETE</a>
ASUS Zenbook 13	ASUS	1499 kr.	inStock	<a href="#">EDIT</a> <a href="#">DELETE</a>
MacBook Air M1	Apple	1399 kr.	inStock	<a href="#">EDIT</a> <a href="#">DELETE</a>

PRODUCTS ADD PRODUCT

# PRODUCT SPA

[BACK](#)

- Use your knowledge about JavaScript and SPA to build a web app with products.
- Use any spa template you like as your code base (`vanilla-js-spa-router-hash`).
- Define products as json in `json/products.json`
- Add at least 4 JSON objects with the following properties: `model`, `brand`, `id`, `price`, `status` and `img`
- Two of your products must have the status “`outOfStock`” and two “`inStock`”.
- Use `fetch()` to get the data from the JSON file. Append and display all products to a page called Products in your SPA.
- Implement search functionality: Search on model and/or brand.
- Implement the add functionality.:Add another page with a form to add new products.
- Extra / advanced:
  - Add functionality to filter and/or sort by the `status`. You could add an input checkbox to show only products “`inStock`”. Make use of your knowledge about functions, loops, filter, etc.
  - Add functionality to handle edit and delete og every product.
  - Reimplement the layout of the SPA by using a fullpage grid and/or flex bow layout
  - Customise and style your web app.

SEARCH, CREATE, UPDATE  
& DELETE OBJECTS

# SEARCH

```
function search(value) {  
    let searchQuery = value.toLowerCase();  
    let filteredProducts = [];  
    for (let product of _products) {  
        let model = product.model.toLowerCase();  
        let brand = product.brand.toLowerCase();  
        if (model.includes(searchQuery) || brand.includes(searchQuery)) {  
            filteredProducts.push(product);  
        }  
    }  
    appendProducts(filteredProducts);  
}
```

```
function search(value) {  
    value = value.toLowerCase();  
    let filteredTeachers = [];  
    for (let teacher of teachers) {  
        let name = teacher.name.toLowerCase();  
        if (name.includes(value)) {  
            filteredTeachers.push(teacher);  
        }  
    }  
    appendTeachers(filteredTeachers);  
}
```

FOR OF LOOP

## .FILTER & ARROW FUNCTION

```
function search(value) {  
    let searchValue = value.toLowerCase();  
    let filteredTeachers = _teachers.filter(teacher => teacher.title.rendered.toLowerCase().includes(searchValue));  
    appendTeachers(filteredTeachers);  
}
```

# SEARCH

```
let _products = [];  
  
function search(value) {  
    let searchQuery = value.toLowerCase();  
    let filteredProducts = [];  
    for (let product of _products) {  
        let model = product.model.toLowerCase();  
        let brand = product.brand.toLowerCase();  
        if (model.includes(searchQuery) || brand.includes(searchQuery)) {  
            filteredProducts.push(product);  
        }  
    }  
    appendProducts(filteredProducts);  
}
```

# SEARCH

```
function search(value) {  
    let searchQuery = value.toLowerCase();  
    let filteredProducts = [];  
    for (let product of _products) {  
        let model = product.model.toLowerCase();  
        let brand = product.brand.toLowerCase();  
        if (model.includes(searchQuery) || brand.includes(searchQuery)) {  
            filteredProducts.push(product);  
        }  
    }  
    appendProducts(filteredProducts);  
}  
  
function search(value) {  
    let searchQuery = value.toLowerCase();  
    let filteredProducts = _products.filter(product => {  
        let model = product.model.toLowerCase();  
        let brand = product.brand.toLowerCase();  
        if (model.includes(searchQuery) || brand.includes(searchQuery)) {  
            return product;  
        }  
    }  
);  
appendProducts(filteredProducts);  
}
```

when you ask Rasmus  
for help and he says  
"Read documentation"



String.prototype.includes() - JavaScript Reference | MDN

developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\_Objects/String/includes

MDN Web Docs

Technologies References & Guides Feedback

Site search... (Press "/" to focus)

Web technology for developers > JavaScript > JavaScript reference > Standard built-in objects > String > String.prototype.includes()

Change language

## Table of contents

- Syntax
- Description
- Polyfill
- Examples
- Specifications
- Browser compatibility
- See also

## Related Topics

- Standard built-in objects
  - String
  - Properties
    - String length
  - Methods
    - String.prototype[`@@iterator`]()
    - String.prototype.anchor()
    - String.prototype.at()
    - String.prototype.big()
    - String.prototype.blink()
    - String.prototype.bold()

# String.prototype.includes()

The `includes()` method performs a case-sensitive search to determine whether one string may be found within another string, returning `true` or `false` as appropriate.

### JavaScript Demo: String.includes()

```
1 const sentence = 'The quick brown fox jumps over the lazy dog.';  
2  
3 const word = 'fox';  
4  
5 console.log(`The word "${word}" ${sentence.includes(word) ? 'is' : 'is not'} in the sentence`);  
6 // expected output: "The word "fox" is in the sentence"  
7
```

Run >

Reset

## Syntax

```
includes(searchString)  
includes(searchString, position)
```

# CREATE

```
/*
global variables: _products, _selectedProductId
*/
let _products = [];

function addNewProduct() {
  showLoader(true);

  let brand = document.querySelector('#brand').value;
  let model = document.querySelector('#model').value;
  let price = document.querySelector('#price').value;
  let img = document.querySelector('#img').value;
  const id = Date.now(); // dummy generated user id

  if (brand && model && price && img) {
    _products.push({
      brand,
      model,
      price,
      img,
      status: 'inStock',
      id
    });

    appendProducts(_products);
    navigateTo('products');
  } else {

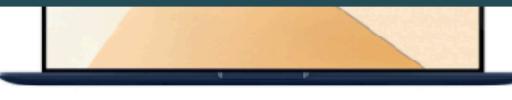
```

# UPDATE

```
function appendProducts(products) {
  let htmlTemplate = "";
  for (let product of products) {
    htmlTemplate += /*html*/
      <article class="${product.status}">
        <article onclick="showDetailView(${product.id})">
          
          <h2>${product.model}</h2>
          <h3>${product.brand}</h3>
          <p>Price: ${product.price} kr.</p>
          <p class="status">Status: ${product.status}</p>
        </article>
        <button onclick="goUpEdit(${product.id})">Edit</button>
        <button onclick="deleteProduct(${product.id})">Delete</button>
      </article>
    `;
  }
  document.querySelector('#products-container').innerHTML = htmlTemplate;
}
```

JavaScript Async Products 127.0.0.1:5501/spa-products-fetch-json-enhanced/index.html

## PRODUCTS



**Zenbook 14"**  
ASUS  
Price: 8099 kr.  
Status: outOfStock

EDIT DELETE



**Ideapad S340 14"**  
(platinum grey)  
Lenovo  
Price: 6099 kr.

button 57.31 × 27.78  
EDIT DELETE



**ASUSBOOK 12"**  
ASUS  
Price: 2099 kr.  
Status: outOfStock



**MacBook Pro 13" M2**  
Apple  
Price: 20499 kr.  
Status: inStock

Elements Console Sources Network »

- > <article class="outOfStock">...</article>
- ><article class="inStock">
- ><article onclick="showDetailView(1)">...</article>
- <button onclick="goUpEdit(1)">Edit</button>
- <button onclick="deleteProduct(1)">Delete</button>
- </article>
- ><article class="outOfStock">...</article>
- ><article class="inStock">
- ><article onclick="showDetailView(3)">...</article>
- <button onclick="goUpEdit(3)">Edit</button> == \$0
- <button onclick="deleteProduct(3)">Delete</button>
- </article>
- ><article class="outOfStock">...</article>
- ><article class="inStock">...</article>
- </section>

... ion#products.page section#products-container.grid-container article.inStock button ...

Styles Computed Layout Event Listeners DOM Breakpoints Properties »

Filter :hov .cls +

```
element.style {  
}  
  
button {  
    font-weight: 300;  
    text-align: center;  
    cursor: pointer;  
    border: ▷ none;  
    border-radius: ▷ 0;  
    color: □ var(--text-color-light);  
    background-color: □ var(--green);  
    letter-spacing: 0.05em;  
    text-transform: uppercase;  
    padding: ▷ .5em 1em;  
}
```

main.css:233

# UPDATE

```
function goToEdit(id) {  
    // save id in global variable  
    _selectedProductId = id;  
    // find product to edit by using array.find and id  
    const productToEdit = _products.find(product => product.id === _selectedProductId);  
    // set input field values with the productToEdit properties  
    document.querySelector('#brandEdit').value = productToEdit.brand;  
    document.querySelector('#modelEdit').value = productToEdit.model;  
    document.querySelector('#priceEdit').value = productToEdit.price;  
    document.querySelector('#imgEdit').value = productToEdit.img;  
    //navigate to edit view  
    navigateTo("edit");  
}
```

JavaScript Async Products 127.0.0.1:5501/spa-products-fetch-json-enhanced/index.html#edit

## EDIT PRODUCT

BACK

Lenovo

Ideapad S340 14" (platinum grey)

6099

https://www.elgiganten.dk/image/dv\_web\_D180001002

SAVE

Elements Console Sources Network » 1 | X

```
<!-- create page -->
<section id="add" class="page" style="display: none;">...</section>
<!-- edit page -->
... <section id="edit" class="page" style="display: block;"> == $0
  <header class="topbar">...</header>
  <form>
    <input id="brandEdit" type="text" name="brand" placeholder="Brand">
    <input id="modelEdit" type="text" name="model" placeholder="Model">
    <input id="priceEdit" type="text" name="price" placeholder="Price">
    <input id="imgEdit" type="text" name="img" placeholder="Image URL">
    <button type="button" name="button" onclick="saveProduct()">Save
    </button>
  </form>
</section>
<!-- edit page -->
```

html body section#edit.page

Styles Computed Layout Event Listeners DOM Breakpoints Properties »

Filter :hov .cls + [ ]

```
element.style {
  display: block;
}

.page {
  display: none;
  min-height: calc(100vh - 110px);
  background: var(--light-grey);
  animation: fadeIn 0.4s;
  padding: 55px 0;
}

section {
  display: block;
}
```

main.css:113 user agent stylesheet

PRODUCTS ADD PRODUCT

section#edit.page 726 x 946

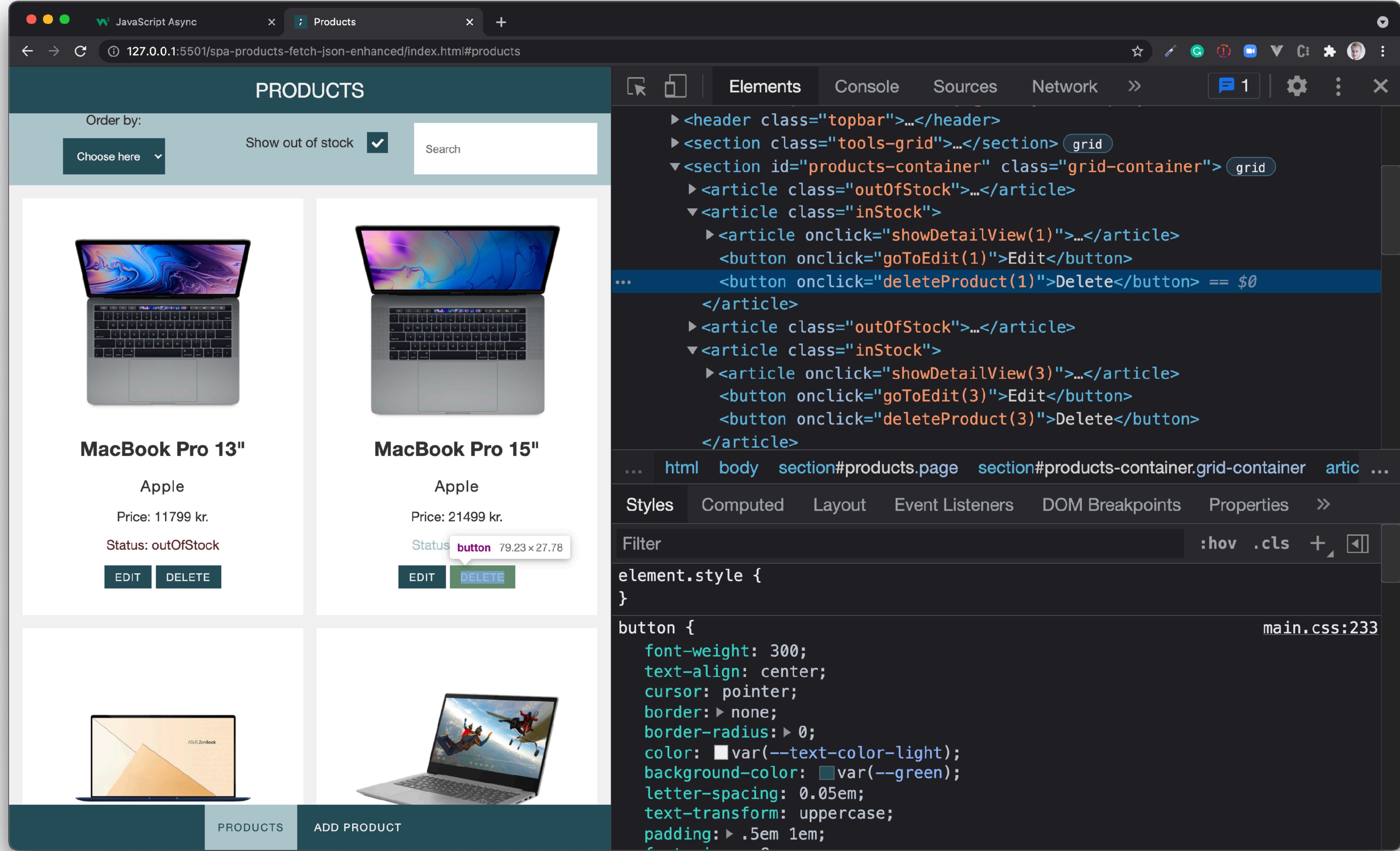
# UPDATE

```
function saveProduct() {  
    // find index of the product to update in _products  
    let index = _products.findIndex(product => product.id === _selectedProductId);  
    // update values of user in array  
    _products[index].brand = document.querySelector('#brandEdit').value;  
    _products[index].model = document.querySelector('#modelEdit').value;  
    _products[index].price = document.querySelector('#priceEdit').value;  
    _products[index].img = document.querySelector('#imgEdit').value;  
    // update dom usind appendProducts()  
    appendProducts(_products);  
    //navigating back  
    navigateTo("products");  
}
```

```
// find user to update by given user id  
const userToUpdate = _users.find(user => user.id === _selectedUserId);  
// update values of user in array  
userToUpdate.name = nameInput.value;  
userToUpdate.mail = mailInput.value;
```

# DELETE

```
function appendProducts(products) {
  let htmlTemplate = "";
  for (let product of products) {
    htmlTemplate += /*html*/
      <article class="${product.status}">
        <article onclick="showDetailView(${product.id})">
          
          <h2>${product.model}</h2>
          <h3>${product.brand}</h3>
          <p>Price: ${product.price} kr.</p>
          <p class="status">Status: ${product.status}</p>
        </article>
        <button onclick="goUpEdit(${product.id})">Edit</button>
        <button onclick="deleteProduct(${product.id})">Delete</button>
      </article>
    ;
  }
  document.querySelector('#products-container').innerHTML = htmlTemplate;
}
```



# DELETE

```
function deleteProduct(id) {  
    // filter _products - all products that doesnt have the id  
    _products = _products.filter(product => product.id !== id);  
    appendProducts(_products);  
}
```

# SORT, FILTER & ARROW FUNCTIONS

JavaScript Async

Products

127.0.0.1:5501/spa-products-fetch-json-enhanced/index.html#products

## PRODUCTS

Order by: Choose here ▾ Show out of stock ✓ Search



**MacBook Pro 13"**

Apple

Price: 11799 kr.

Status: outOfStock

[EDIT](#) [DELETE](#)



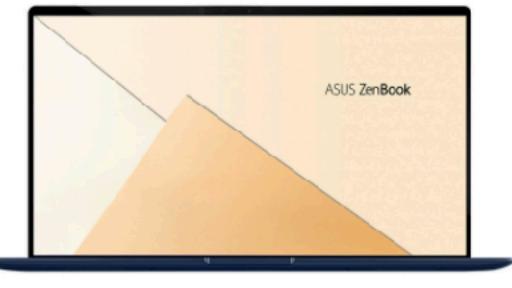
**MacBook Pro 15"**

Apple

Price: 21499 kr.

Status: inStock

[EDIT](#) [DELETE](#)



ASUS ZenBook

[PRODUCTS](#) [ADD PRODUCT](#)

Elements Console Sources »

```
<section id="products" class="page" style="display: block;">...

Order by:

...... == $0 out of stock





MacBook Pro 13"



Apple



Price: 11799 kr.



Status: outOfStock



EDIT DELETE





MacBook Pro 15"



Apple



Price: 21499 kr.

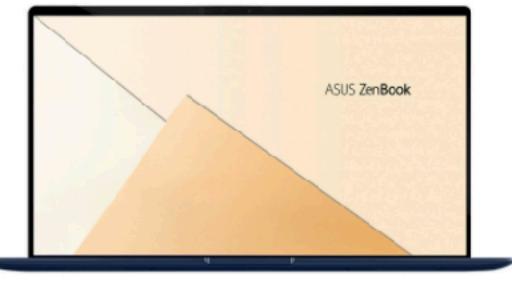


Status: inStock



EDIT DELETE





ASUS ZenBook



PRODUCTS ADD PRODUCT


```

...

html body section#products.page section.tools-grid label select ...

Styles Computed Layout Event Listeners DOM Breakpoints »

Filter :hov .cls +

element.style {

}

select {

background-color: var(--green);

border: none;

color: var(--text-color-light);

padding: 1em;

margin: 1em;

max-width: 350px;

}

select:not(:-internal-list-box) {

overflow: visible !important;

}

main.css:366

user agent stylesheet

# SORT

```
function orderBy(option) {
  if (option === "brand") {
    orderByBrand();
  } else if (option === "model") {
    orderByModel();
  } else if (option === "price") {
    orderByPrice();
  }
}

function orderByBrand() {
  _products.sort((product1, product2) => {
    return product1.brand.localeCompare(product2.brand);
  });
  appendProducts(_products);
}
```

```
function orderByBrand() {  
  _products.sort((product1, product2) => {  
    return product1.brand.localeCompare(product2.brand);  
  });  
  appendProducts(_products);  
}
```

```
function orderByModel() {  
  _products.sort((product1, product2) => {  
    return product1.model.localeCompare(product2.model);  
  });  
  appendProducts(_products);  
}
```

```
function orderByPrice() {  
  _products.sort((product1, product2) => {  
    return product1.price - product2.price;  
  });  
  appendProducts(_products);  
}
```

.SORT & ARROW FUNCTIONS  
INSIDE FUNCTIONS DECLARATIONS

JavaScript Async

Products

127.0.0.1:5501/spa-products-fetch-json-enhanced/index.html#products

label.checkmark-container 178.09x72

Order by: Choose here ▾ Show out of stock ✓ Search

MacBook Pro 13"

Apple

Price: 11799 kr.

Status: outOfStock

EDIT DELETE

MacBook Pro 15"

Apple

Price: 21499 kr.

Status: inStock

EDIT DELETE

ASUS ZenBook

PRODUCTS ADD PRODUCT

Elements Console Sources »

label.checkmark-container

<header class="topbar">...</header>

<section class="tools-grid"> grid

<label for="sortBy">

"Order by: "

<select id="sortBy" onchange="orderBy(this.value)">...</select>

</label>

<label for="outOfStock" class="checkmark-container">

"Show out of stock "

<input type="checkbox" id="outOfStock" onchange="showHideOutStock(this.checked)" checked> == \$0

<span class="checkmark">...</span>

</label>

<input type="search" placeholder="Search" onkeyup="search(this.value)" onsearch="search('')">

... page section.tools-grid label.checkmark-container input#outOfStock ...

Styles Computed Layout Event Listeners DOM Breakpoints »

Filter :hov .cls +

element.style {

}

.checkmark-container input {

position: absolute;

opacity: 0;

cursor: pointer;

height: 0;

width: 0;

}

input {

margin: 1em auto;

width: 100%;

max-width: 350px;

main.css:313

main.css:247

# FILTER

```
function showHideOfStock(checked) {  
  if (checked) {  
    appendProducts(_products);  
  } else {  
    const inStockProducts = _products.filter(product => product.status === "inStock");  
    appendProducts(inStockProducts);  
  }  
}
```

C O D E   E V E R Y   D A Y !