Most popular programming languages on GitHub

Source: GitHub Archive

CODERSRANK.IO

JAVASCRIPT & WEB APP DEVELOPMENT

FRONTEND DEVELOPER

BUSINESS ACADEMY AARHUS
UNIVERSITY OF APPLIED SCIENCES

AGENDA

Wrap up & Canvas Users Case

Function-based JS

Canvas Users SPA Case

AGENDA

10.00
var, object, array, loop, array methods & template string

11:00
Web App, SPA & Routing

12:07
read: fetch, JSON, async/ await & promises

13:09
global variables, create, update, delete, sort, filter & arrow function

EXERCISES

Test & explore: Product SPA

Wrap up: Canvas Users Case

Canvas Users SPA Case

Product SPA

Check out: `spa-products-fetch-json`

BACK

In pairs (do your best!):

1. Test the solution and explore the project structure. What files are inside the project and how are they connected?
2. What is the entry file? And how are the JS files loaded? How's the JSON file loaded?
3. Go through `router.js` and help each other explaining the overall functionality of the router (I'm going to explain SPA & router in detail later on today).
4. Go through `app.js` and try to explain every function (take turns).
5. In the `app.js` script, add missing comments before every function (copy the project into your own dev folder).
6. Note your comments and questions.

WRAP UP: CANVAS USERS CASE

CANVAS USERS CASE #1

BACK

CREATE A GRID DISPLAYING USERS FROM CANVAS

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

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.

FUNCTION-BASED JS

OR FUNCTIONAL PROGRAMMING

FUNCTION-BASED VS OBJECT-ORIENTED JAVASCRIPT

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.

app.js — web-frontend

JS app.js

```javascript
"use strict";

/* ---------- Global Variables ---------- */

let _products = [];
let _selectedProductId;

/* ------------------------------------ */

async function initApp() {
  _products = await fetchData();
  appendProducts(_products);
  showLoader(false);
}

initApp();

/**
 * Getting product data from JSON file
 */
async function fetchData() {
  const response = await fetch('json/products.json');
  const data = await response.json();
  return data;
}

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

main ⊗ 0 ⚠ 0 Ln 52, Col 54 Spaces: 2 UTF-8 LF

It’s clean and straightforward.

Some would say it’s easier.

It depends 🤷🏼‍♂️

app.js — web-frontend

app.js

```javascript
"use strict";

/* ---------- Global Variables ---------- */

let _products = [];
let _selectedProductId;

/* ------------------------------------ */

async function initApp() {
  _products = await fetchData();
  appendProducts(_products);
  showLoader(false);
}

initApp();

/**
 * Getting product data from JSON file
 */
async function fetchData() {
  const response = await fetch('json/products.json');
  const data = await response.json();
  return data;
}

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

main ⊗ 0 ⚠ 0 Ln 52, Col 54 Spaces: 2 UTF-8 LF

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

https://scrimba.com/p/pPPeCy/cpKZdcQ

FUNCTIONS

FUNCTION DECLARATION

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

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

https://javascript.info/function-basics

FUNCTIONS

FUNCTION DECLARATION

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

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

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

https://javascript.info/function-basics

FUNCTIONS

FUNCTION DECLARATION

```
function append(htmlTemplate, idOfElement) {
  console.log(htmlTemplate);
  document.getElementById(idOfElement).innerHTML += htmlTemplate;
  alert("Yeaaah, you did it!");
}

append("<h2>Hi Frontenders!</h2>", "content");
```

https://javascript.info/function-basics

FUNCTIONS

FUNCTION DECLARATION

The name of the function

Parameters

```
function append(htmlTemplate, idOfElement) {
  console.log(htmlTemplate);
  document.getElementById(idOfElement).innerHTML += htmlTemplate;
  alert("Yeaaah, you did it!");
}

append("<h2>Hi Frontenders!</h2>", "content");
```

Body of the function
(code block)

How to call the function

https://javascript.info/function-basics

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);
```

Body of the function
(code block)

How to call the function

FUNCTION CALLED BY ANOTHER FUNCTION

```
let users = […
    }];

function appendUsers(users) {
    let htmlTemplate = "";
    for (const user of users) {
        console.log(user);
        htmlTemplate += /*html*/`
            <article>
                <img src="${user.avatar_url}">
                <h2>${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();
```

Use the name of the function to call / execute

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

ARROW FUNCTIONS

SHORTER 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

Template: `spa-canvas-users-template`

Course roster: WU-E21s - 1. se

eaaa.instructure.com/courses/13351/users

Account
Admin
Dashboard
Courses
Calendar
Inbox
History
Commons
Student links

WU-E21s › People

Student view

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

Elements | Console | Sources | Network | Performance

Preserve log | Disable cache | No throttling

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
- unread_count
- group_categories...
- users?include_ina...
- collect?v=1&_v=j9...
- unread_count
- courses?include[]...
- accounts

7 / 138 requests | 10.2

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
  1: {id: "17929", name: "Nicklas Eibye Andersen", created_at: "2019-08-07T0
  2: {id: "17476", name: "Piotr Andrzej Pospiech", created_at: "2019-08-02T0
  3: {id: "30257", name: "Lasse Bickmann", created_at: "2021-08-05T00:58:06+
  4: {id: "30252", name: "Thomas Hyllegaard Busk", created_at: "2021-08-05T0
  5: {id: "11879", name: "Jesper Nissen Byg", created_at: "2018-08-08T02:10:
  6: {id: "17960", name: "Barbora Byrtusová", created_at: "2019-08-07T02:34:
  7: {id: "14427", name: "Rasmus Cederdorff", created_at: "2018-09-06T02:23:
    avatar_url: "https://eaaa.instructure.com/images/thumbnails/649049/1EeRF
    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: "2017-05-10T14:09:2
  9: {id: "30251", name: "Sarah Øster Dybvad", created_at: "2021-08-05T00:57
  10: {id: "17477", name: "Kristine Dzumakajeva", created_at: "2019-08-02T02
```

spa-products-fetch-json-enhanced

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, image, mail, course` & `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 og 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.

VARIABLES

... ARE USED TO STORE DATA IN THE MEMORY

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;
```

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

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

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

```html
<body>
  <header>
    <h1>PROJECT TEMPLATE</h1>
  </header>
  <section id="content"></section>

  <!-- main js file -->
  <script src="js/main.js"></script>
</body>
```

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

```javascript
// 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

VARIABLES

... STORE DATA IN THE MEMORY

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

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

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`.

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

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 `±(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.

https://javascript.info/types

OBJECTS

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.");
```

```
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

ARRAYS

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

ARRAYS

Rasmus Cederdorff
Position:
Lecturer
Department/
Multimedia De
Digital Concep
Address:
Ringvej Syd 10
Mail:
race@baaa.dk
Phone:
7228 6318
Michael Hvidtfeldt
Position:
Lecturer
Department/
Multimedia De
Address:
Ringvej Syd 10
Mail:
mhv@baaa.dk
Phone:
7228 6328
Birgitte Kirk Iversen
Position:
Senior Lecturer
Department/programme:
Multimedia Design
Address:
Sønderhøj 30, 8260 Viby J
Mail:
bki@baaa.dk
Phone:
7228 6316

push
pop

Course roster: WU-E21s - 1. se

eaaa.instructure.com/courses/13351/users

Account
Admin
Dashboard
Courses
Calendar
Inbox
History
Commons
Student links

WU-E21s › People

Student view

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

Elements | Console | Sources | Network | Performance

Preserve log | Disable cache | No throttling

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

property

value

Name

- unread_count
- group_categories...
- users?include_ina...
- collect?v=1&_v=j9...
- unread_count
- courses?include[]...
- accounts

7 / 138 requests | 10.2

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
  1: {id: "17929", name: "Nicklas Eibye Andersen", created_at: "2019-08-07T0
  2: {id: "17476", name: "Piotr Andrzej Pospiech", created_at: "2019-08-02T0
  3: {id: "30257", name: "Lasse Bickmann", created_at: "2021-08-05T00:58:06+
  4: {id: "30252", name: "Thomas Hyllegaard Busk", created_at: "2021-08-05T0
  5: {id: "11879", name: "Jesper Nissen Byg", created_at: "2018-08-08T02:10:
  6: {id: "17960", name: "Barbora Byrtusová", created_at: "2019-08-07T02:34:
  7: {id: "14427", name: "Rasmus Cederdorff", created_at: "2018-09-06T02:23:
    avatar_url: "https://eaaa.instructure.com/images/thumbnails/649049/1EeRF
    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: "2017-05-10T14:09:2
  9: {id: "30251", name: "Sarah Øster Dybvad", created_at: "2021-08-05T00:57
  10: {id: "17477", name: "Kristine Dzumakajeva", created_at: "2019-08-02T02
```

DATA 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://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-teachers-search

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) i
  ▶ 0: {age: 40, name: "Pete"}
  ▶ 1: {age: 44, name: "Mary"}
    length: 2
```

ARRAY METHODS

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

Chapter

Data types

Lesson navigation

Add/remove items

Iterate: forEach

Searching in array

Transform an array

Array.isArray

Most methods support "thisArg"

Summary

Tasks (13)

Comments

Share

Edit on GitHub

Ads

filter

The `find` method looks for a single (first) element that makes the function return `true`.

If there may be many, we can use arr.filter(fn).

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

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

For instance:

```
let users = [
  {id: 1, name: "John"},
  {id: 2, name: "Pete"},
  {id: 3, name: "Mary"}
];

// returns array of the first two users
let someUsers = users.filter(item => item.id < 3);

alert(someUsers.length); // 2
```

Transform an array

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

map

The arr.map method is one of the most useful and often used.

It calls the function for each element of the array and returns the array of results.

ARRAY METHODS

https://www.w3schools.com/jsref/jsref_obj_array.asp

JavaScript

JS Array
concat()
constructor
copyWithin()
entries()
every()
fill()
filter()
find()
findIndex()
forEach()
from()
includes()
indexOf()
isArray()
join()
keys()
length
lastIndexOf()
map()
pop()
prototype
push()
reduce()
reduceRight()
reverse()
shift()
slice()
some()

JavaScript Array Reference

‹ Previous Next ›

Array Object

The Array object is used to store multiple values in a single variable:

```
var cars = ["Saab", "Volvo", "BMW"];
```

Try it Yourself »

Array indexes are zero-based: The first element in the array is 0, the second is 1, and so on.

For a tutorial about Arrays, read our JavaScript Array Tutorial.

Array Properties

Property	Description

COLOR PICKER

ARRAY METHODS

.PUSH()

.FIND()

.MAP()

.REDUCE()

.FILTER()

.SORT()

.CONCAT()

...

Computer science student

Senior developer, 10+ years experience

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

`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

SPA & ROUTER PROJECTS

- vanilla-js-spa-router
  - css
  - img
  - js
    - app.js
    - router.js
  - index.html
- vanilla-js-spa-router-hash
  - css
  - img
  - js
    - app.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

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

S P A

*"**SPAs** are based on a single document model. This means that web applications' lifespan happens on a single html page, along with the transitions between the different views. But since links no longer imply the fetching and generation of a new document, how are those transitions modelled? they are achieved by using a **router**."*

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

Single-Page Applications (SPAs) are Web **apps** that load a **single** HTML **page** and dynamically update that **page** as the user interacts with the **app**. SPAs use AJAX and HTML5 to create fluid and responsive Web **apps**, without constant **page** reloads. However, this means much of the work happens on the client side, in JavaScript.

https://msdn.microsoft.com/en-us/magazine/dn463786.aspx

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.

S P A

- 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.

TRADITIONAL VS SPA

TRADITIONAL VS SPA LIFECYCLE

MULTI PAGE APPLICATION

VS

SINGLE PAGE APPLICATION

R O U T E R

“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

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

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

/create
route
Router
(router.js)
view
view
view
users
page
create
page
update
page

```javascript
 * 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", ""); // remove
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
 */
```

vanilla-js-spa-router-hash

route

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

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>

    <!-- clients page -->
    <section id="contact" class="page">…
    </section>

  </main>
```

change default event for all `.nav-link` items

```html
<!-------------- 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>
```

```javascript
/**
 * 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`

```html
<!-------------- 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>
```

```javascript
/**
 * 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);
}
```

SPA & ROUTER EXERCISE

BACK

- Inspect the `index.html` & `router.js` in `vanilla-js-spa-router-hash`
- Add another page called users.
- Remember to add a new section for users page, a nav-link and the route in `router.js`.
- Test your are able to navigate to the users page.
- Change the clients page to a posts page. Remember to update id's, href & routes.
- Test your solution.
- Explain `navigateTo()`, `showPage()` & `initRouter()`

WHAT THE "#/HASH"?

WITHOUT HASH

OPEN & RUN THIS PROJECT ONLY

index.html — vanilla-js-spa-router

EXPLORER

VANILLA-JS-SPA-ROUTER

- css
- img
- js
- index.html

index.html

```html
<!DOCTYPE html>
<html>

<head>
  <title>Single Page Application - RACE</title>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=
  <meta name="description" content="Get started with HTML, CSS and JavaScript">
  <meta name="author" content="Rasmus Cederdorff">
  <link rel="stylesheet" href="./css/styles.css">
  <link rel="shortcut icon" type="image/png" href="./img/favicon.png" />

</head>

<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 -->
```

CHANGE LIVE SERVER ROOT

REQUIRES SOME CONFIG WHEN UPLOAD TO SERVER/ OWN DOMAIN:

ALWAYS LOOK AT ROOT OR INDEX.HTML

OR USE HASH 🤷🏼‍♂️

index.html — vanilla-js-spa-router-hash

EXPLORER

VANILLA-JS-SPA-ROUTER-HASH

- css
- img
- js
- index.html

index.html

```html
<!DOCTYPE html>
<html>

<head>
  <title>Single Page Application - RACE</title>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=n
  <meta name="description" content="Get started with HTML, CSS and JavaScript">
  <meta name="author" content="Rasmus Cederdorff">
  <link rel="stylesheet" href="./css/styles.css">
  <link rel="shortcut icon" type="image/png" href="./img/favicon.png" />

</head>

<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 -->
```

READ

READ OBJECTS: FETCH, JSON, ASYNC/ AWAIT & GLOBAL VARIABLES

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

WEB APP

YOUR SPA

INDEX.HTML

MAIN.CCS

MAIN.JS

(API)

FETCH(...)

REQUEST

JSON DATA

RESPONSE

DATA SOURCE

FETCH

WEB APP

```
fetch("...wp-json/wp/v2/posts")
  .then(function(response) {
    return response.json();
  })
  .then(function(json) {
    appendPosts(json);
  });
```

REQUEST (fetch) →

← RESPONSE (then)

DATA SOURCE

```javascript
/*
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>
        <img src="${person.img}">
        <h4>${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;
}
```

```javascript
/*
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>
        <img src="${person.img}">
        <h4>${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;
}
```


```json
[
  {
    "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"
  }
]
```

```javascript
/*
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>
        <img src="${person.img}">
        <h4>${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)


```json
[
  {
    "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"
  }
]
```

```javascript
/*
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>
        <img src="${person.img}">
        <h4>${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;
}
```

```json
},
{
  "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

STRUCTURE
CONTENT

JS

FUNCTIONALITY
BEHAVIOR

CSS

LAYOUT
PRESENTATION

JSON

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

Project

- fetch
  - css
    - main.css
  - img
  - js
    - main.js
  - json
    - persons.json
  - index.html

index.html | persons.json | main.js

```
[{
    "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

FORMATTER & VALIDATOR

HTTPS://JSONFORMATTER.CURIOUSCONCEPT.COM

FETCH

JSON, DATA SOURCES & API'S

```
// Simple javascipt 😄

//Synchronous fetch using async/await.

// Usual way
✓ const jsonData = fetch('URL')
                    .then(response ⇒ response.json())
                    .then(json ⇒ console.log(json));

// Using await
✓ const jsonData = await fetch('URL').then(res ⇒ res.json())


// Shorter syntax 😍
✓ const jsonData = await (await fetch('URL')).json();
```

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

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

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

THE TWO STATES OF PROMISE

https://javascript.info/promise-basics

https://www.instagram.com/p/CVQGdNIjWMx/

```javascript
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
  }
);
```

https://www.w3schools.com/js/js_promise.asp

read
the
documentation
read the documentation
read
The
documentation
Ask Rasmus

```javascript
function fetchInstagramPosts(userName) {
  let url = `https://instagram.com/${userName}/?__a=1`;

  fetch(url)
    .then(function (response) {
      return response.json();
    })
    .then(function (data) {
      let posts = data.graphql.user.edge_owner_to_timeline_media.edges;
      appendPosts(posts);
    });
}
```

```javascript
async function fetchInstagramPosts(userName) {
  let url = `https://instagram.com/${userName}/?__a=1`;
  let response = await fetch(url);
  let data = await response.json();
  let posts = data.graphql.user.edge_owner_to_timeline_media.edges;
  appendPosts(posts);
}
```

```
/*
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();
```

GLOBAL VARIABLES

... & LOCAL VARIABLES

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

movie**s**

```
// 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>
        <img src="${movie.acf.img}">
        <p>${movie.acf.description}</p>
        <iframe src="${movie.acf.trailer}"></iframe>
      </article>
    `;
  }

  document.querySelector('#movies-container').innerHTML = htmlTemplate;
}
```

function argument

variable inside the function

Inline variable

Store the current object from the array

```
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>
        <img src="${movie.acf.img}">
```

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})">
```

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"

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()

```
const sentence = 'The quick brown fox jumps over the lazy dog.';

const word = 'fox';

console.log(`The word "${word}" ${sentence.includes(word) ? 'is' : 'is not'} in the sentence`);
// expected output: "The word "fox" is in the sentence"

```

Run ›

Reset

Syntax

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

CREATE

```
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 {
    alert('Please fill out all fields');
  }
  showLoader(false);
}
```

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})">
          <img src="${product.img}">
          <h2>${product.model}</h2>
          <h3>${product.brand}</h3>
          <p>Price: ${product.price} kr.</p>
          <p class="status">Status: ${product.status}</p>
        </article>
        <button onclick="goToEdit(${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

PRODUCTS ADD PRODUCT

Elements Console Sources Network

```
▶ <article class="outOfStock">…</article>
▼ <article class="inStock">
  ▶ <article onclick="showDetailView(1)">…</article>
    <button onclick="goToEdit(1)">Edit</button>
    <button onclick="deleteProduct(1)">Delete</button>
  </article>
▶ <article class="outOfStock">…</antecr>
▼ <article class="inStock">
  ▶ <article onclick="showDetailView(3)">…</article>
    <button onclick="goToEdit(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 {                                                main.css:233
  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;
```

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

BACK

EDIT PRODUCT

Lenovo

Ideapad S340 14" (platinum grey)

6099

https://www.elgiganten.dk/image/dv_web_D180001002

SAVE

PRODUCTS ADD PRODUCT

section#edit.page 726 × 946

Elements Console Sources Network

```
<!-- 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 {                                                  main.css:113
  display: none;
  min-height: calc(100vh - 110px);
  background: var(--light-grey);
  animation: fadeIn 0.4s;
  padding: 55px 0;
}
section {                                   user agent stylesheet
  display: block;
}
```

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})">
          <img src="${product.img}">
          <h2>${product.model}</h2>
          <h3>${product.brand}</h3>
          <p>Price: ${product.price} kr.</p>
          <p class="status">Status: ${product.status}</p>
        </article>
        <button onclick="goToEdit(${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

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

button 79.23 × 27.78

EDIT DELETE

PRODUCTS ADD PRODUCT

Elements Console Sources Network

```
▶ <header class="topbar">…</header>
▶ <section class="tools-grid">…</section> grid
▼ <section id="products-container" class="grid-container"> grid
  ▶ <article class="outOfStock">…</article>
  ▼ <article class="inStock">
    ▶ <article onclick="showDetailView(1)">…</article>
      <button onclick="goToEdit(1)">Edit</button>
      <button onclick="deleteProduct(1)">Delete</button> == $0
    </article>
  ▶ <article class="outOfStock">…</article>
  ▼ <article class="inStock">
    ▶ <article onclick="showDetailView(3)">…</article>
      <button onclick="goToEdit(3)">Edit</button>
      <button onclick="deleteProduct(3)">Delete</button>
    </article>
```

… html body section#products.page section#products-container.grid-container artic …

Styles Computed Layout Event Listeners DOM Breakpoints Properties

Filter :hov .cls

```
element.style {
}
```

```
button {                                                main.css:233
  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;
```

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

PRODUCTS ADD PRODUCT

Elements Console Sources

```
▼<section id="products" class="page" style="display: block;">
  ▶<header class="topbar">…</header>
  ▼<section class="tools-grid">  grid
    ▼<label for="sortBy">
       "Order by: "
      ▶<select id="sortBy" onchange="orderBy(this.value)">…
      </select> == $0
    </label>
    ▶<label for="outOfStock" class="checkmark-container">…
    </label>
      <input type="search" placeholder="Search" onkeyup="search
      (this.value)" onsearch="search('')">
    </section>
  ▼<section id="products-container" class="grid-container">
  grid
```

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

Styles Computed Layout Event Listeners DOM Breakpoints

Filter :hov .cls

```
element.style {
}
select {                                          main.css:366
  background-color: var(--green);
  border: ▶ none;
  color: var(--text-color-light);
  padding: ▶ 1em;
  margin: ▶ 1em;
  max-width: 350px;
}
select:not(:-internal-list-box) {          user agent stylesheet
  overflow: ▶ visible !important;
}
```

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 178.09 × 72

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

PRODUCTS ADD PRODUCT

Elements Console Sources

```
▶<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="showHid
     eOfStock(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 {                    main.css:313
  position: absolute;
  opacity: 0;
  cursor: pointer;
  height: 0;
  width: 0;
}
input {                                         main.css:247
  margin: ▸ 1em auto;
  width: 100%;
  max-width: 350px;
```

FILTER

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

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

Chapter
Data types

Lesson navigation
Add/remove items
Iterate: forEach
Searching in array
Transform an array
Array.isArray
Most methods support "thisArg"
Summary

Tasks (13)
Comments

Share

Edit on GitHub

Ad by Carbon

Byg og implementer Node.js-, Java- og Python-apps med Azure.

filter

The `find` method looks for a single (first) element that makes the function return `true`.

If there may be many, we can use arr.filter(fn).

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

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

For instance:

```
let users = [
  {id: 1, name: "John"},
  {id: 2, name: "Pete"},
  {id: 3, name: "Mary"}
];

// returns array of the first two users
let someUsers = users.filter(item => item.id < 3);

alert(someUsers.length); // 2
```

Transform an array

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

map

The arr.map method is one of the most useful and often used.

It calls the function for each element of the array and returns the array of results.

PRODUCT SPA

BACK

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.

NEXT UP

MORE SPA & ROUTING

FETCH & API'S, PROMISES, AWAIT AND ASYNC

CRUD & JSONBIN.IO

CODE EVERY DAY!