



Loops

... meget mere end bare .foreach (...)

Agenda

- Status
 - Hvad kan vi nu, og hvor skal vi hen?
- Loops og iterating
 - forEach
 - For
 - For Of
 - For In
 - While
- Kort om Array-metoder
 - .filter
 - .find
 - .sort

The screenshot shows a code editor window with a tab bar at the top containing "View", "Selection", "Find", "Debugger", "Packages", "Window", "Help", and "Native". Below the tabs, there is a list of file names on the left side: "products", "cederdorff.e", "index", "about", "started", "onchange", "ion", "ge", "inner.material", "ze_template", "teachers", "teachers.grid", "ui.nav.alert", "ui.template", and "ui.template.tab". The main area of the editor displays a file named "main.js" with the following content:

```
1  "use strict";
2
3  fetch("http://headlesscms.cederdorff.com/wp-json/wp/v2/posts?_embed")
4    .then(function(response) {
5      return response.json();
6    })
7    .then(function(json) {
8      appendPosts(json);
9    });
10
11 function appendPosts(posts) {
12   for (let post of posts) {
13     console.log(post);
14     document.querySelector("#grid-posts").innerHTML += `
15       <article>
16         <h3>${post.title.rendered}</h3>
17         <p>Email: <a href="mailto:${post.acf.email}">${post.acf.email}</a></p>
18         <p>Phone: ${post.acf.phone}</p>
19       </article>
20     `;
21   }
22 }
```

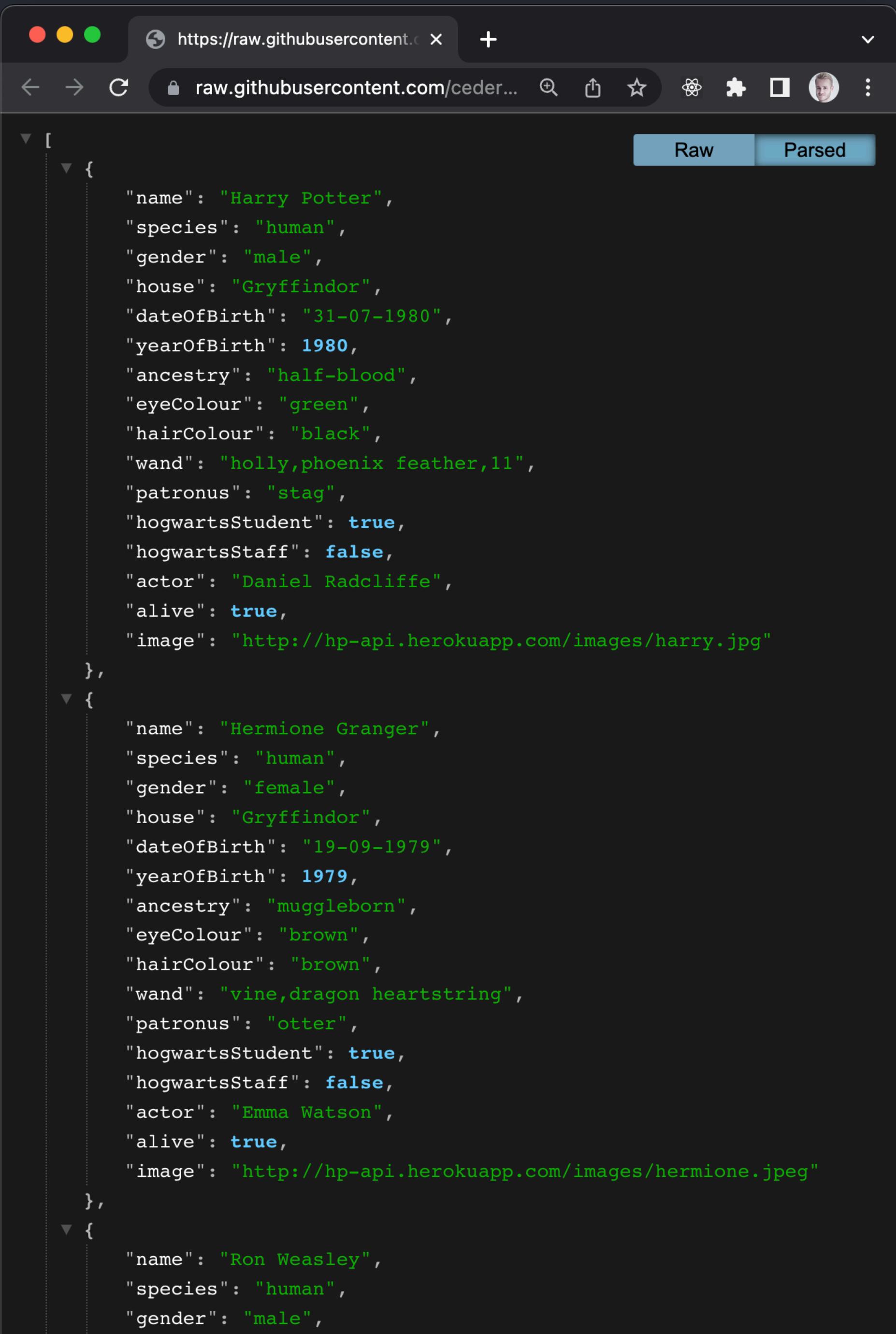
1. SEMESTERS STRUKTUR

Semesteret består af 16 undervisningsuger (inkl. en masse helligdage)



Data

JavaScript, lister,
objekter, JSON, fetch,
DOM-manipulation og
meget mere...



The screenshot shows a browser window displaying a JSON array of three characters from Harry Potter. The array contains objects for Harry Potter, Hermione Granger, and Ron Weasley, each with various properties like name, species, gender, house, date of birth, year of birth, ancestry, eye colour, hair colour, wand, patronus, Hogwarts status, staff status, actor, and image URL.

```
[{"name": "Harry Potter", "species": "human", "gender": "male", "house": "Gryffindor", "dateOfBirth": "31-07-1980", "yearOfBirth": 1980, "ancestry": "half-blood", "eyeColour": "green", "hairColour": "black", "wand": "holly,phoenix feather,11", "patronus": "stag", "hogwartsStudent": true, "hogwartsStaff": false, "actor": "Daniel Radcliffe", "alive": true, "image": "http://hp-api.herokuapp.com/images/harry.jpg"}, {"name": "Hermione Granger", "species": "human", "gender": "female", "house": "Gryffindor", "dateOfBirth": "19-09-1979", "yearOfBirth": 1979, "ancestry": "muggleborn", "eyeColour": "brown", "hairColour": "brown", "wand": "vine,dragon heartstring", "patronus": "otter", "hogwartsStudent": true, "hogwartsStaff": false, "actor": "Emma Watson", "alive": true, "image": "http://hp-api.herokuapp.com/images/hermione.jpeg"}, {"name": "Ron Weasley", "species": "human", "gender": "male", "house": "Gryffindor", "dateOfBirth": "11-08-1980", "yearOfBirth": 1980, "ancestry": "pureblood", "eyeColour": "brown", "hairColour": "red", "wand": "willow,mongoose hair,12", "patronus": "ewe", "hogwartsStudent": true, "hogwartsStaff": false, "actor": "Rupert Grint", "alive": true, "image": "http://hp-api.herokuapp.com/images/ron.jpg"}]
```

Fetch Persons

127.0.0.1:5503/fetch-persons-grid/index.html

Apps KEA DAT DAT.js IT Arkitektur EAAA BOOKS

Fetch Persons

 <p>Birgitte Kirk Iversen Senior Lecturer hki@mail.dk</p>	 <p>Martin Aagaard Nøhr Lecturer mnr@mail.dk</p>	 <p>Rasmus Cederdorff Senior Lecturer rnc@mail.dk</p>
 <p>Dan Okkels Brendstrup Lecturer dob@mail.dk</p>	 <p>Line Skjødt Senior Lecturer & Internship Coordinator lskj@mail.dk</p>	 <p>Kasper Fischer Topp Lecturer keto@mail.dk</p>
 <p>Anne Kirketerp Head of Department anki@mail.dk</p>	 <p>Maria Louise Bendixen Senior Lecturer mlbe@basa.dk</p>	 <p>Marlene Ahlgreen Jensen Senior Lecturer maj@basa.dk</p>

Forståelse for objekter og arrays

Hvad ser du?

The screenshot shows a web browser window with the URL kea.dk/find-medarbejder#qx-kea-phonebook-7910. The page title is "Find en medarbejder - Københavns Erhvervsakademi". The navigation bar includes links for "For studerende", "For ansatte", "For alumner", "Blog", "Bibliotek", "Job", and "Kontakt". A language switcher shows "EN" and a search icon. The main heading is "FIND EN MEDARBEJDER". Below it is a search form with fields for "DEL AF NAVN, E-MAIL ELLER TELEFON" containing "RACE" and "AFDELING" dropdown set to "Alle". A search button labeled "Søg..." is present. To the right, there is a profile card for "RASMUS CEDERDORFF", a Lector at KEA Digital. The card includes a photo, name, title, email ("E: RACE@kea.dk"), address ("Guldbergsgade 29N, 2200 København N"), and a "LÆS MERE" link. At the bottom, there are footer sections for "KEA - KØBENHAVNS ERHVERVSAKADEMI", "OM OS", "FOR STUDERENDE", and "PRAKTISK INFO".

```
const race = {  
    name: "Rasmus Cederdorff",  
    position: "Lektor",  
    mail: "race@kea.dk",  
    department: "KEA Digital",  
    address: "Guldbergsgade 29N, 2200 København N",  
    image: "https://kea.dk/slir/w180-c1x1/images/us";  
};
```

Hvad ser du?

The screenshot shows a web browser window for the KEA website. The URL is kea.dk/find-medarbejder#qx-kea-phonebook-7910. The page title is "Find en medarbejder". The header includes the KEA logo and navigation links for students, staff, alumni, blog, library, jobs, and contact. A search bar at the top has fields for "DEL AF NAVN, E-MAIL ELLER TELEFON" and "AFDELING" (set to "KEA Digital"), with a "Søg..." button. Below the search bar, there are five circular profile icons representing different staff members. The profiles are arranged in two rows: the first row contains Lars Bogetoft, Magdalena Maria Otap, and Oskar Tuska; the second row contains Peter Lind and Rasmus Cederdorff. Each profile includes a small photo, the name, title, and contact information (phone, email, address).

Name	Title	Contact Information
Lars Bogetoft	Uddannelseschef	T: 51850497 M: 51850497 E: larb@kea.dk KEA Digital Lytten 37 2400 København NV
Magdalena Maria Otap	Adjunktvikar	E: mago@kea.dk KEA Digital Guldbergsgade 29N 2200 København N
Oskar Tuska	Adjunkt	E: ostu@kea.dk KEA Digital Guldbergsgade 29N 2200 København N
Peter Lind	Lektor	E: petl@kea.dk KEA Digital Guldbergsgade 29N 2200 København N
Rasmus Cederdorff	Lektor	E: RACE@kea.dk KEA Digital Guldbergsgade 29N 2200 København N

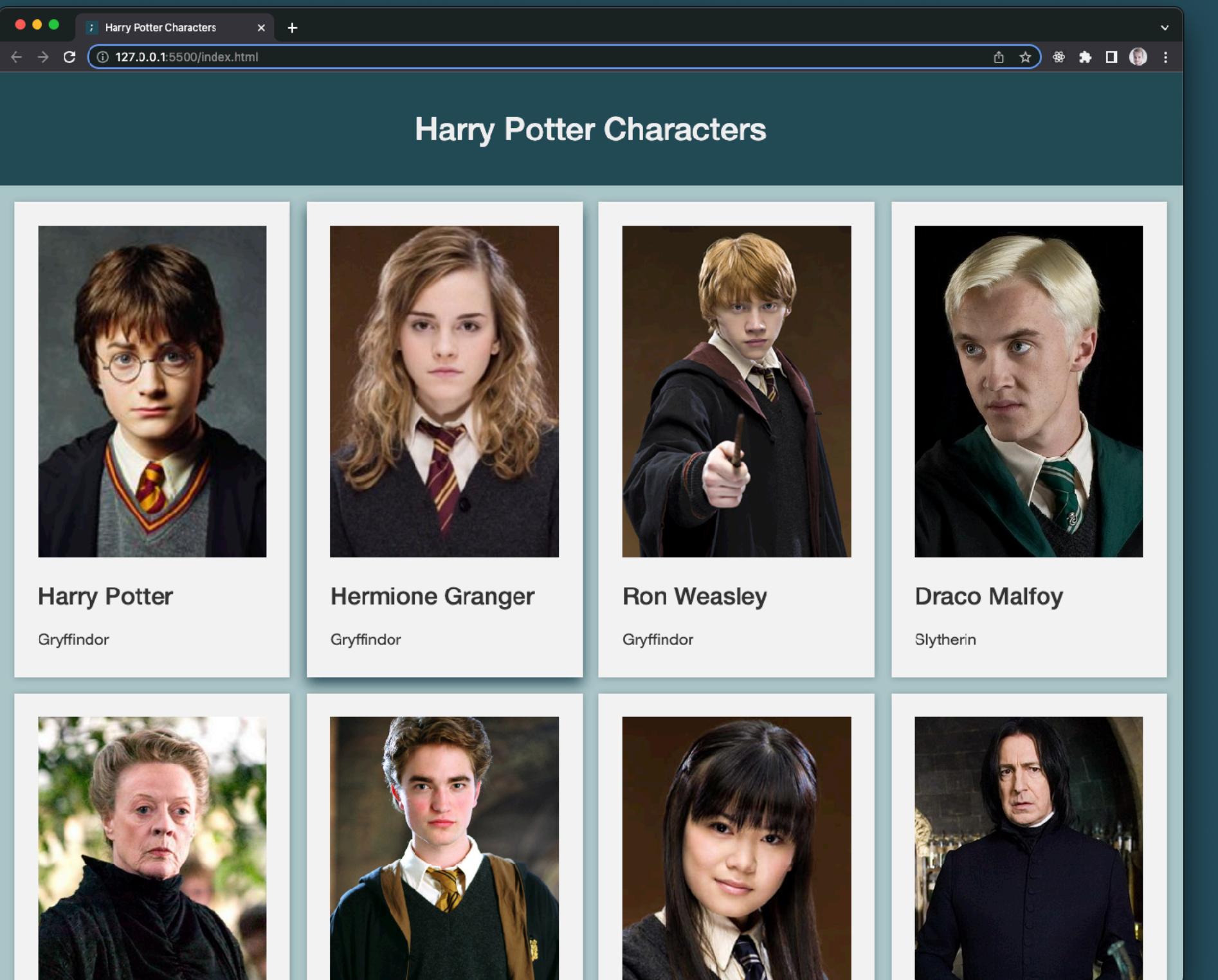
```
const lecturers = [
  {
    name: "Peter Lind",
    position: "Lektor",
    mail: "petl@kea.dk",
    department: "KEA Digital",
    address: "Guldbergsgade 29N, 2200 København N",
    image: "https://share.cederdorff.com/images/petl.jpg"
  },
  {
    name: "Rasmus Cederdorff",
    position: "Lektor",
    mail: "race@kea.dk",
    department: "KEA Digital",
    address: "Guldbergsgade 29N, 2200 København N",
    image: "https://kea.dk/slir/w180-c1x1/images/race.jpg"
  }
];
```


It's all objects & arrays!

The screenshot shows the homepage of the Danish Broadcasting Corporation (DR) website. At the top, there are navigation links for 'NYHEDER' (News), 'DRTV', 'DR LYD', 'KONTAKT DR', and a search bar. Below the navigation, there are six thumbnail images of TV shows: 'DR1: Løvens Hule', 'DR3: Nationens stærkeste', 'P1: LSD kælderen', 'DR LYD: Annas Margrethe', 'DR3: Du fucker med de forkerte', and 'A Very British Scandal'. A horizontal slider below these shows three news snippets: 'EU klager over Kinas hårde kurs over for Litauen' (5 MIN. SIDEN), 'Børn og skoleelever opfordres stadig til to ugentlige coronatest' (13 MIN. SIDEN), and 'England skrætter størstedelen af coronarestriktionerne fra i dag' (25 MIN. SIDEN). The main content area features a large image of medical supplies (a mask, a thermometer, a syringe, and a bottle of hand sanitizer) against a blue background. Below this image, a headline reads '15 lande bakker Danmark op: Danske soldater skal blive i Mali'. At the bottom, a red banner says 'Liveblog ...' and 'Regeringen har meldt genåbning - men ikke'.

The screenshot shows the website for the Vocational Academy in Copenhagen (Erhvervsakademi - København). The page title is 'ALLE UDDANNELSER UD FRA INTERESSE'. It features a grid of 12 portrait photos of students or professionals, each representing a different vocational program. The programs listed are: AUTOMATIONSTEKNOLOG, BYGGEKOORDINATOR, BYGGETEKNIKER, DATAMATIKER, DESIGNTEKNOLOG, ENTREPRENØRSKAB OG DESIGN, EL-INSTALLATOR, ENERGITEKNOLOG, IT-TEKNOLOG, KORT- OG LANDMÅLING, MULTIMEDIEDESIGNER, and VVS-INSTALLATOR. Each program name is followed by a small arrow pointing to the right.

Frontend (client)



<https://cederdorff.github.io/potter-app/>

JSON Data Source (Server)

A screenshot of a browser developer tools JSON viewer. It shows two tabs: "Raw" and "Parsed". The "Raw" tab displays the raw JSON data, and the "Parsed" tab displays the data as a hierarchical object. The JSON data represents the characters from the Harry Potter series, including their names, species, gender, house, birth dates, ancestry, eye and hair colors, wands, patronuses, and actors.

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

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

fetch(...)

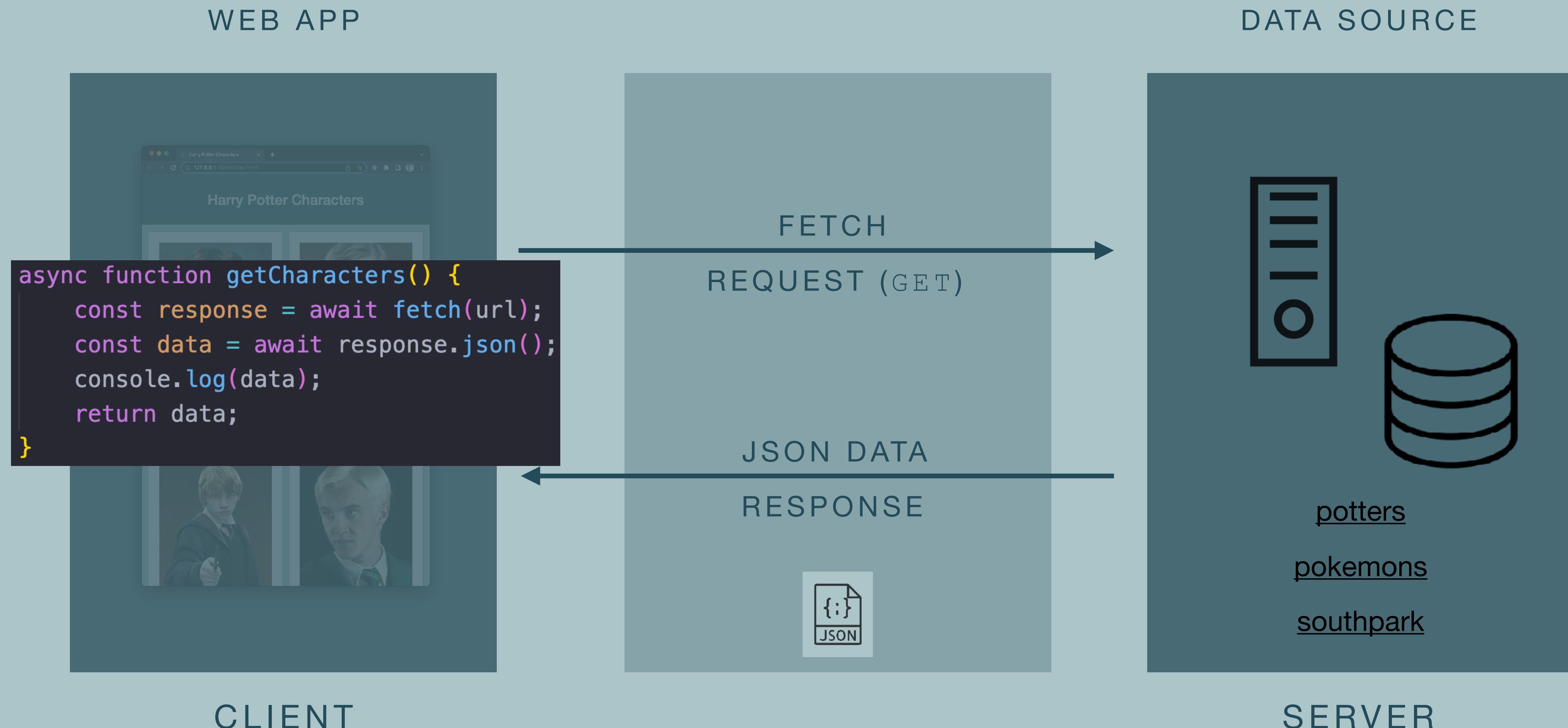
HTTP Requests in
JavaScript.

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

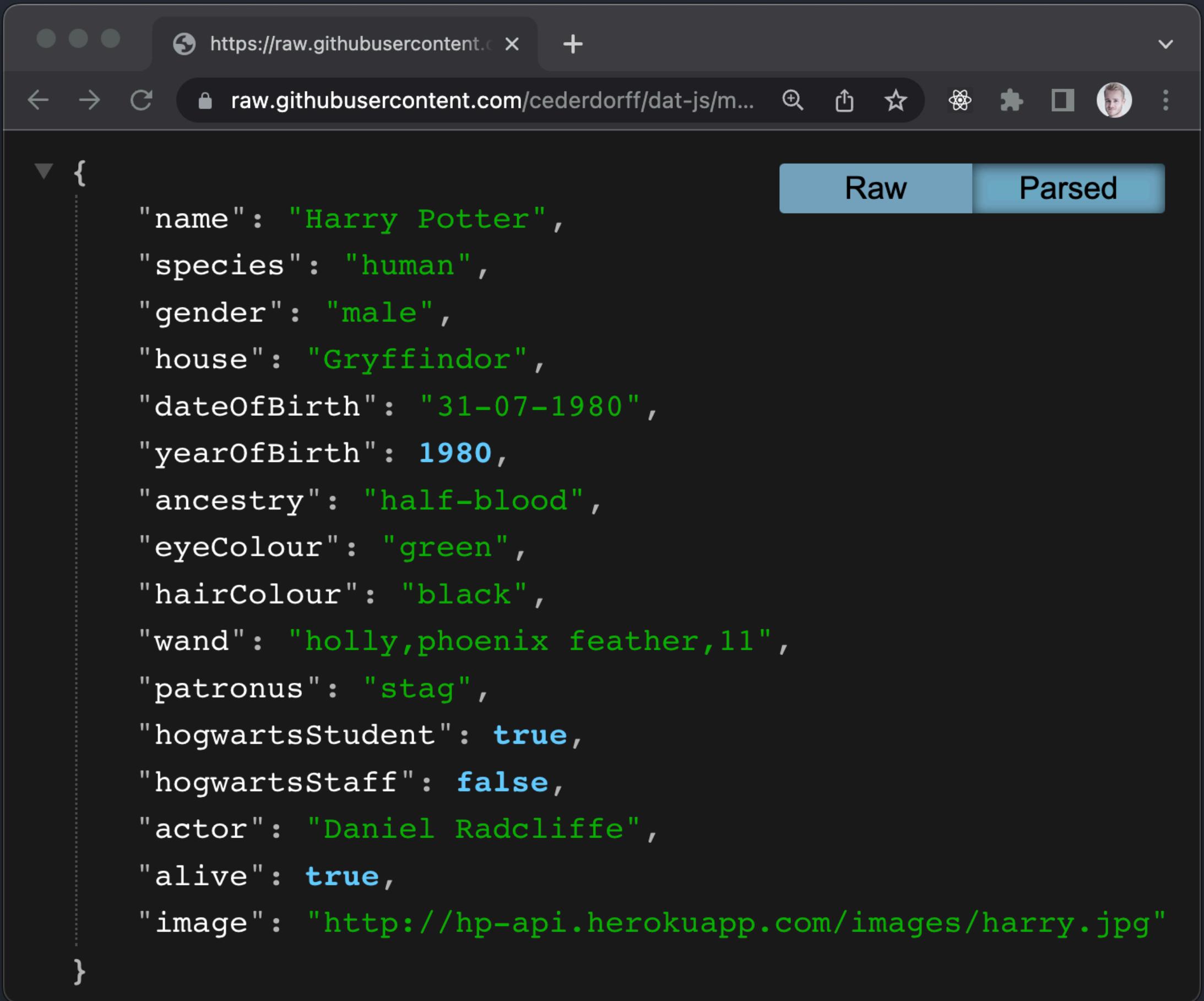
getCharacters fetches a list of characters
from a JSON data source, parses the JSON
to JS and returns the data.

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

Fetch, HTTP Request & Response



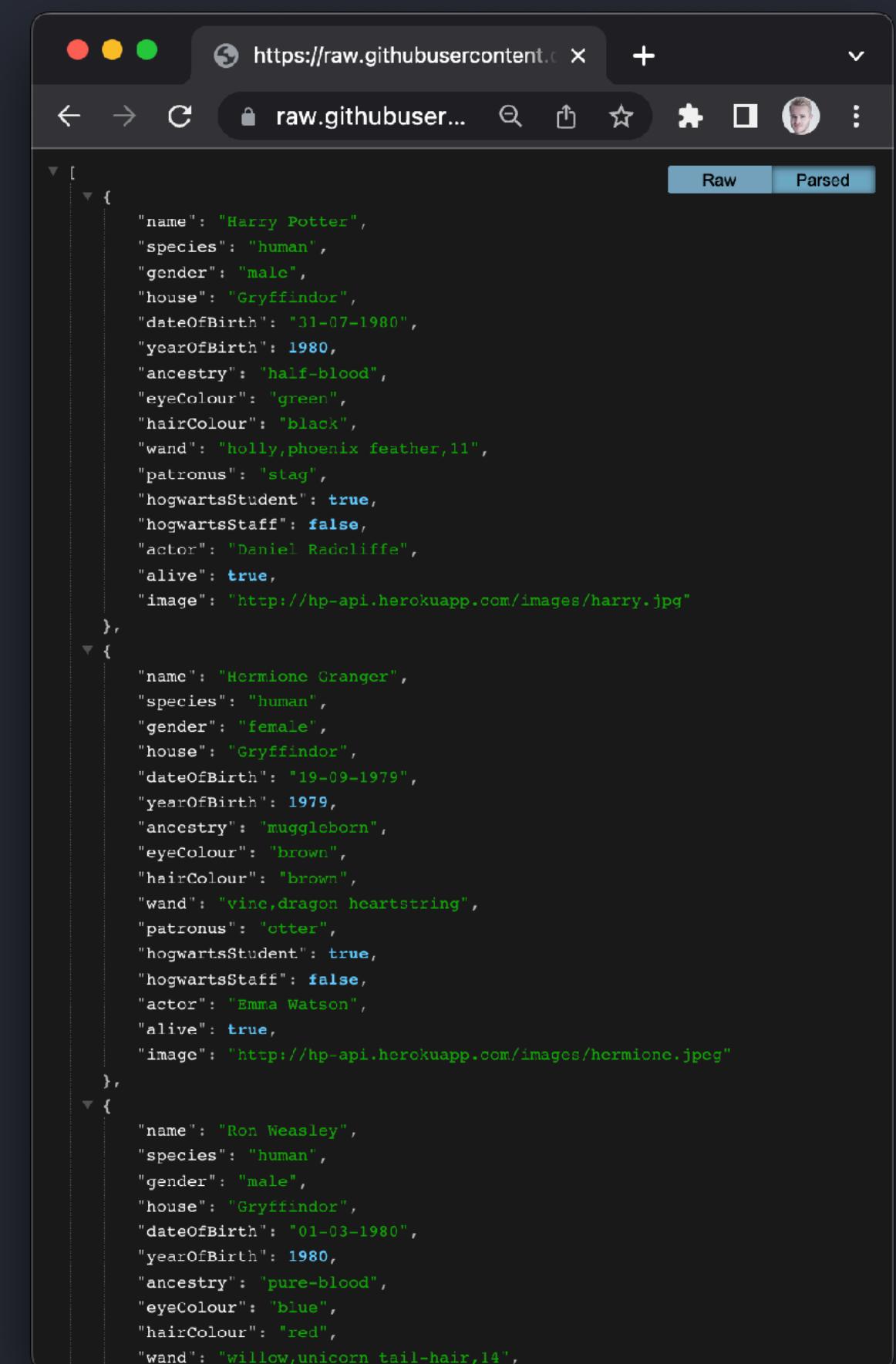
JSON



A screenshot of a web browser window displaying a single JSON object. The URL is <https://raw.githubusercontent.com/cederdorff/dat-js/master/data/harry.json>. The JSON object represents Harry Potter's character details. The 'Raw' tab is selected, showing the raw JSON code, while the 'Parsed' tab shows the same data as an expandable tree structure.

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

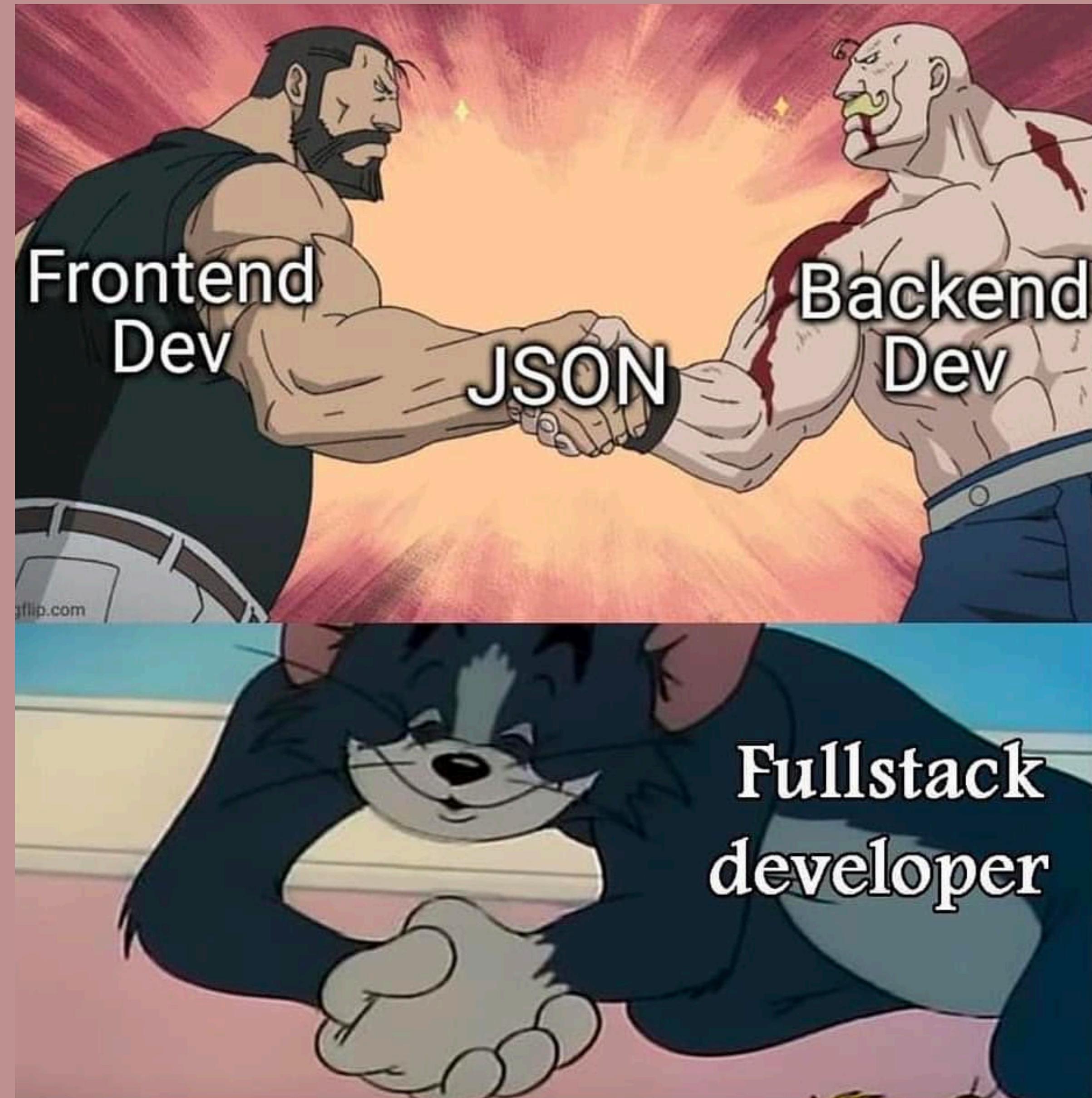
JSON OBJECT



A screenshot of a web browser window displaying a list of three JSON objects. The URL is <https://raw.githubusercontent.com/cederdorff/dat-js/master/data/characters.json>. The JSON array contains three objects: Harry Potter, Hermione Granger, and Ron Weasley. The 'Raw' tab is selected, showing the raw JSON code, while the 'Parsed' tab shows the data as an expandable tree structure.

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

LIST OF JSON OBJECTS



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

Frontend (client)



Så hvad kan du så nu?

<https://cederdorff.github.io/potter-app/>

JSON Data Source (Server)

A screenshot of a browser developer tools JSON viewer showing the raw JSON data for the Harry Potter characters. The JSON is an array of four objects, each representing a character with properties like name, species, gender, house, date of birth, year of birth, ancestry, eye colour, hair colour, wand, patronus, Hogwarts student status, staff status, actor, and alive status. Each object also includes a URL for the character's image.

```
[{"name": "Harry Potter", "species": "human", "gender": "male", "house": "Gryffindor", "dateOfBirth": "31-07-1980", "yearOfBirth": 1980, "ancestry": "half-blood", "eyeColour": "green", "hairColour": "black", "wand": "holly, phoenix feather, 11", "patronus": "stag", "hogwartsStudent": true, "hogwartsStaff": false, "actor": "Daniel Radcliffe", "alive": true, "image": "http://hp-api.herokuapp.com/images/harry.jpg"}, {"name": "Hermione Granger", "species": "human", "gender": "female", "house": "Gryffindor", "dateOfBirth": "19-09-1979", "yearOfBirth": 1979, "ancestry": "muggle-born", "eyeColour": "brown", "hairColour": "brown", "wand": "rowan, unicorn hair, 11", "patronus": "otter", "hogwartsStudent": true, "hogwartsStaff": false, "actor": "Emma Watson", "alive": true, "image": "http://hp-api.herokuapp.com/images/hermione.jpeg"}, {"name": "Ron Weasley", "species": "human", "gender": "male", "house": "Gryffindor", "dateOfBirth": "01-03-1980", "yearOfBirth": 1980, "ancestry": "pure-blood", "eyeColour": "blue", "hairColour": "red", "wand": "willow,unicorn tail-hair,11", "patronus": "Jack Russell terrier", "hogwartsStudent": true, "hogwartsStaff": false, "actor": "Rupert Grint", "alive": true, "image": "http://hp-api.herokuapp.com/images/ron.jpg"}, {"name": "Draco Malfoy", "species": "human", "gender": "male", "house": "Slytherin", "dateOfBirth": "15-07-1980", "yearOfBirth": 1980, "ancestry": "pure-blood", "eyeColour": "grey", "hairColour": "blonde", "wand": "ash, dragon heartstring, 12", "patronus": "owl", "hogwartsStudent": true, "hogwartsStaff": false, "actor": "Tom Felton", "alive": true, "image": "http://hp-api.herokuapp.com/images/draco.jpg"}]
```

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

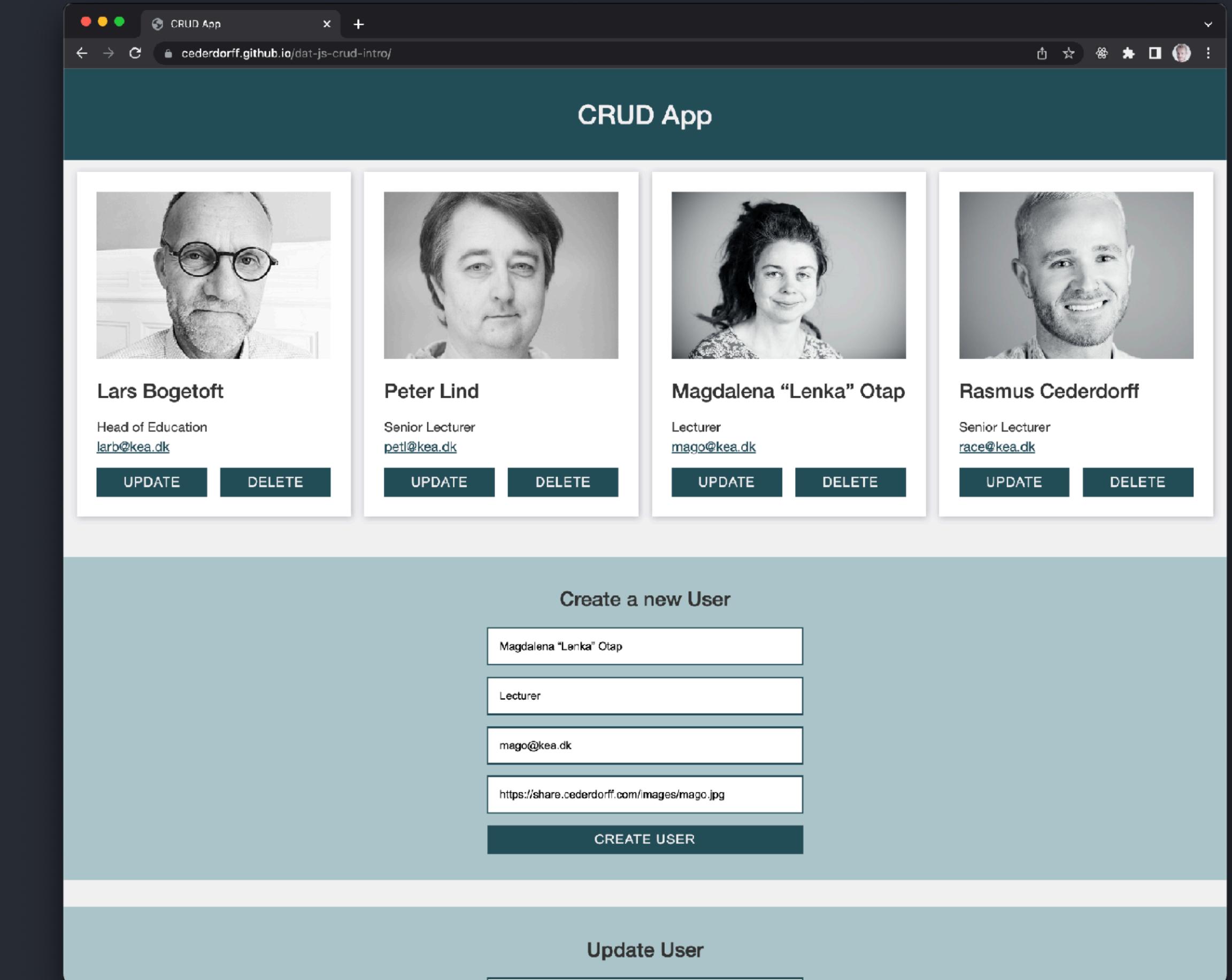
Og, hvor skal vi hen, du?



Forms, REST & CRUD

Formularer, API'er, REST,
Fetch, HTTP-metoder,
CRUD, fejlhåndtering &
validering

CRUD-projekt



<https://cederdorff.github.io/dat-js-crud-intro/>

Frontend (client)



Hvad mangler du?

Er der noget, du gerne ville kunne
have gjort med dit data-projekt, som
du ikke kunne?

<https://cederdorff.github.io/potter-app/>

JSON Data Source (Server)

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

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

Sortering, filtrering, søgning ...

Harry Potter Characters

Filter by house: All Search by name

Image	Name	House	Gender	Year of birth
	Harry Potter	Gryffindor	male	1980
	Hermione Granger	Gryffindor	female	1979
	Ron Weasley	Gryffindor	male	1980
	Draco Malfoy	Slytherin	male	1980

ALL USERS

Order by: Choose here Filter by: All Search

	Diana Riis Myjak Andersen eaadimy@students.eaaa.dk Enrollment: StudentEnrollment UPDATE DELETE
	German Arias Rodriguez eaagar@students.eaaa.dk Enrollment: StudentEnrollment UPDATE DELETE
	Haya Barakat eaahbar@students.eaaa.dk Enrollment: StudentEnrollment UPDATE DELETE

USERS CREATE RANDOM

Canvas Users

Show Students Show Teachers Search Search by name Sort by First name

	Anders Husted eaaahu@students.eaaa.dk Role: Student
	Anders Lassen Skrydstrup eaaanls@students.eaaa.dk Role: Student
	Aron Ingi Svansson eaaarsv@students.eaaa.dk Role: Student

Loops

Iterate over arrays or other iterable objects.

Ex loop through an array of objects.

```
// .forEach
characterList.forEach(showCharacter);

// for of loop
for (const character of characterList) {
    showCharacter(character);
}

// for loop
for (let index = 0; index < characterList.length; index++) {
    const character = characterList[index];
    showCharacter(character);
}
```

Loops

Loops can execute a block of code a number of times.

There are a few ways to do that in JavaScript 🤔

JavaScript supports different kinds of loops:

- `for` - loops through a block of code a number of times
- `for/in` - loops through the properties of an object
- `for/of` - loops through the values of an iterable object
- `while` - loops through a block of code while a specified condition is true
- `do/while` - also loops through a block of code while a specified condition is true

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

.forEach

an array method

```
const names = ["Peter", "Oskar", "Lenka", "Rasmus"];
names.forEach(showName);

function showName(name) {
  console.log(name);
}
```

You can use it on an array to do *something* for each element in the array.

forEach will call a function for each element in an array.

forEach only iterates over an array

You can only have one kind of "loop" with forEach, as the name implies:

one that does something for each item in an array

You can't make it stop half-way – or make it run a certain number of times – or even run without an array.

But other kinds of loops can!

For Loop

```
const names = ["Peter", "Lenka", "Oskar", "Rasmus"];

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

You can use it to loop over an array.

A for loop repeats until a specified condition is false.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Loops_and_iteration#for_statement
<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/for>
https://www.w3schools.com/js/js_loop_for.asp

for to the rescue

“all in one”

sets up a counter

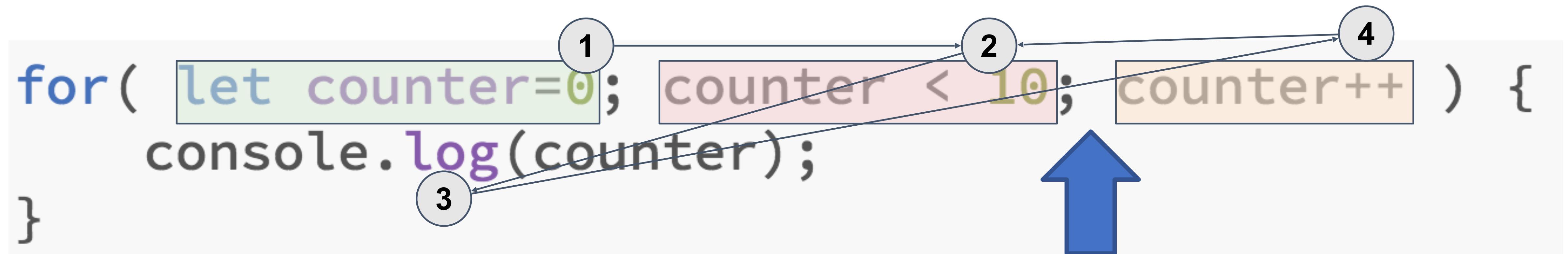
checks the counter

increments the counter

```
for( let counter=0; counter < 10; counter++ ) {  
    console.log(counter);  
}
```

for loop flow

It is a bit complicated



The important thing to remember is:

The code is executed *after* the check, but *before* the increment!

The scope is also a bit weird

What happens?

```
for( let counter=0; counter < 10; counter++ ) {  
    console.log(counter);  
}  
  
console.log(`after the loop, counter is ${counter}`);
```

This is the
block-scope of
the counter

Even though it was
created outside the
block ...

Try it!

For of Loop

```
const names = ["Peter", "Lenka", "Oskar", "Rasmus"];
for (const name of names) {
  console.log(name);
}
```

You can use it to loop over an array to do *something* for every element.

for of loops through the values of an iterable object (an array is an iterable object).

For of loop

Iterates over arrays or other iterable objects

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

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

For in

Loop

```
const person = {  
    name: "Peter",  
    mail: "petl@kea.dk",  
    age: 45  
};  
  
for (const key in person) {  
    console.log(key);  
    const value = person[key];  
    console.log(value);  
}
```

for in loops
through all
properties of an
Object.

While

Loop

```
const names = ["Peter", "Oskar", "Lenka", "Rasmus"];  
  
let index = 0;  
  
while (index < names.length) {  
    const name = names[index];  
    console.log(name);  
    index++;  
}
```

Loops executes a block of code as long as a specified condition is true.

The while loop loops through a block of code as long as a specified condition is true.

... while the index is less than the length of the array, execute...

Loops, iteration & Array-metoder

Jeg har samlet lidt
lækkert til jer 

Loops, iteration & Array-metoder

↳ 1 backlink

0. Projektskabelon

1. Loops & iteration

1.1. forEach

Gennemløb names

forEach og index

Gentag med years og teachers

DOM-Manipulation (ekstra)

1.2. For Loop

Gennemløb years

Numbers

Mere for træning (ekstra)

1.3. For Of Loop

Log teachers

Hvis age er over 40

Søgefunktion med for of

Mere for of træning (ekstra)

1.4. For In Loop

Gennemløb alle props i teacher

Hvorfor for in?

1.5. While Loop

Sålænge index er mindre end teachers.length

Hvis teacher er "Lecturer"

Søgefunktion med while

Mere while træning (ekstra)

2. Array-metoder

.filter

.find

.sort

0. Projektskabelon

- Clone eller fork følgende projekt fra GitHub:  [loops-filter-find-and-sort](#) 
- Undersøg projekt. Hvad har du? Hvad er der i JavaScript-filen?

Loops, iteration & Array-metoder

Array Reference

Array Methods

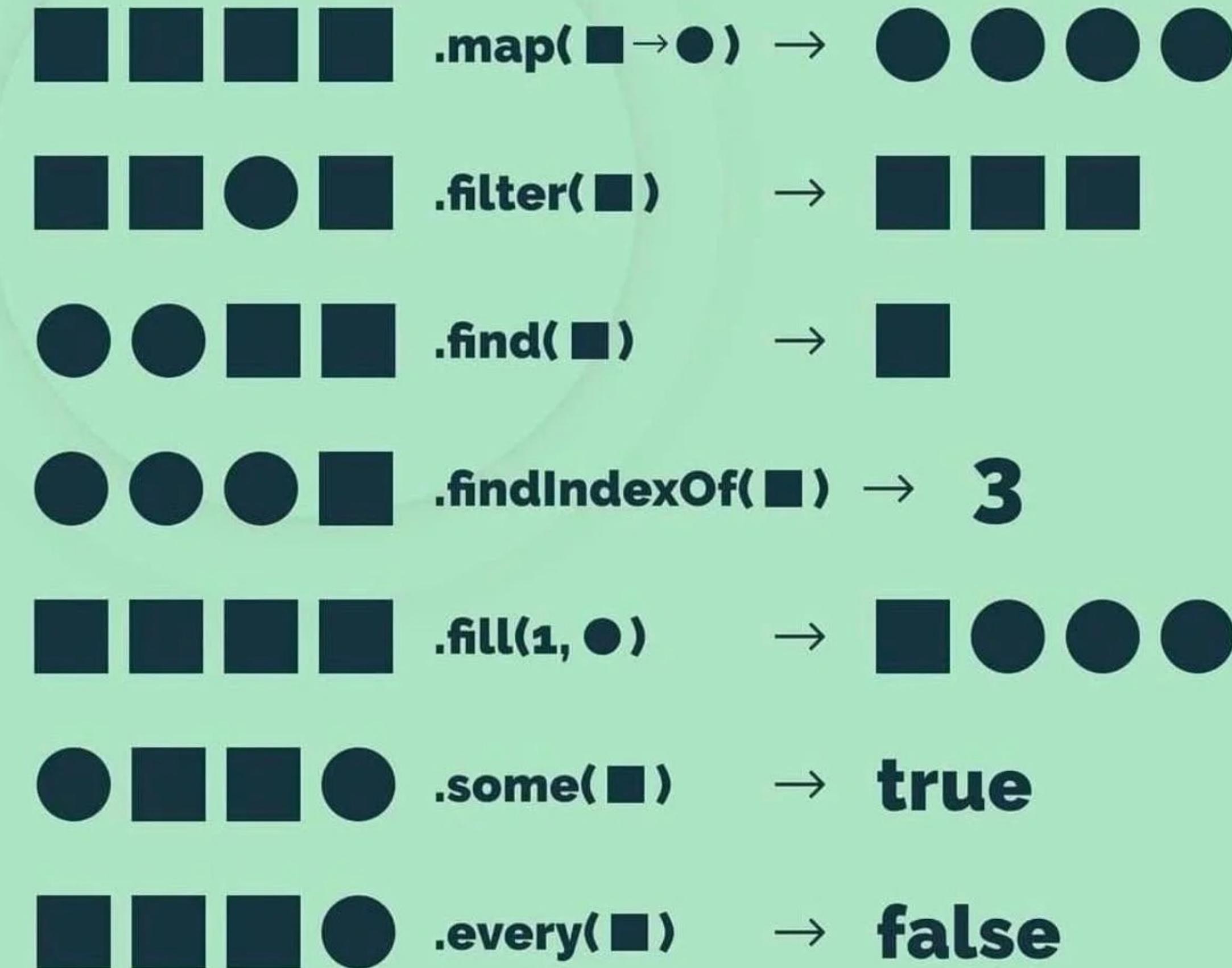
Der er mange, og vi skal nok vende tilbage til nogle af dem.

The screenshot shows a web browser window with the title "JavaScript Array Reference" from w3schools.com. The page lists various array methods and properties with their descriptions. On the left, there's a sidebar titled "JS Array" with a list of methods. The main content area has a table with columns for "Name" and "Description". A sidebar on the right promotes w3schools services like "Start DEVELOPING Without Installations" and "BACKEND Support Included".

Name	Description
concat()	Joins arrays and returns an array with the joined arrays
constructor	Returns the function that created the Array object's prototype
copyWithin()	Copies array elements within the array, to and from specified positions
entries()	Returns a key/value pair Array Iteration Object
every()	Checks if every element in an array pass a test
fill()	Fill the elements in an array with a static value
filter()	Creates a new array with every element in an array that pass a test
find()	Returns the value of the first element in an array that pass a test
findIndex()	Returns the index of the first element in an array that pass a test
flat()	Creates an array from an object
flatMap()	Check if an array contains the specified element
forEach()	Calls a function for each array element
from()	Search the array for an element and returns its position
includes()	Checks whether an object is an array
indexOf()	Joins all elements of an array into a string
join()	
keys()	
lastIndexOf()	
length	
map()	
pop()	
push()	
shift()	
slice()	
sort()	
splice()	
unshift()	

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

Array Methods



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

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

.filter()

Array Method

```
const users = [  
    { name: "John", age: 35 },  
    { name: "Pete", age: 40 },  
    { name: "Mary", age: 44 }  
];  
  
// returns an array with users over the age 39  
const results = users.filter(checkAge);  
  
function checkAge(user) {  
    return user.age > 39;  
}  
  
console.log(results);
```

.filter() creates a new array with elements that pass a test provided by a function.

The filter() method does not execute the function for empty elements.

The filter() method does not change the original array.

.find()

Array Method

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

// returns an object with the
// first matching user of the age 40
const result = users.find(matchAge);

function matchAge(user) {
  return user.age === 40;
}

console.log(result);
```

find() returns the value of the first element that passes a test.

find() executes a function for each array element.

find() returns undefined if no elements are found.

find() does not execute the function for empty elements.

find() does not change the original array.

.sort()

Array Method

```
const names = ["Peter", "Magdalena", "Frederikke", "Anders"];
names.sort(); // sorts the original array with strings
console.log(names);
```

```
const years = [2003, 2032, 1990, 1989, 1975, 2023, 1933];
years.sort(); // sorts the original array of numbers
console.log(years);
```

.sort() sorts the elements of an array.

.sort() overwrites the original array.

.sort() sorts the elements as strings in alphabetical and ascending order.

.sort()

Array Method

```
const users = [
    { name: "John", age: 35 },
    { name: "Pete", age: 40 },
    { name: "Mary", age: 44 }
];
// compare and sort by the age prop
users.sort(compareAge);

function compareAge(teacher1, teacher2) {
    return teacher1.age - teacher2.age;
}

// compare and sort by the name prop
users.sort(compareName);

function compareName(teacher1, teacher2) {
    return teacher1.name.localeCompare(teacher2.name);
}
```

If you need to sort an array of objects, you must provide a compare function.

The compare function defines how to compare the objects.

.sort() uses the compare function to sort the original array.

Code Every Day



```
while (alive) {  
    eat();  
    sleep();  
    code();  
    repeat();  
}
```