

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

JavaScript meeting Backend

Frontend Programming

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

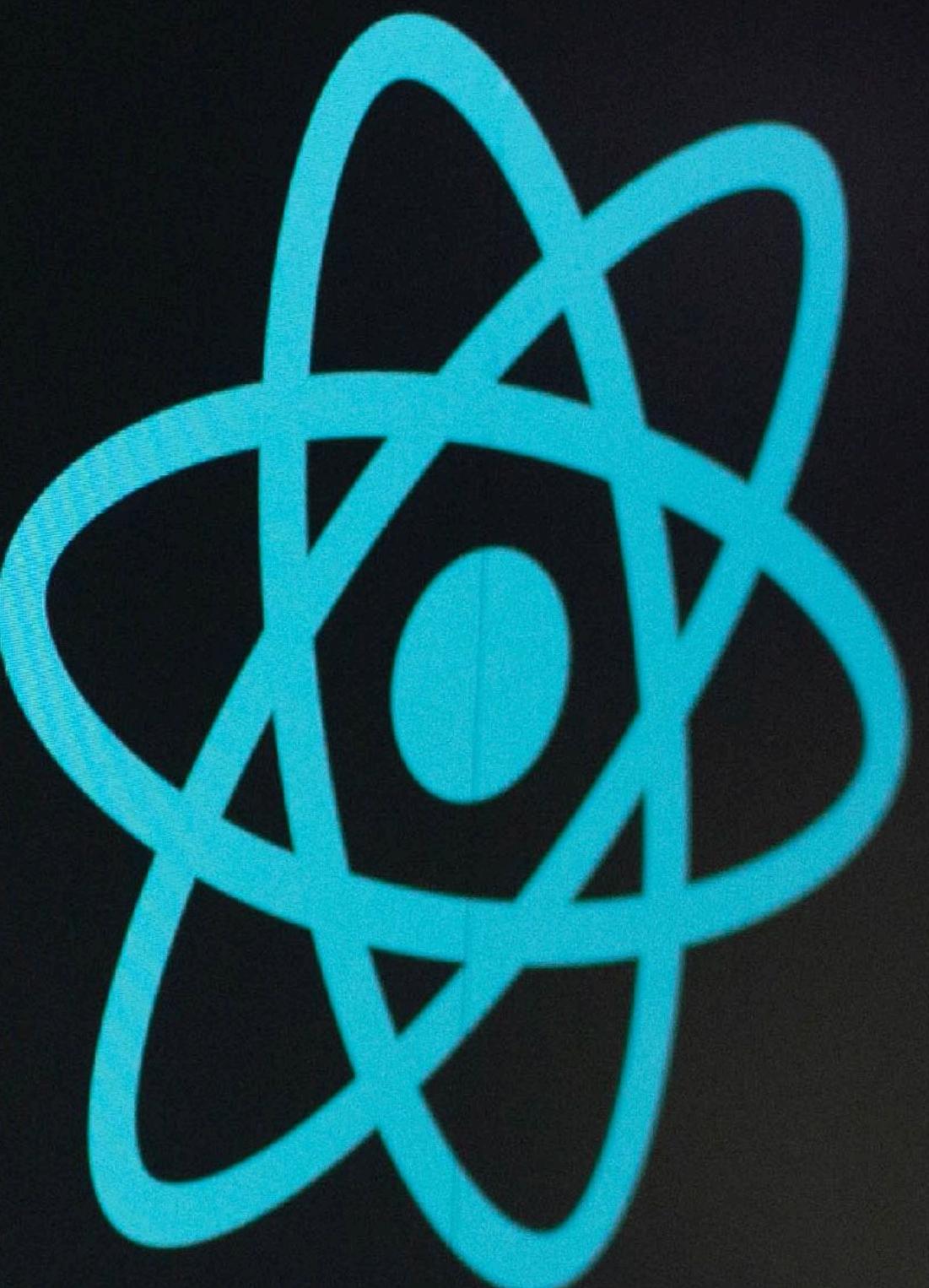


Uge 17				
	Mandag d. 25 - 04	Tirsdag d. 26 - 04	Onsdag d. 27 - 04	Fredag d. 29 - 04
08:30 - 10:00		Frontend programming RACE eaa-R104-1.16	Interaction and experience design SBJ eaa-R104-S.30	Frontend programming RACE KATO eaa-R104-1.16
10:30 - 12:00		Frontend programming RACE eaa-R104-1.16	Interaction and experience design SBJ eaa-R104-S.30	Backend programming RACE KATO eaa-R104-1.16
12:30 - 14:00		Frontend programming RACE eaa-R104-1.16	Interaction and experience design SBJ eaa-R104-S.30	Interdisciplinary project RACE SBJ KATO eaa-R104-1.16
14:15 - 15:45				
Uge 18				
	Mandag d. 02 - 05	Tirsdag d. 03 - 05	Onsdag d. 04 - 05	Torsdag d. 05 - 05
08:30 - 10:00	Interdisciplinary project	Interdisciplinary project	Interdisciplinary project	Interdisciplinary project
10:30 - 12:00	Interdisciplinary project	Interdisciplinary project	Interdisciplinary project	Interdisciplinary project
12:30 - 14:00	Interdisciplinary project	Interdisciplinary project	Interdisciplinary project	Interdisciplinary project
14:15 - 15:45				
Uge 19				
	Mandag d. 09 - 05	Tirsdag d. 10 - 05	Onsdag d. 11 - 05	Torsdag d. 12 - 05
08:30 - 10:00	Interdisciplinary project	Interdisciplinary project	Interdisciplinary project	Interdisciplinary project
10:30 - 12:00	Interdisciplinary project	Interdisciplinary project	Interdisciplinary project	Interdisciplinary project
12:30 - 14:00	Interdisciplinary project	Interdisciplinary project	Interdisciplinary project	Interdisciplinary project
14:15 - 15:45				Bededag
Uge 20				
	Mandag d. 16 - 05	Tirsdag d. 17 - 05	Onsdag d. 18 - 05	Torsdag d. 19 - 05
08:30 - 10:00	Interdisciplinary project	Interdisciplinary project	Interdisciplinary project	Interdisciplinary project
10:30 - 12:00	Interdisciplinary project	Interdisciplinary project	Interdisciplinary project	Interdisciplinary project
12:30 - 14:00	Interdisciplinary project	Interdisciplinary project	Interdisciplinary project	Interdisciplinary project
14:15 - 15:45				
Uge 21				
	Mandag d. 23 - 05	Tirsdag d. 24 - 05	Onsdag d. 25 - 05	Torsdag d. 26 - 05
08:30 - 10:00	Interdisciplinary project	Interdisciplinary project	Interdisciplinary project RACE SBJ KATO eaa-R104-S.30	Kristi himmelfart

Purpose

- How to use JavaScript to interact with a PHP Backend Service through a JSON API.
- Deeper knowledge of how to implement fetch with different HTTP methods using JavaScript, React and forms.
- Introduce you to authentication.

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



- React CRUD App Wrap Up
- Demo of Full-stack solution
- `firebase.replaceWith(KATO);`
- Set up your own Full-stack solution
- Interdisciplinary Project Kick Off.

Agenda

localhost:3000

React Firebase REST Post App

<https://cederdorff.github.io/react-cdn-starters/react-cdn-firebase-rest-post-app/>

POSTS CREATE

Morten Algy Bonderup
Senior Lecturer



qui est esse

est rerum tempore vitae sequi sint nihil reprehenderit dolor beatae ea dolores neque fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis qui aperiam non debitibus possimus qui neque nisi nulla

Dan Okkels Brendstrup
Lecturer



consequuntur deleniti eos quia temporibus ab aliquid at

voluptatem cumque tenetur consequatur expedita ipsum nemo quia explicabo aut eum minima consequatur tempore cumque quae est et et in consequuntur voluptatem voluptates aut

Kim Elkjær Marcher-Jepsen
Senior Lecturer



at nam consequatur ea labore ea harum

cupiditate quo est a modi nesciunt soluta ipsa voluptas error itaque dicta in autem qui minus magnam et distinctio eum accusamus ratione error aut

Birgitte Kirk Iversen
Senior Lecturer



doloremque illum aliquid sunt

Jes Arbov
Lecturer



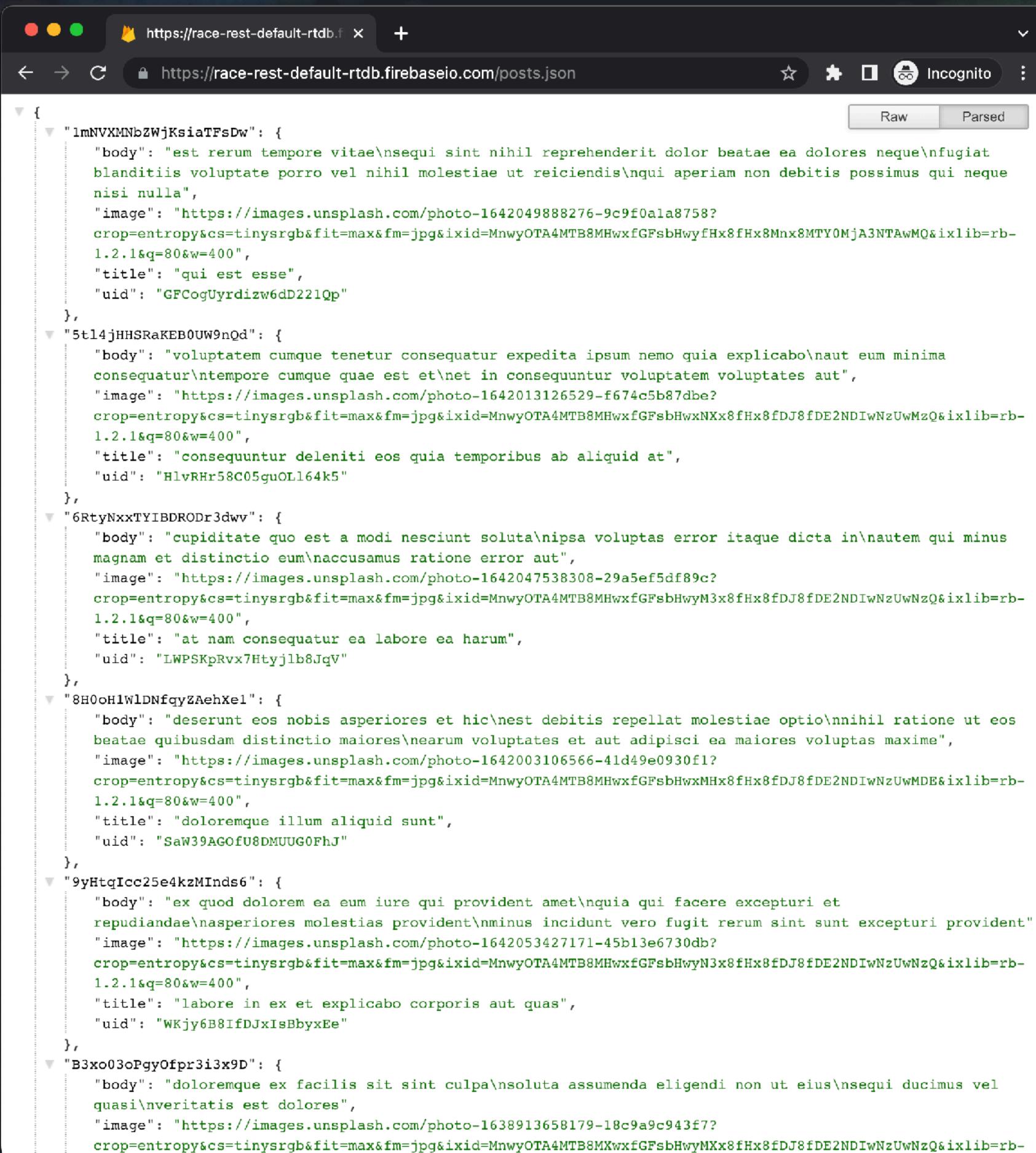
labore in ex et explicabo corporis aut quas

Maria Louise Bendixen
Senior Lecturer



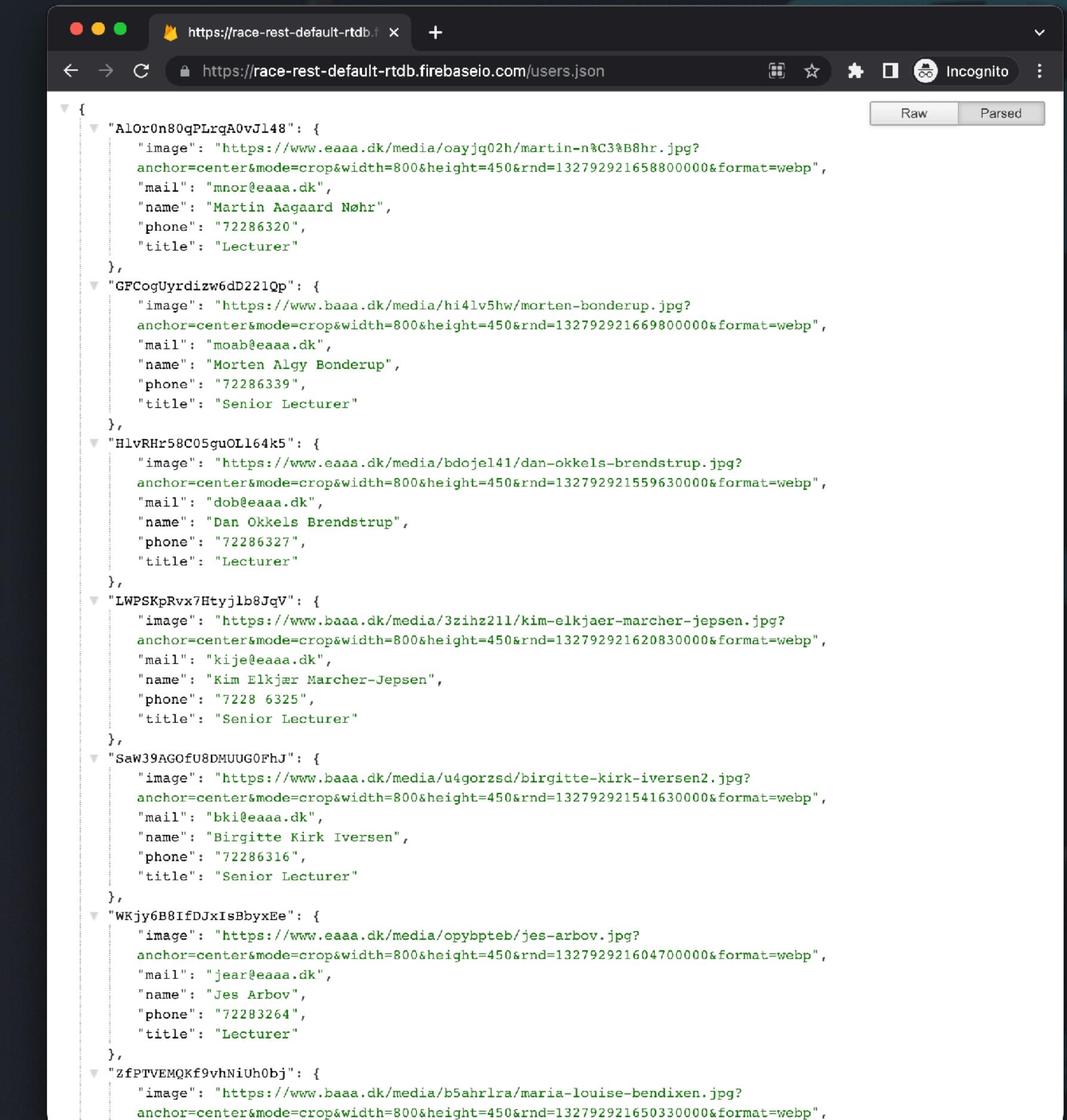
laboriosam dolor voluptates

Two data sources



```
Raw Parsed
{
  "1mNVXMMNbZWjKsiaTFsDw": {
    "body": "est rerum tempore vitae\\nsequi sint nihil reprehenderit dolor beatae ea dolores neque\\nfugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis\\nqui aperiam non debitis possimus qui neque nisi nulla",
    "image": "https://images.unsplash.com/photo-1642049888276-9c9f0ala8758?crop=entropy&cs=tinysrgb&fit=max&fm=jpg&ixid=MnwyOTA4MTB8MHwxfgFsbHwyfHx8fHx8Mnx8MTY0MjA3NTAwMQ&ixlib=rb-1.2.1&q=80&w=400",
    "title": "qui est esse",
    "uid": "GFCogUyrdizw6dD221Qp"
  },
  "5t14jHHSRAKEB0UW9npd": {
    "body": "voluptatem cumque tenetur consequatur expedita ipsum nemo quia explicabo\\naut eum minima consequatur\\ntempore cumque quae est et\\net in consequuntur voluptatem voluptates aut",
    "image": "https://images.unsplash.com/photo-1642013126529-f674c5b87dbe?crop=entropy&cs=tinysrgb&fit=max&fm=jpg&ixid=MnwyOTA4MTB8MHwxfgFsbHwxNx8fHx8fDJ8fDE2NDIwNzUwMzQ&ixlib=rb-1.2.1&q=80&w=400",
    "title": "consequuntur deleniti eos quia temporibus ab aliquid at",
    "uid": "HlvRHR58C05guOLl64k5"
  },
  "6RtyNxxTYIBDRDr3dwv": {
    "body": "cupiditate quo est a modi nesciunt soluta\\nipsa voluptas error itaque dicta in\\nautem qui minus magnam et distinctio eum\\naccusamus ratione error aut",
    "image": "https://images.unsplash.com/photo-1642047538308-29a5ef5df89c?crop=entropy&cs=tinysrgb&fit=max&fm=jpg&ixid=MnwyOTA4MTB8MHwxfgFsbHwyM3x8fHx8fDJ8fDE2NDIwNzUwNzQ&ixlib=rb-1.2.1&q=80&w=400",
    "title": "at nam consequatur ea labore ea harum",
    "uid": "LWPSKpRvx7Htyjl8JqV"
  },
  "8H0oH1wLDNfqryZAehXei": {
    "body": "deserunt eos nobis asperiores et hic\\nest debitis repellat molestiae optio\\nnihil ratione ut eos beatae quibusdam distinctio maiores\\nearum voluptates et aut adipisci ea maiores voluptas maxime",
    "image": "https://images.unsplash.com/photo-1642003106566-41d49e0930f1?crop=entropy&cs=tinysrgb&fit=max&fm=jpg&ixid=MnwyOTA4MTB8MHwxfgFsbHwxMHx8fHx8fDJ8fDE2NDIwNzUwMDE&ixlib=rb-1.2.1&q=80&w=400",
    "title": "doloremque illum aliquid sunt",
    "uid": "SaW39AGOfU8DMUUG0PhJ"
  },
  "9yHtqIcc25e4kzMInds6": {
    "body": "ex quod dolorum ea eum iure qui provident amet\\nquia qui facere excepturi et repudiandae\\nasperiores molestias provident\\nminus incidente vero fugit rerum sint sunt excepturi provident",
    "image": "https://images.unsplash.com/photo-1642053427171-45b13e6730db?crop=entropy&cs=tinysrgb&fit=max&fm=jpg&ixid=MnwyOTA4MTB8MHwxfgFsbHwyN3x8fHx8fDJ8fDE2NDIwNzUwNzQ&ixlib=rb-1.2.1&q=80&w=400",
    "title": "labore in ex et explicabo corporis aut quas",
    "uid": "WKjy6B8IfDJxIsBbyxEe"
  },
  "B3x03oPgyOfpr3i3x9D": {
    "body": "doloremque ex facilis sit sint culpa\\nsoluta assumenda eligendi non ut eius\\nsequi ducimus vel quasi\\nveritatis est dolores",
    "image": "https://images.unsplash.com/photo-1638913658179-18c9a9c943f7?crop=entropy&cs=tinysrgb&fit=max&fm=jpg&ixid=MnwyOTA4MTB8MXwxfgFsbHwyMxx8fHx8fDJ8fDE2NDIwNzUwNzQ&ixlib=rb-1.2.1&q=80&w=400"
  }
}
```

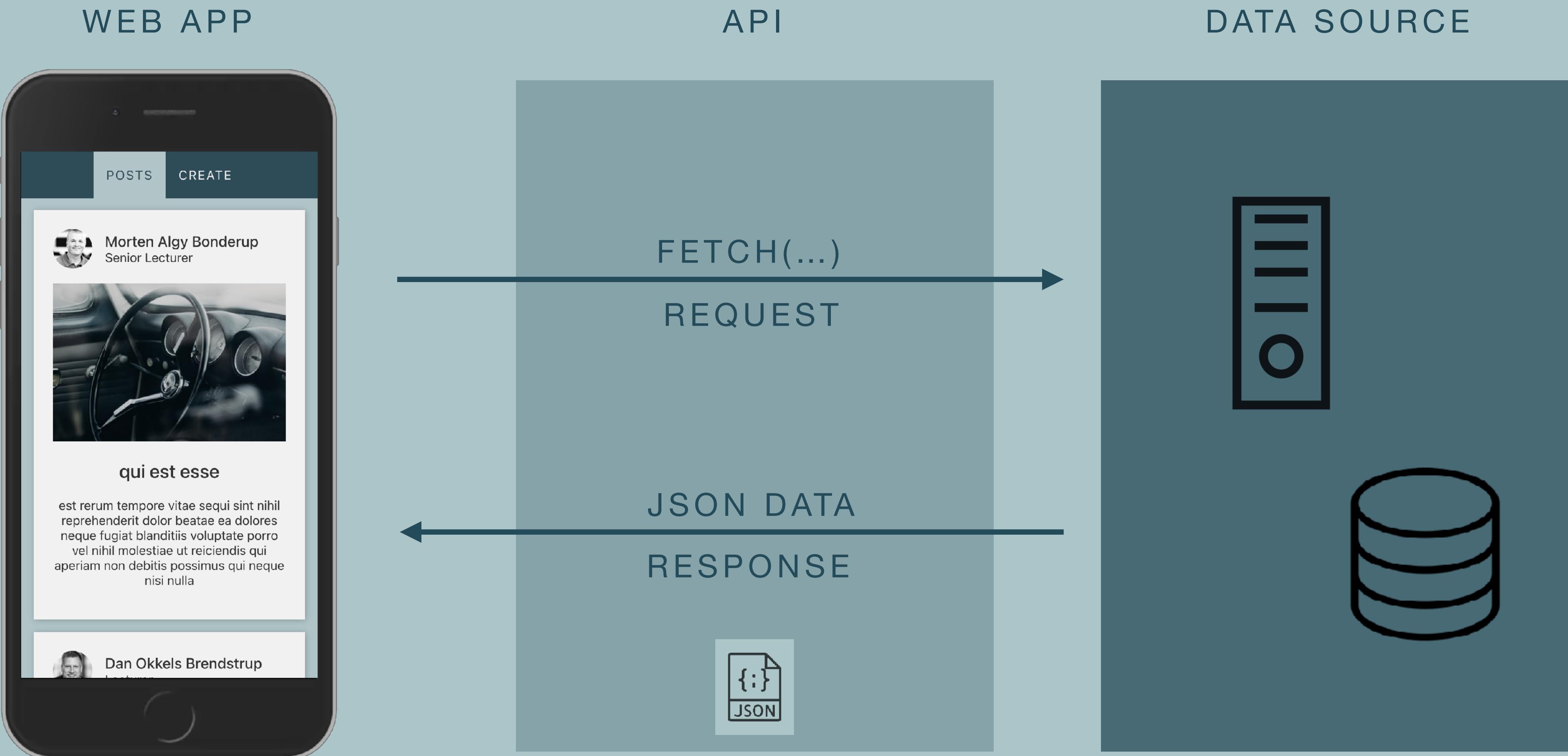
race-rest-default-rtdb.firebaseio.com/posts.json



```
Raw Parsed
{
  "AlOr0n80qPLrqA0vJ148": {
    "image": "https://www.eaaa.dk/media/oayjq02h/martin-n%C3%B8hr.jpg?anchor=center&mode=crop&width=800&height=450&rnd=132792921658800000&format=webp",
    "mail": "mnor@eaaa.dk",
    "name": "Martin Aagaard Nøhr",
    "phone": "72286320",
    "title": "Lecturer"
  },
  "GFCogUyrdizw6dD221Qp": {
    "image": "https://www.baaa.dk/media/hi4lv5hw/morten-bonderup.jpg?anchor=center&mode=crop&width=800&height=450&rnd=132792921669800000&format=webp",
    "mail": "moab@eaaa.dk",
    "name": "Morten Algy Bonderup",
    "phone": "72286339",
    "title": "Senior Lecturer"
  },
  "HlvRHR58C05guOLl64k5": {
    "image": "https://www.eaaa.dk/media/bdoje141/dan-okkels-brendstrup.jpg?anchor=center&mode=crop&width=800&height=450&rnd=132792921559630000&format=webp",
    "mail": "dob@eaaa.dk",
    "name": "Dan Okkels Brendstrup",
    "phone": "72286327",
    "title": "Lecturer"
  },
  "LWPSKpRvx7Htyjl8JqV": {
    "image": "https://www.baaa.dk/media/3zihz211/kim-elkjær-marcher-jepsen.jpg?anchor=center&mode=crop&width=800&height=450&rnd=132792921620830000&format=webp",
    "mail": "kije@eaaa.dk",
    "name": "Kim Elkjær Marcher-Jepsen",
    "phone": "7228 6325",
    "title": "Senior Lecturer"
  },
  "SaW39AGOfU8DMUUG0PhJ": {
    "image": "https://www.baaa.dk/media/u4gorzsdbirgitte-kirk-iversen2.jpg?anchor=center&mode=crop&width=800&height=450&rnd=132792921541630000&format=webp",
    "mail": "bki@eaaa.dk",
    "name": "Birgitte Kirk Iversen",
    "phone": "72286316",
    "title": "Senior Lecturer"
  },
  "WKjy6B8IfDJxIsBbyxEe": {
    "image": "https://www.eaaa.dk/media/opybpceb/jes-arbov.jpg?anchor=center&mode=crop&width=800&height=450&rnd=132792921604700000&format=webp",
    "mail": "jear@eaaa.dk",
    "name": "Jes Arbov",
    "phone": "72283264",
    "title": "Lecturer"
  },
  "ZfPTVEMQKF9vhNiUh0bj": {
    "image": "https://www.baaa.dk/media/b5ahrlra/maria-louise-bendixen.jpg?anchor=center&mode=crop&width=800&height=450&rnd=132792921650330000&format=webp",
    "mail": "mlb@eaaa.dk"
  }
}
```

race-rest-default-rtdb.firebaseio.com/users.json

Web Development



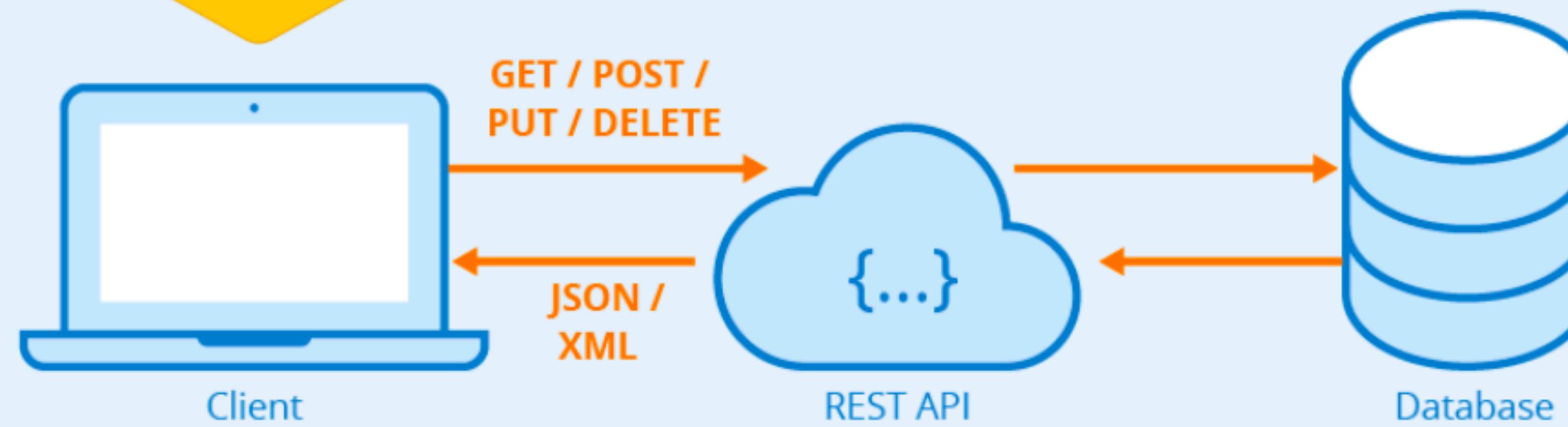
When someone asks you how to get data from a database in a js code



made with mematic

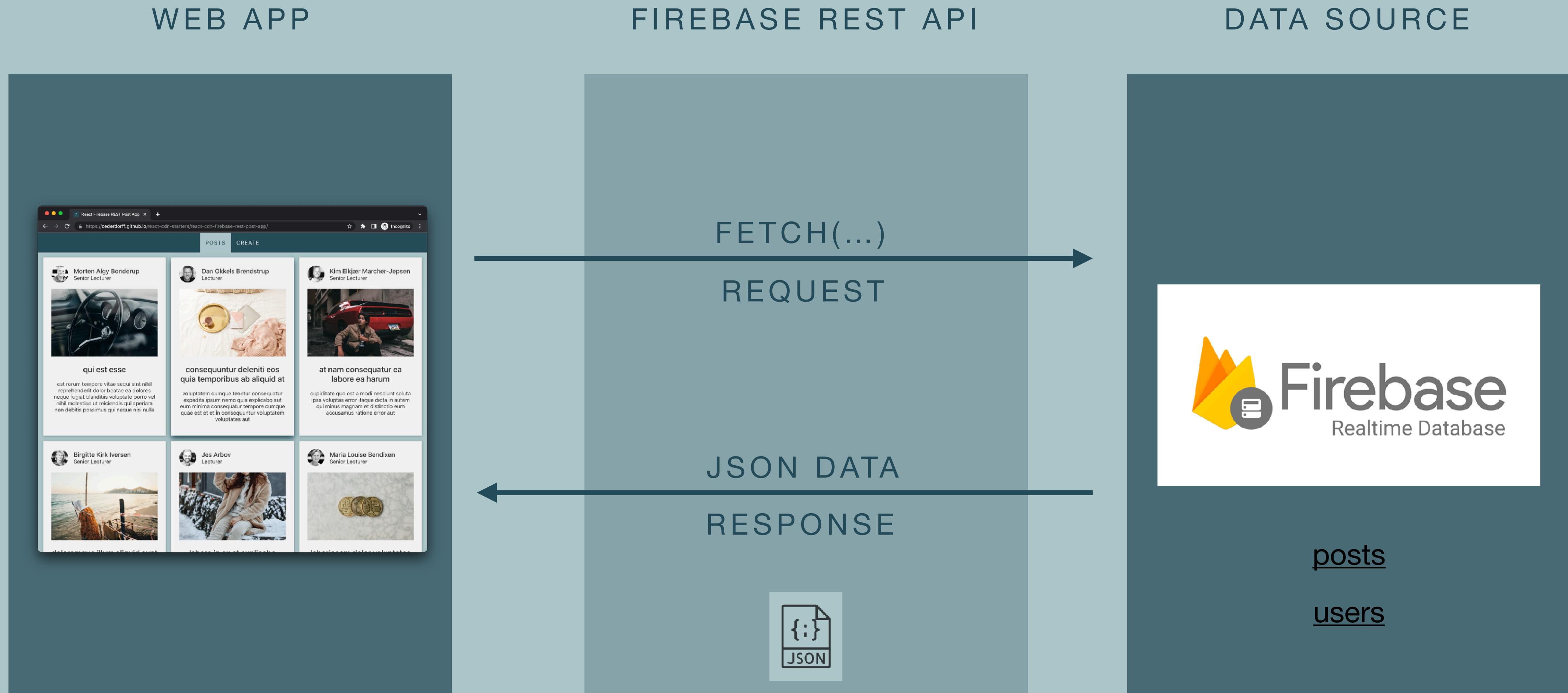


Firebase





Web Development

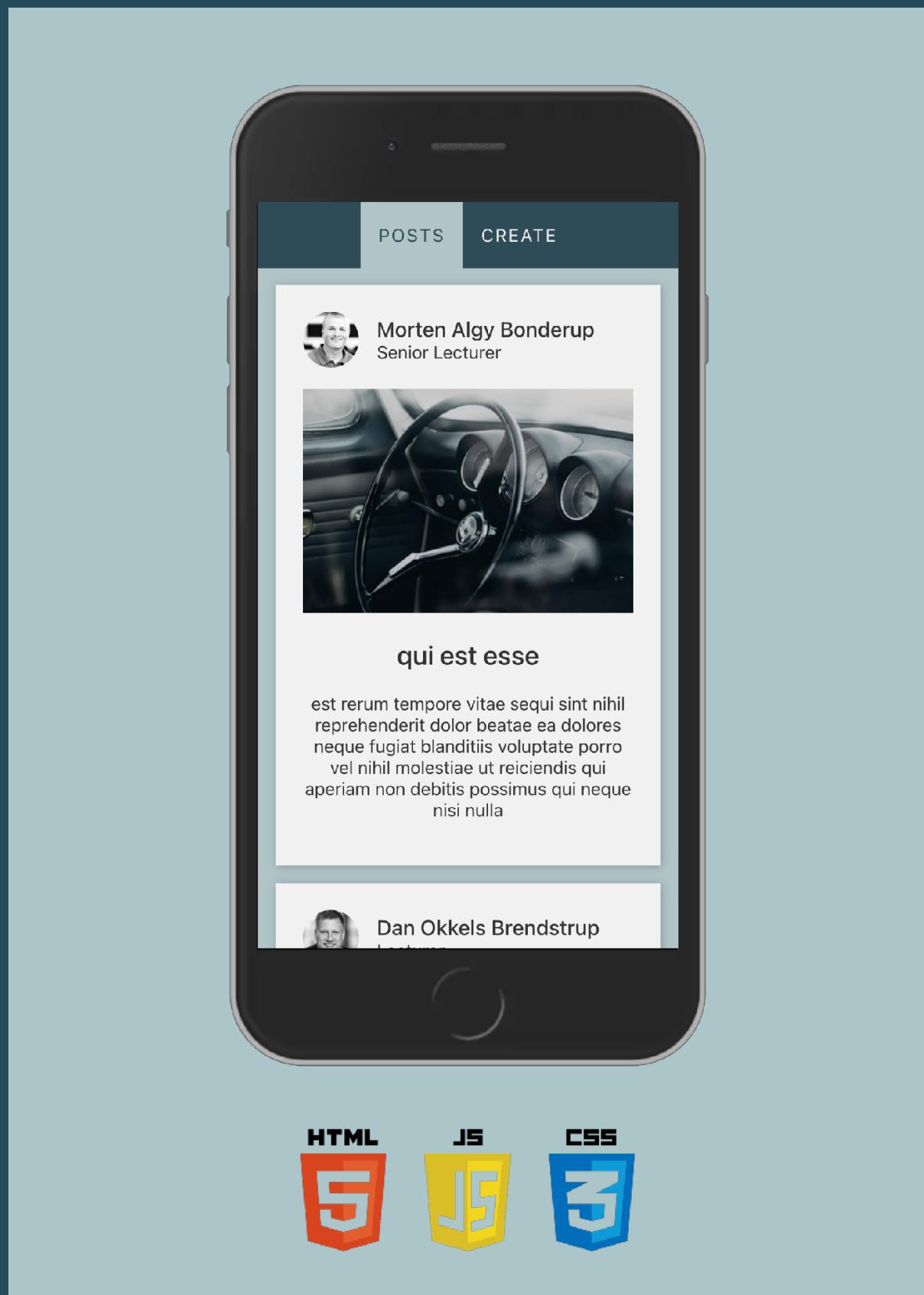


Interdisciplinary project

FRONTEND

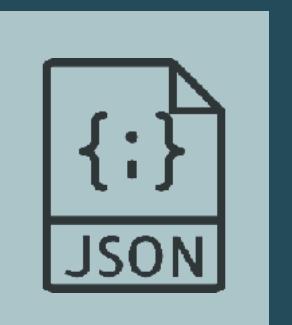
(API)

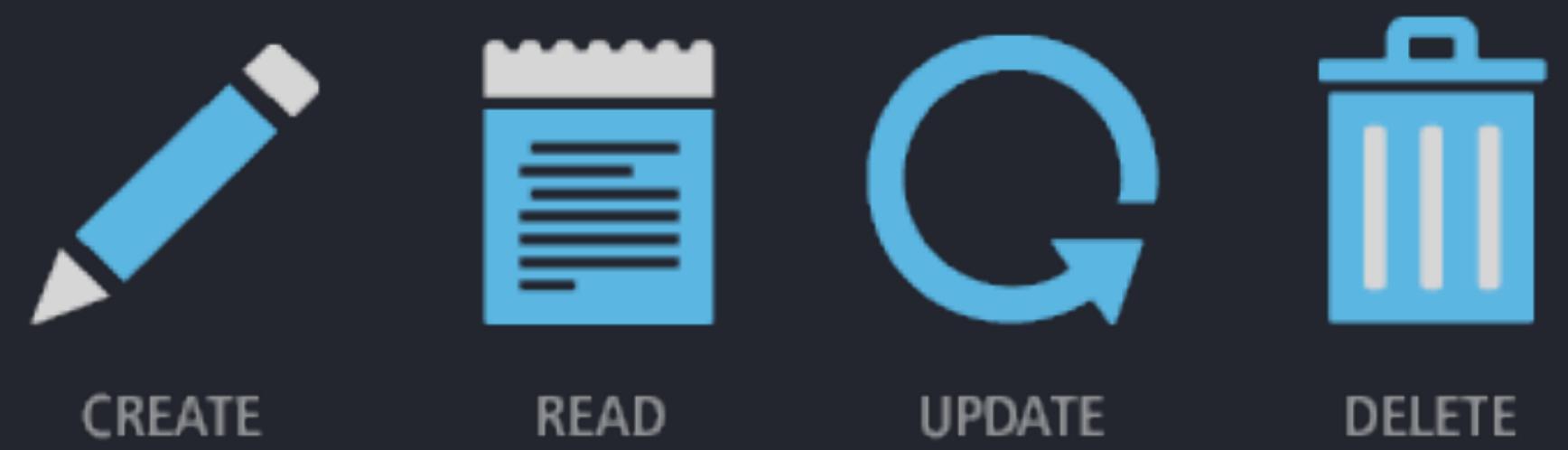
BACKEND



FETCH(...)
REQUEST

JSON DATA
RESPONSE





C R U D

What's CRUD?

- CREATE objects like a post, user, movie, product, etc.
- READ objects like an array (or object) of objects (posts, users, movies, products, etc)
- UPDATE an object, often given by a unique id.
- DELETE an object, often given by a unique id.

What's REST?

GET

POST

PUT

DELETE

- REpresentational State Transfer
- A standard for systems (client & server) to communicate over HTTP in order to retrieve or modify (data) resources.
- Stateless, meaning the two systems doesn't need to know anything about the state.
- The client makes the requests using the 4 basic HTTP verbs to define the operation.

HTTP REQUEST METHODS (verbs)

GET - POST - PUT - DELETE

HTTP (Hypertext Transfer Protocol) is the standard way to communicate between clients and servers (request-response protocol).

"HTTP defines a set of **request methods** to indicate the desired action to be performed for a given resource."

https://www.w3schools.com/tags/ref_httpmethods.asp

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods>

CRUD vs REST & HTTP Verbs

CREATE -> POST: create a new resource (object)

READ -> GET: retrieve a specific resource or a collection

UPDATE -> PUT: update a specific resource (by id)

DELETE -> DELETE: remove a specific resource by id

Fetch and read posts

GET is the default request method for fetch.

```
async function getPosts() {  
  const url = "https://race-rest-default-rtbd.firebaseio.com/posts.json";  
  const response = await fetch(url);  
  const data = await response.json();  
  const postsArray = Object.keys(data).map(key => ({ id: key, ...data[key] })); // from object to array  
  return postsArray  
}
```

Fetch & useEffect in React

In useEffect, perform the sideeffect (fetch) to get data from a data source.
And save the response data in a state using useState.

```
function PostsPage() {
  const [posts, setPosts] = useState([]);

  useEffect(() => {
    async function getPosts() {
      const url = "https://raw.githubusercontent.com/cederdorff/web-frontend/main/data/posts.json";
      const response = await fetch(url);
      const data = await response.json();
      setPosts(data);
    }
    getPosts();
  }, []);

  return (
    <section className="page">
```

REQUEST headers

“[...] contain more information about the resource to be fetched, or about the client requesting the resource.”

"A request header is an HTTP header that can be used in an HTTP request to provide information about the request context, so that the server can tailor the response. For example, the Accept-* headers indicate the allowed and preferred formats of the response. Other headers can be used to supply authentication credentials (e.g. Authorization), to control caching, or to get information about the user agent or referrer, etc."

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>

https://developer.mozilla.org/en-US/docs/Glossary/Request_header

REQUEST body

When making HTTP requests we sometimes need to send data.

The data is wrapped inside of the request body.

The request body is one of the following:

a string (often JSON encoded string with data, object, arrays, etc.)

form data (form/multipart)

blob/ buffer source - binary data

URL search params (x-www-form-urlencoded)

<https://javascript.info/fetch#post-requests>

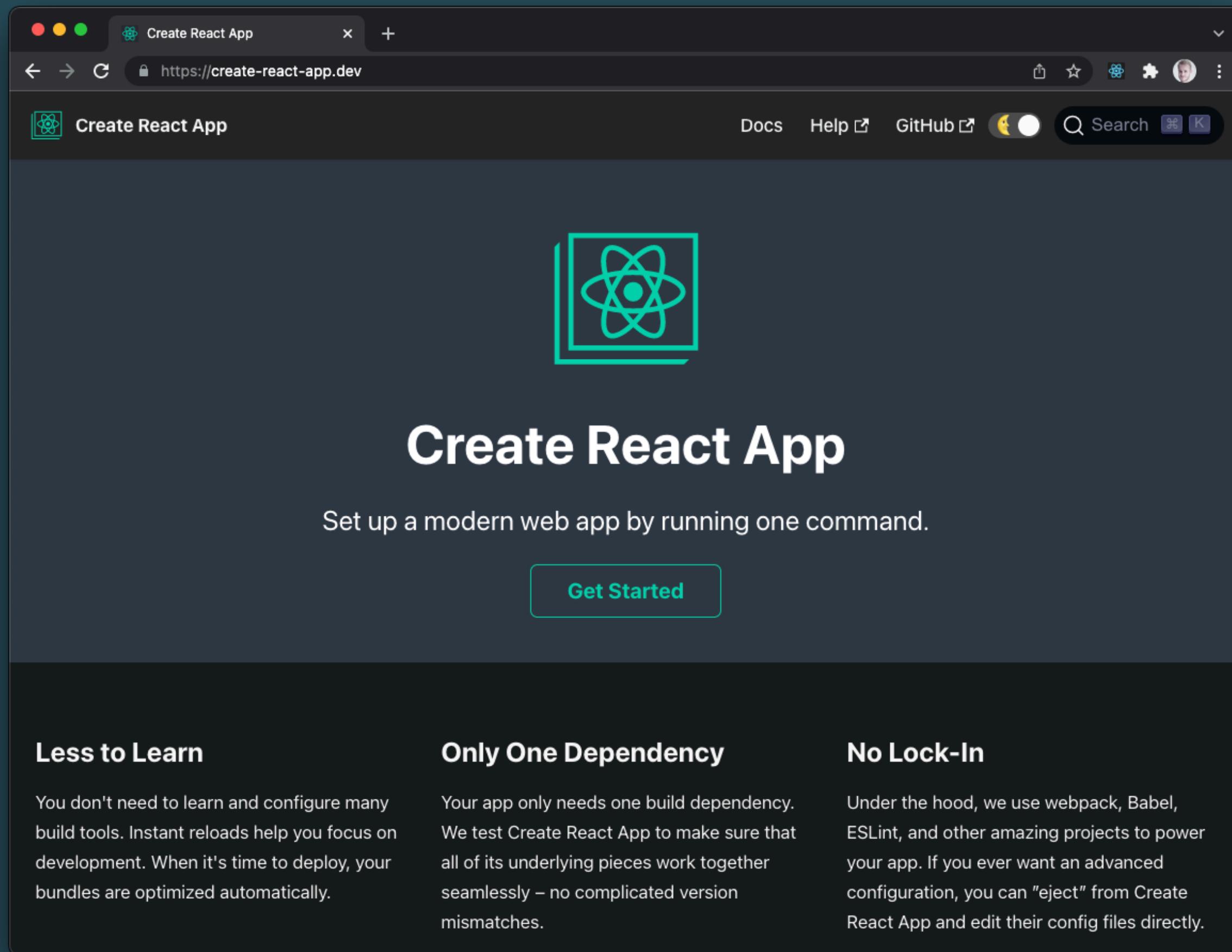
REQUEST body

```
const url = "http://localhost:3000/json-api/posts/";
await fetch(url, {
  method: "POST",
  body: JSON.stringify(newPost)
});
```

```
async function savePost(postToUpdate) {
  const response = await fetch(url, {
    method: "PUT",
    body: JSON.stringify(postToUpdate)
  });
  const data = await response.json();
  console.log(data);
  navigate("/");
}
```

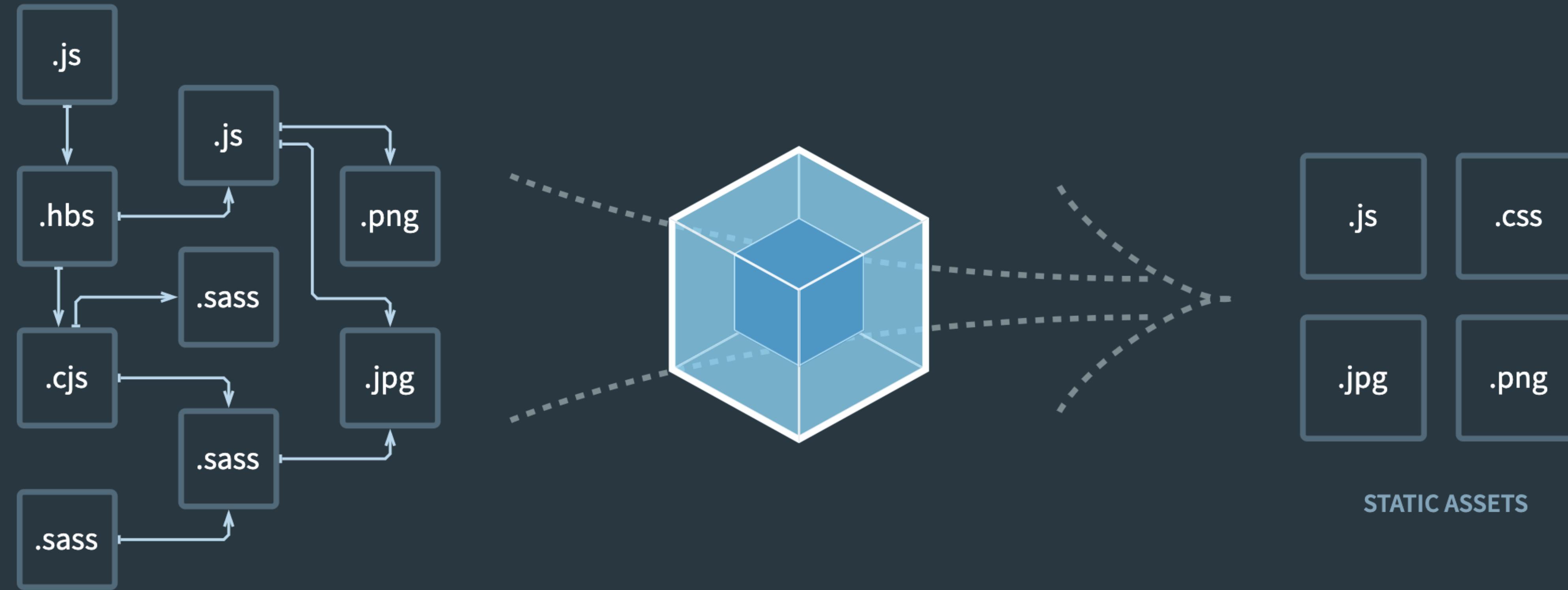
create-react-app

Starter template for your react app



Generate create-react-app

bundle your scripts

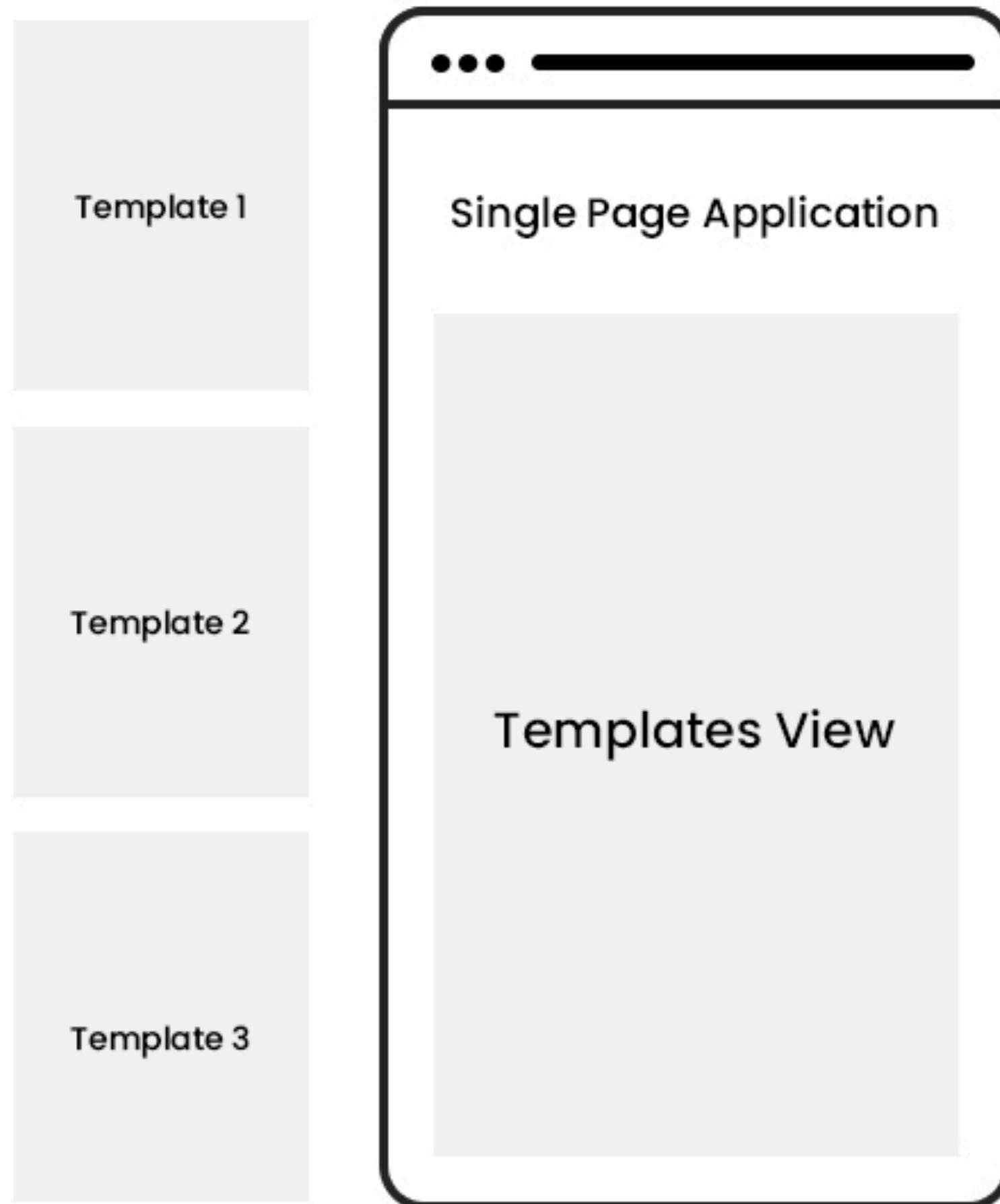


Single Page Apps

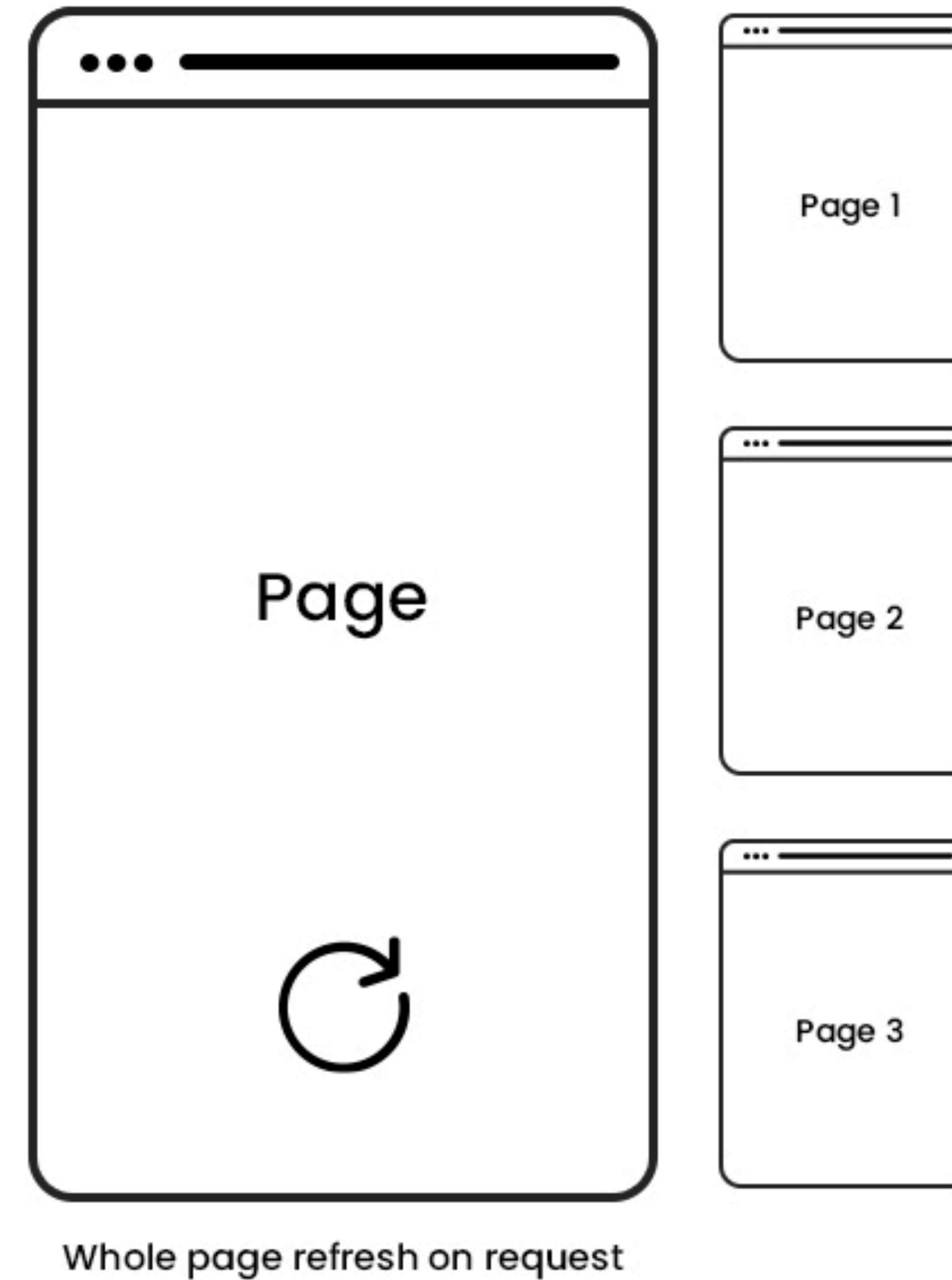
“A single-page application is an application that loads a single HTML page and all the necessary assets (such as JavaScript and CSS) required for the application to run. Any interactions with the page or subsequent pages do not require a round trip to the server which means the page is not reloaded.”

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

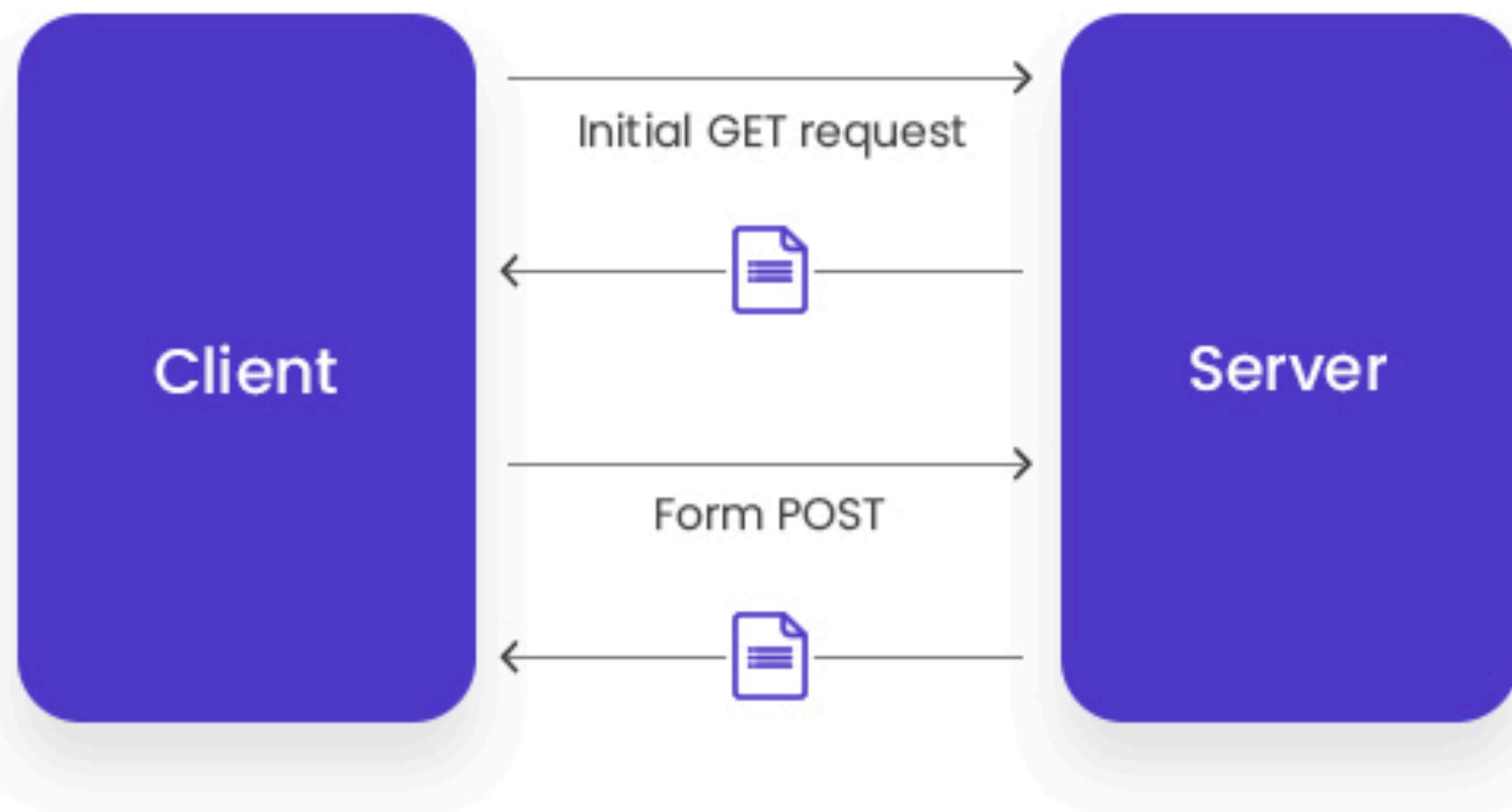
SPA



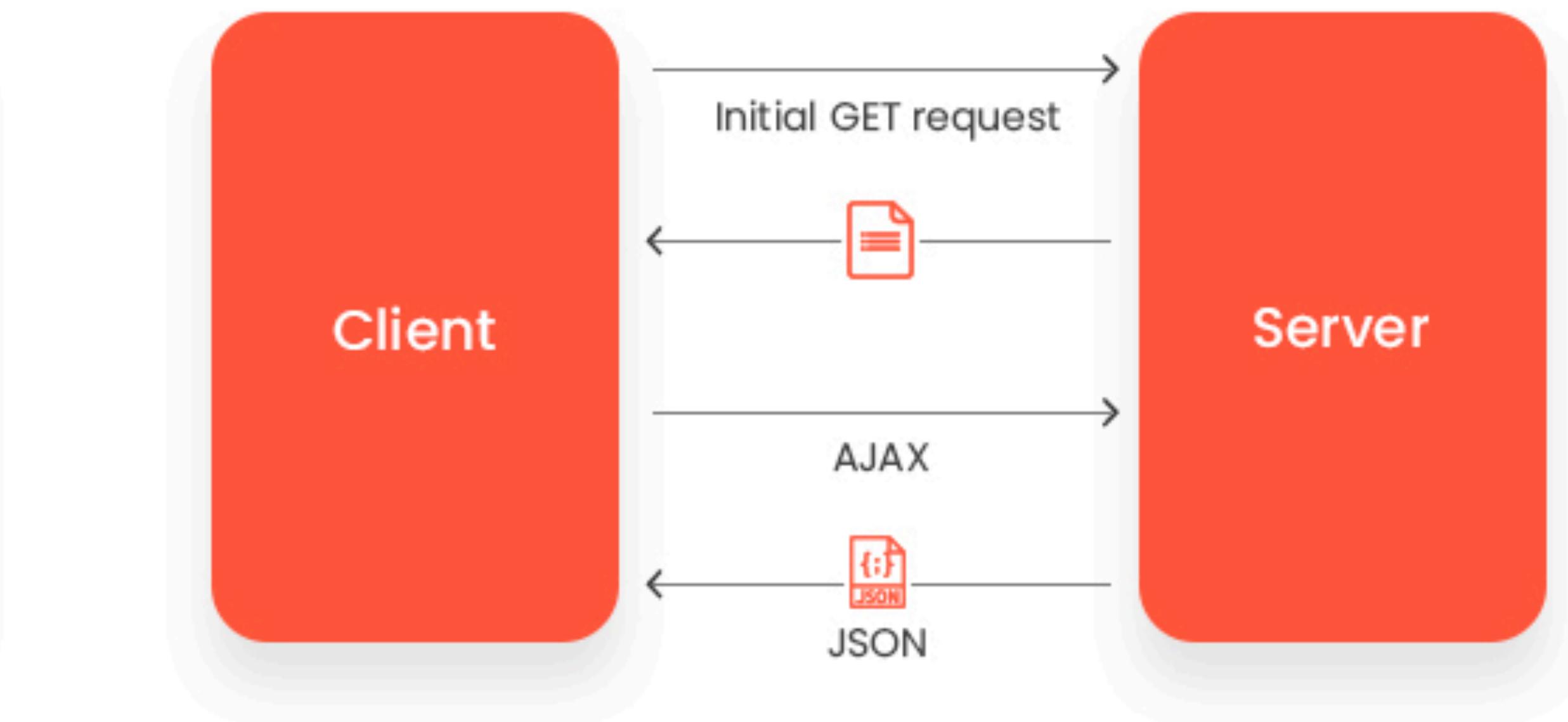
MPA



Traditional Page Lifecycle



SPA Lifecycle



React Router

... a client & server-side routing library for React

```
<Routes>
  <Route path="/" element={<HomePage />} />
  <Route path="about" element={<AboutPage />} />
  <Route path="*" element={<Navigate to="/" />} />
</Routes>
```

What's a Router?

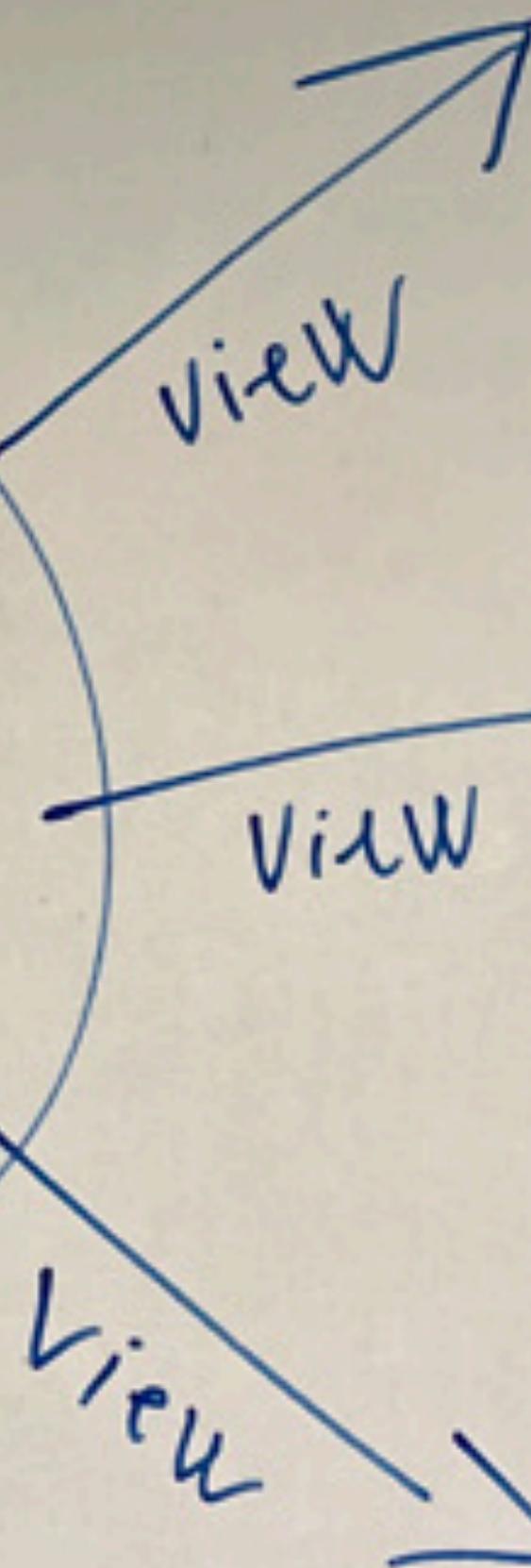
“It is the piece of software in charge to organize the states of the application, switching between different views.”

It's a key component in Single Page Applications.



/create
route

Router
(router.js)



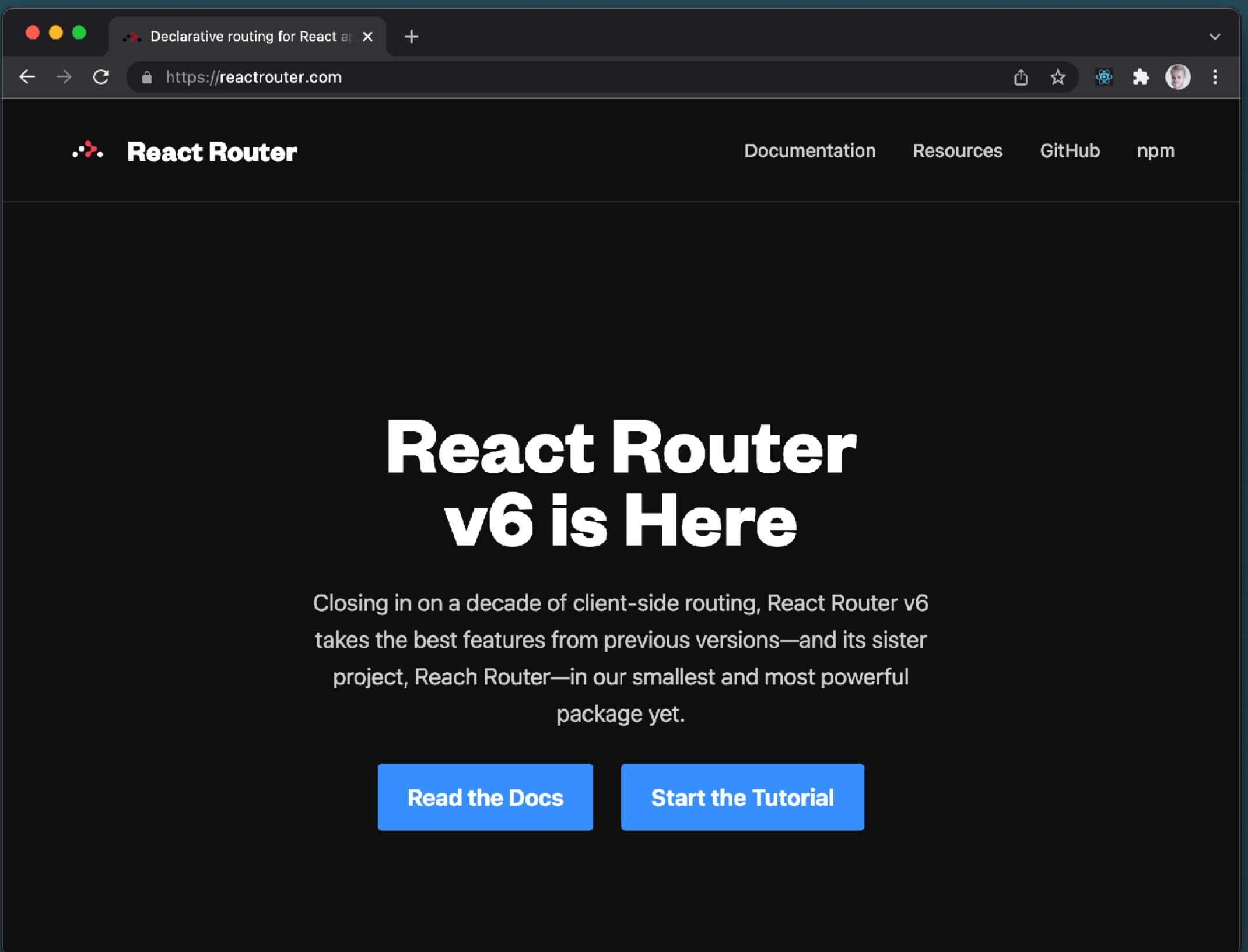
users
page

create
page

update
page

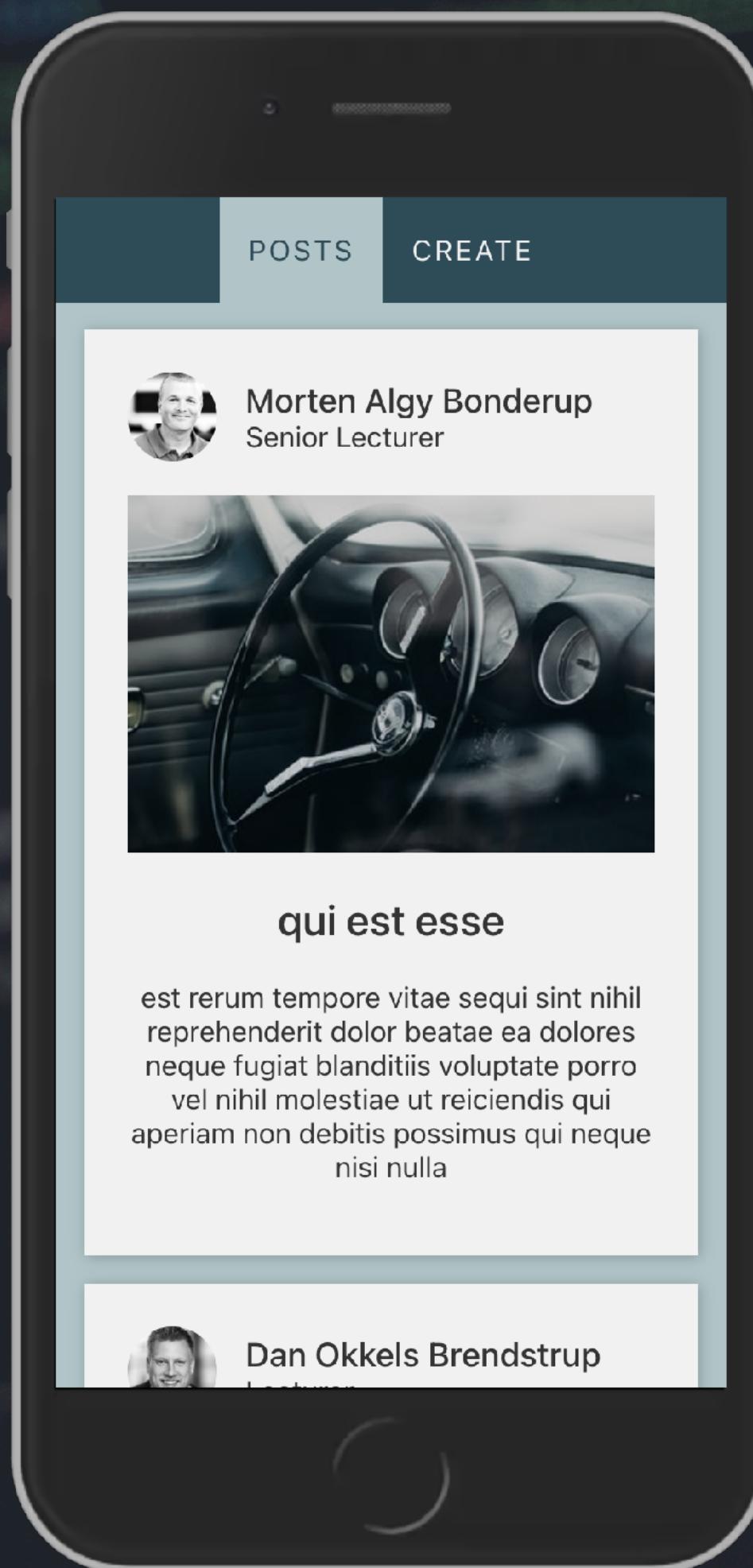
Create React SPA

With React Router



Create React SPA

React CRUD App



- GitHub Repo (code)
- Running solution
- Data Sources:
 - posts
 - users

Wrap Up



1. In Pairs: review your solutions step by step: CRUD
2. What are the differences in your implementations?
3. When and where do you make API calls? List them.
4. Improve and finalise the CRUD app.
5. Note comments and questions.

Shorter and simpler way to do if/else

The screenshot shows a web browser window with the title "Conditional branching: if, ?" and the URL <https://javascript.info/ifelse#conditional-operator>. The page content is about the conditional operator (ternary operator). It includes a sidebar with navigation links for "Chapter" (JavaScript Fundamentals), "Lesson navigation" (The "if" statement, Boolean conversion, The "else" clause, Several conditions: "else if", Conditional operator "?", Multiple "?", Non-traditional use of "?", Tasks (5), Comments, Share (Twitter, Facebook), and Edit on GitHub. There is also an advertisement for Kendo UI Kits for Figma.

Conditional operator ?'

Sometimes, we need to assign a variable depending on a condition.

For instance:

```
1 let accessAllowed;
2 let age = prompt('How old are you?', '');
3
4 if (age > 18) {
5   accessAllowed = true;
6 } else {
7   accessAllowed = false;
8 }
9
10 alert(accessAllowed);
```

The so-called "conditional" or "question mark" operator lets us do that in a shorter and simpler way.

The operator is represented by a question mark `?`. Sometimes it's called "ternary", because the operator has three operands. It is actually the one and only operator in JavaScript which has that many.

The syntax is:

```
1 let result = condition ? value1 : value2;
```

The `condition` is evaluated: if it's truthy then `value1` is returned, otherwise – `value2`.

For example:

```
1 let accessAllowed = (age > 18) ? true : false;
```

<https://javascript.info/ifelse#conditional-operator>
Ternary Operator

? . is a safe way to access nested object properties

The screenshot shows a dark-themed web browser window displaying the [JavaScript.info](https://javascript.info/optional-chaining) website. The page title is "Optional chaining '?.'". The URL in the address bar is <https://javascript.info/optional-chaining>. The page content is about optional chaining, a recent addition to the JavaScript language. It discusses the "non-existing property" problem and provides code examples. A sidebar on the left contains navigation links for chapters like "Objects: the basics" and "Lesson navigation". A sidebar at the bottom is sponsored by LogicMonitor.

Chapter
Objects: the basics

Lesson navigation

The "non-existing property" problem

Optional chaining

Short-circuiting

Other variants: `?()`, `?[]`

Summary

Comments

Share

[Edit on GitHub](#)

LogicMonitor

Intelligent insights you can actually use. Smarter troubleshooting, more possibilities for your applications.

Optional chaining '?.' is a safe way to access nested object properties, even if an intermediate property doesn't exist.

Optional chaining '?.'

A recent addition

This is a recent addition to the language. Old browsers may need polyfills.

The optional chaining `?.` is a safe way to access nested object properties, even if an intermediate property doesn't exist.

The "non-existing property" problem

If you've just started to read the tutorial and learn JavaScript, maybe the problem hasn't touched you yet, but it's quite common.

As an example, let's say we have `user` objects that hold the information about our users.

Most of our users have addresses in `user.address` property, with the street `user.address.street`, but some did not provide them.

In such case, when we attempt to get `user.address.street`, and the user happens to be without an address, we get an error:

```
1 let user = {}; // a user without "address" property
2
3 alert(user.address.street); // Error!
```

<https://javascript.info/optional-chaining>

?? returns the first argument if it's not null/undefined

The screenshot shows a dark-themed web browser window displaying the [Nullish coalescing operator ??](https://javascript.info/nullish-coalescing-operator) article from the [JavaScript.info](https://javascript.info) website.

The page title is "Nullish coalescing operator '??'" and the subtitle indicates it is a "Recent addition". A note states: "This is a recent addition to the language. Old browsers may need polyfills."

The main content explains that the nullish coalescing operator is written as two question marks `??`. It treats `null` and `undefined` similarly. The result of `a ?? b` is:

- if `a` is defined, then `a`,
- if `a` isn't defined, then `b`.

In other words, `??` returns the first argument if it's not `null/undefined`. Otherwise, the second one.

The nullish coalescing operator isn't anything completely new. It's just a nice syntax to get the first "defined" value of the two.

We can rewrite `result = a ?? b` using the operators that we already know, like this:

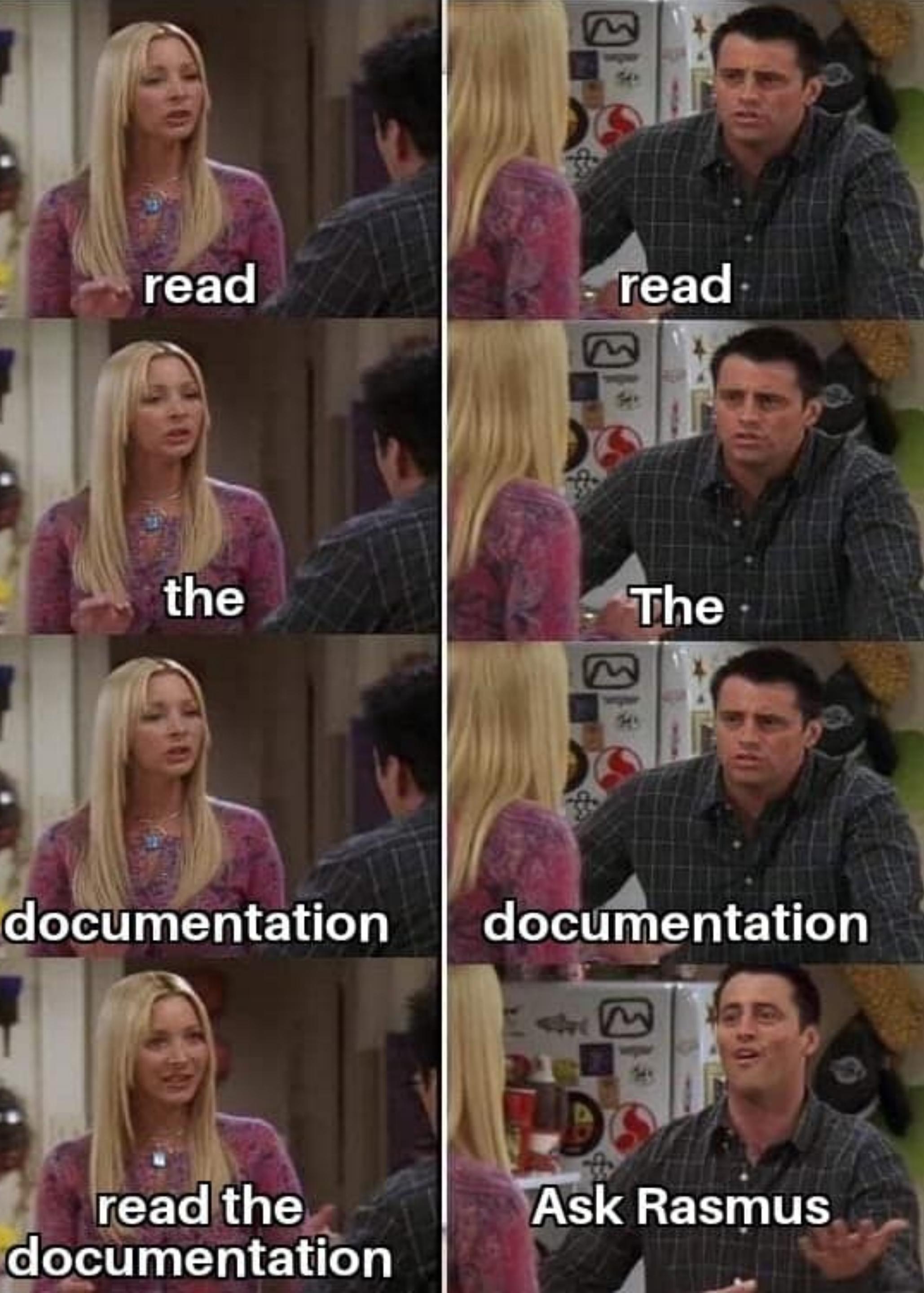
```
1 result = (a !== null && a !== undefined) ? a : b;
```

The left sidebar contains a navigation menu with links to "Chapter", "JavaScript Fundamentals", "Lesson navigation", "Comparison with ||", "Precedence", "Summary", "Comments", "Share" (with Twitter and Facebook icons), and "Edit on GitHub". There is also an advertisement for "Ad by Carbon" featuring "PagerDuty" and an "eBook" titled "Best practices for full-service ownership".

<https://javascript.info/nullish-coalescing-operator>

But RACE, how do you know all that stuff?

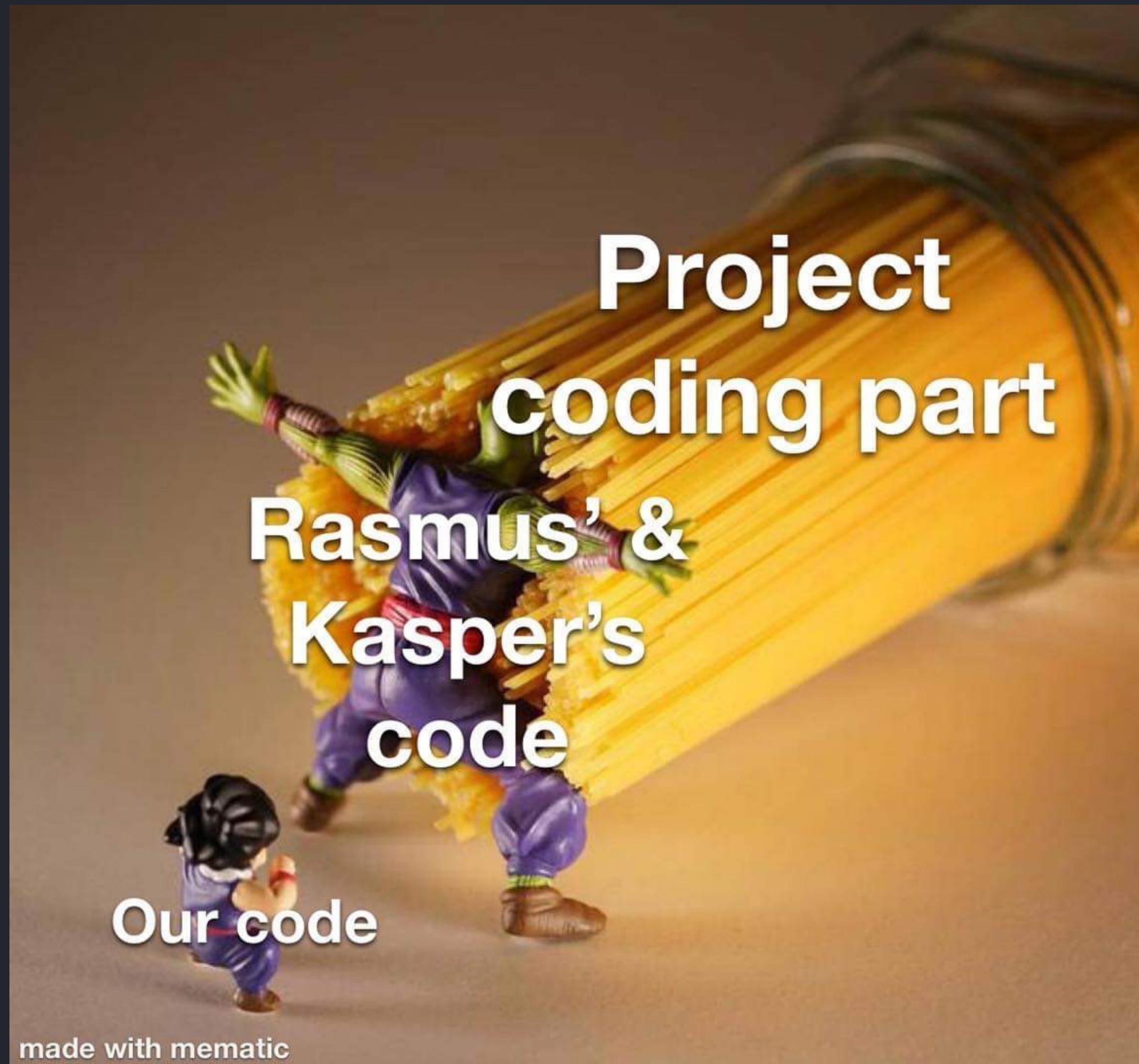
- [React Resources](#)
- [React Self-study](#)
- [Guides & Tutorials](#)
- [From Vanilla JavaScript to React Developer](#)





Full-stack Demo

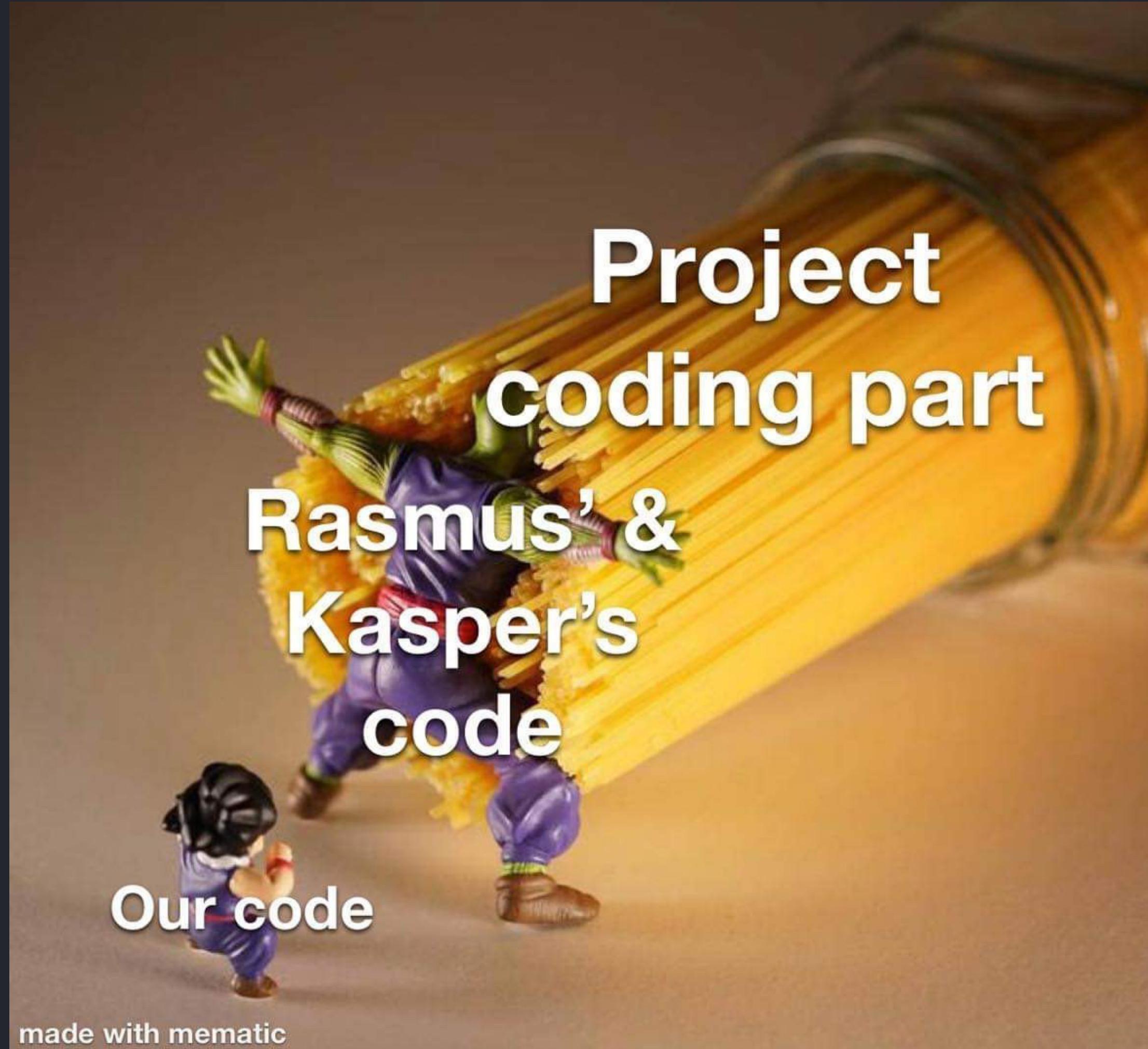
- React CRUD App (Frontend)
- PHP Services / JSON API (Backend)
- MySQL DB (Database)



made with mematic

Full-stack Demo

- Frontend
- Backend + SQL import



Set up

1. Import and set up local database
2. Run and test PHP Service (JSON API)
3. Customise React CRUD App



Interdisciplinary Project

WU-F22V – 1st semester

Kick Off

Code Every Day

