Forms, REST og CRUD

REST - Update og delete

Datamatiker

kea
KØBENHAVNS ERHVERVSAKADEMI

Dagens Formål

- Viden om hvordan vi opdaterer og sletter eksisterende data (objekter) i databasen via REST API'et.

- Derudover hvordan vi skaber overblik og "ro" i de seneste dages "kaos" af begreber, koncepter og kodeblokke.

Agenda

1. **Demo af Post App med GET, POST, PUT og DELETE**
   - Demo af Create, Read, Update og Delete
   - Networktab og requests
   - Gennemgå struktur og kode med slides
   - Importer ny datastruktur til Firebase: 15. Import a New Data Structure
2. **REST, CRUD og HTTP-metoder: PUT og DELETE**
   - Hvad er særligt for PUT og DELETE?
   - REST og URL-struktur
3. **Update og delete med Firebase Database REST API**
   - Anvendelse af Fetch, HTTP-metoden og request body med JSON

1. SEMESTERS STRUKTUR

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

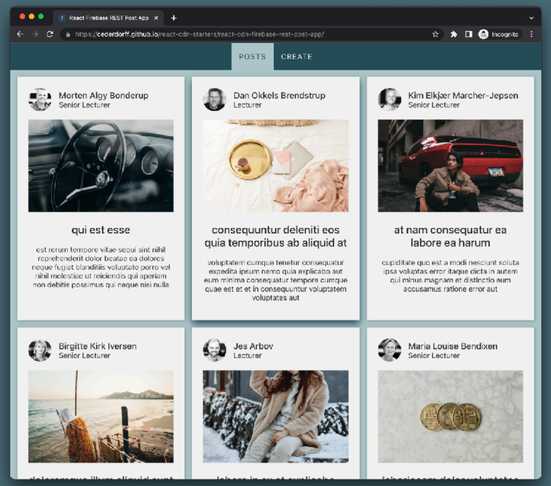
Eksamen ligger 12.-14. juni

Fetch, HTTP Request & Response

WEB APP

FIREBASE REST API

DATA SOURCE



FETCH(GET, POST, PUT, DELETE)

REQUEST

JSON DATA

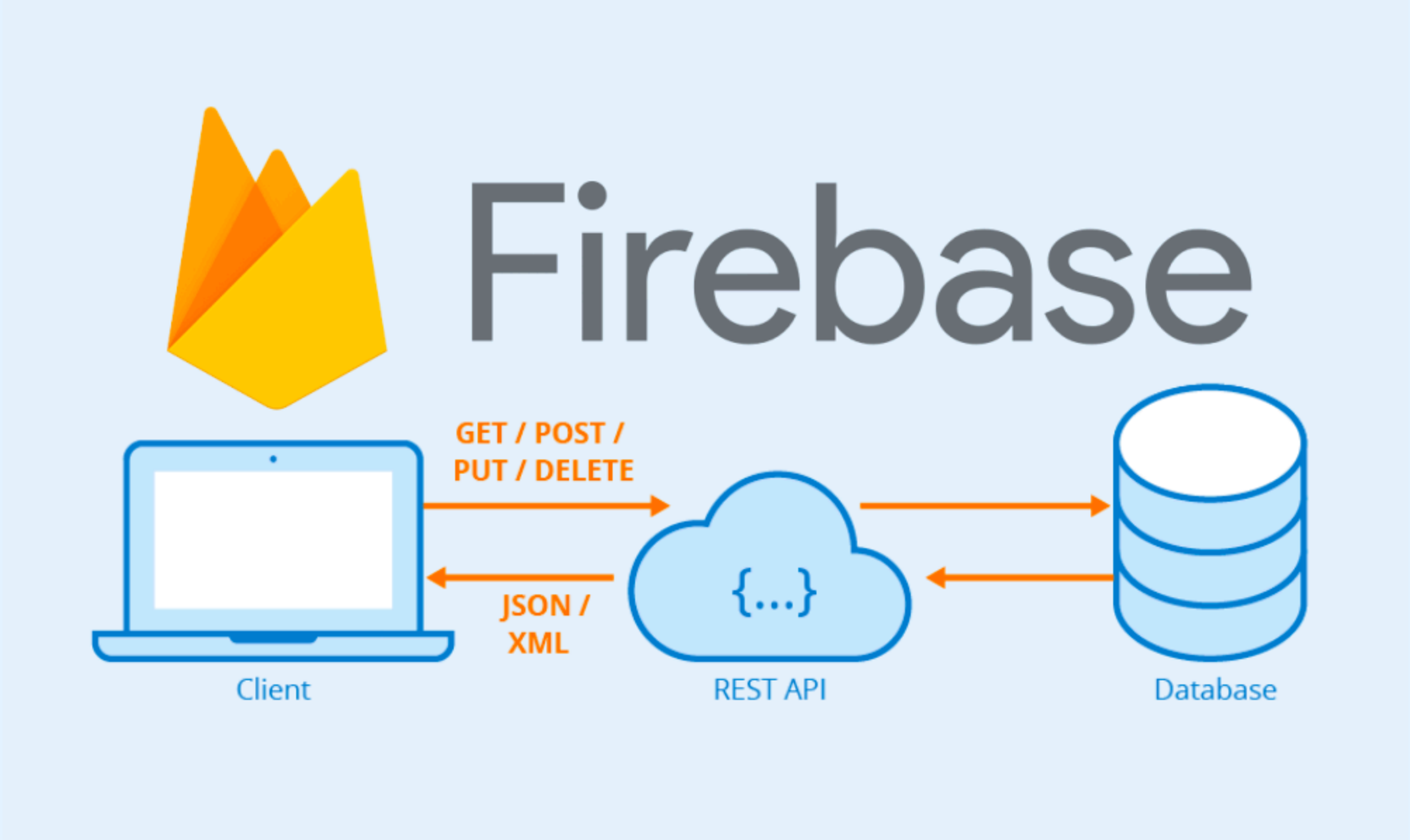
RESPONSE

JSON

Firebase
Realtime Database

posts

users



javascript.plainenglish.io/firebase-cloud-functions-tutorial-creating-a-rest-api-8cbc51479f80

Fronter | Post App | 127.0.0.1:5500/index.html

Post App with Firebase REST API

CREATE NEW POST

Posts

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 debitis possimus qui neque nisi nulla

DELETE UPDATE

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

DELETE UPDATE

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

DELETE UPDATE

Doloremque illum aliquid sunt Updated 🔥

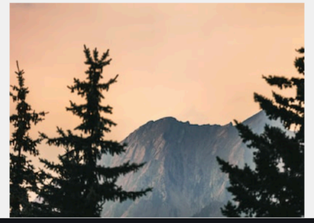
Doloremque ex facilis sit sint culpa soluta assumenda eligendi non ut eius sequi ducimus vel quasi veritatis est dolores

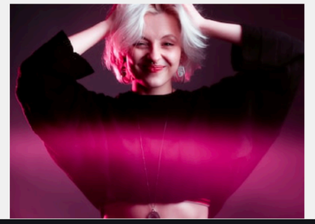
DELETE UPDATE

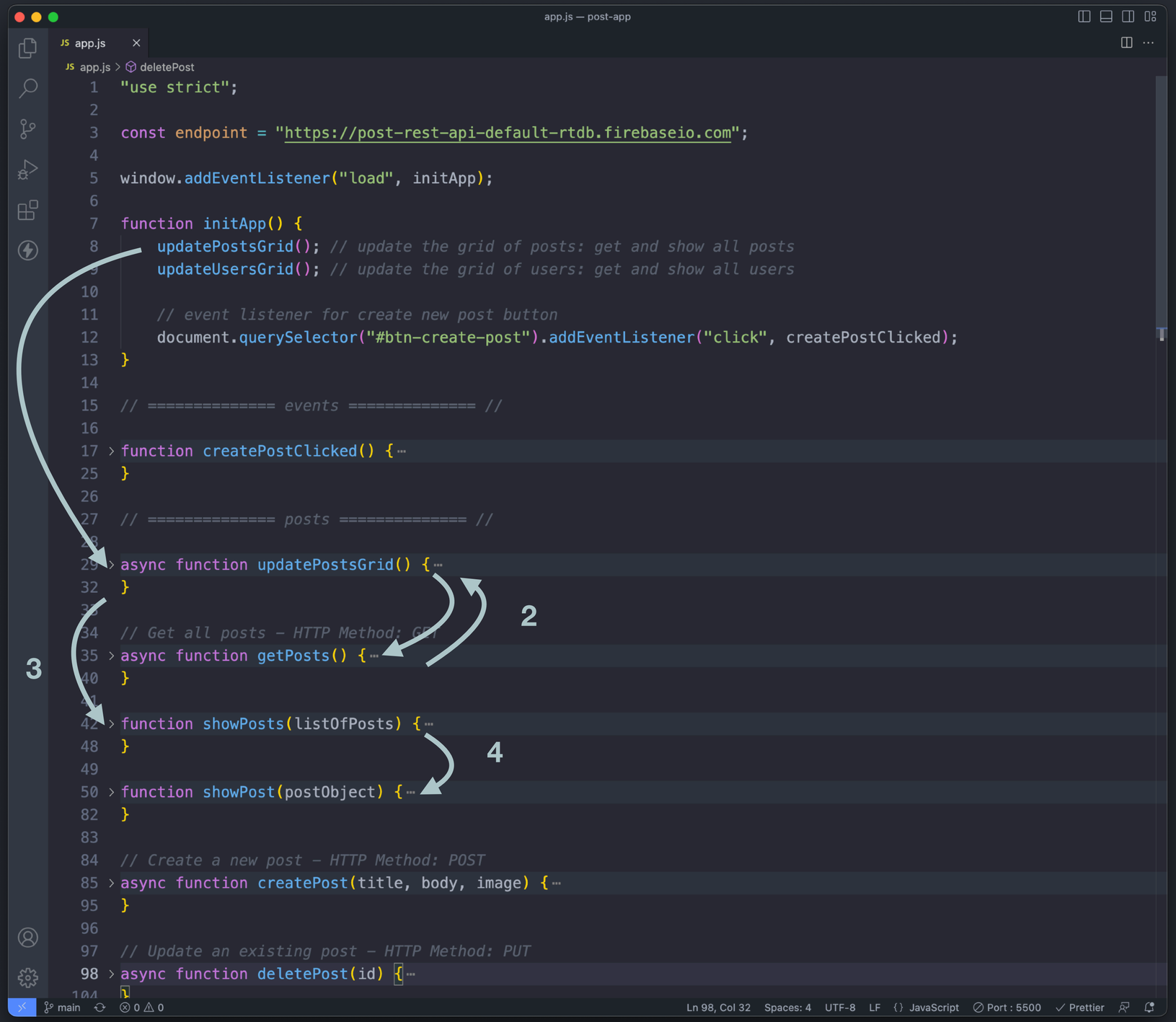
Labore in ex et explicabo corporis aut quas

Ex quod dolorem ea eum iure qui provident amet quia qui facere excepturi et repudiandae asperiores molestias provident minus incidunt vero fugit rerum sint sunt excepturi provident

DELETE UPDATE





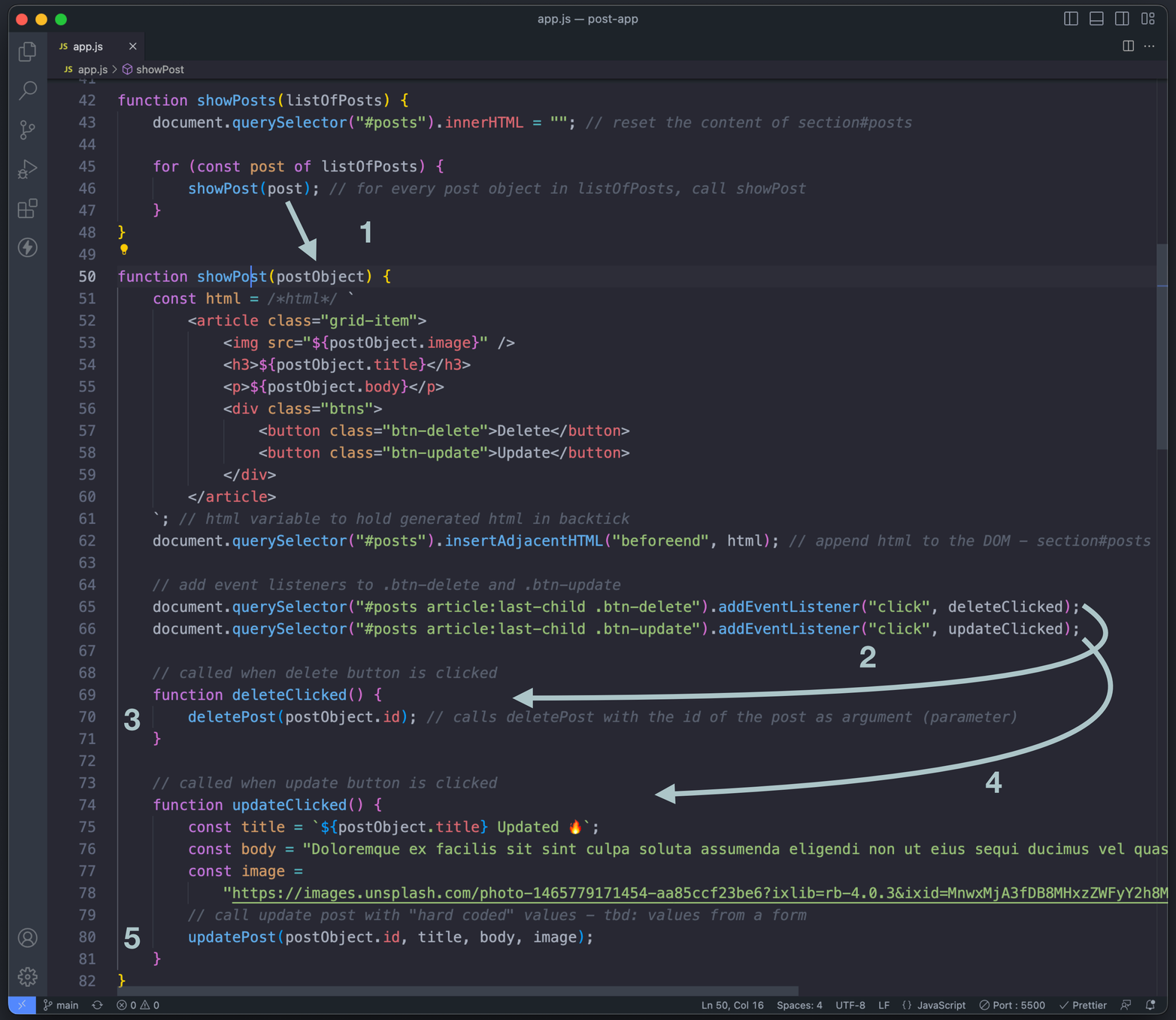


1. initApp kalder updatePostsGrid

2. updatePostsGrid kalder showPosts. showPosts henter alle posts via et GET Request. showPosts laver objekt med objekter om til array med objekter og returnerer array tilbage til updatePostsGrid.

3. updatePostsGrid kalder showPosts. showPosts looper igennem alle posts og for hver post kaldes showPost.

4. showPost foretager DOM-Manipulation på baggrund af ét postobjekt, der gives som argument (fra loop i showPosts).



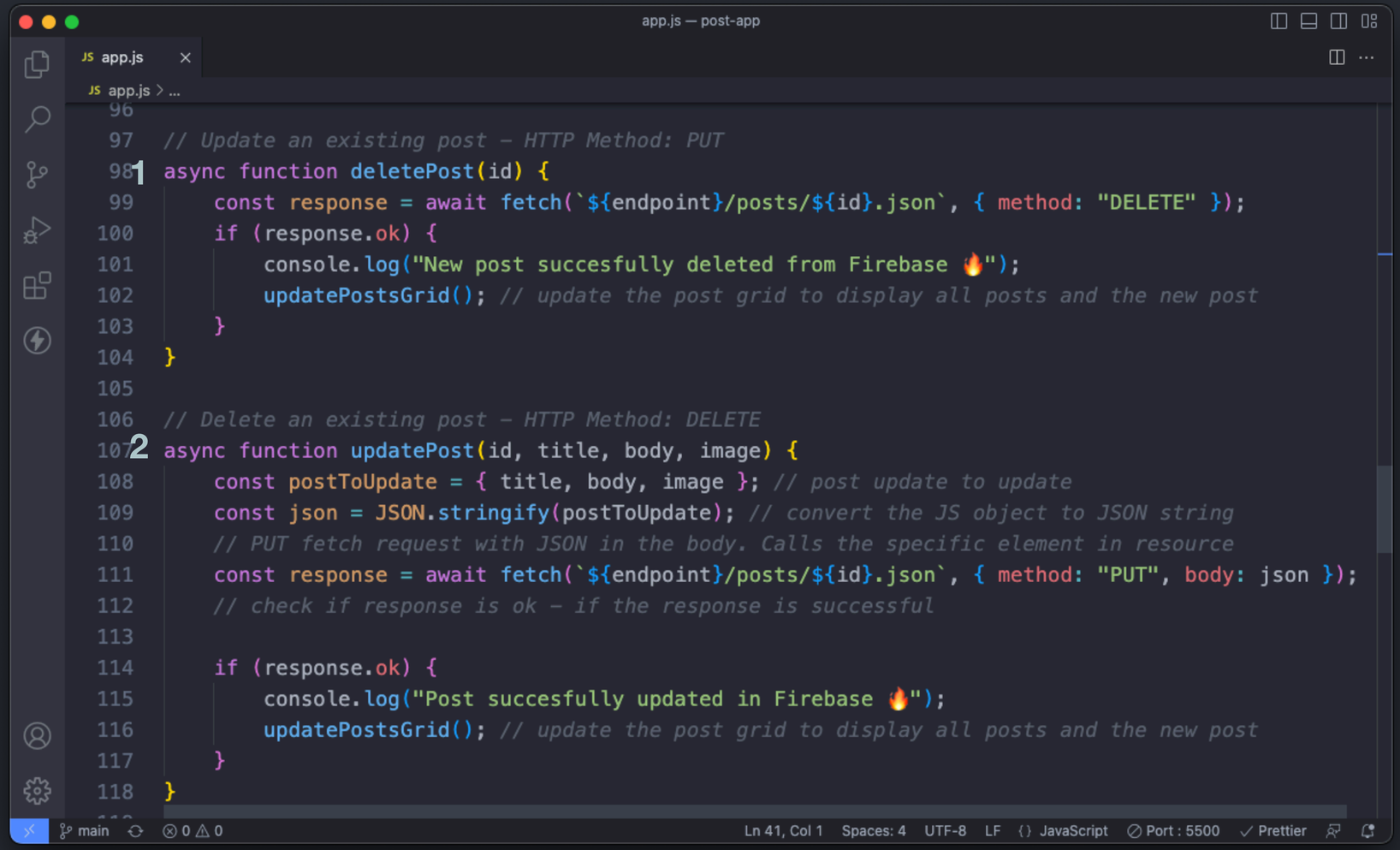
1. showPosts looper igennem alle posts og for hver post kaldes showPost med et post som argument (parameter).

2. deleteClicked event tilføjes Delete-knappen.

3. deletePost kaldes med postobjektets id.

4. updateClicked event tilføjes Update-knappen.

5. updatePost kaldes med de informationer der skal til for at opdatere et post's properties: id, title, body og image.



1. deletePost kaldes af deleteClicked.

   deletePost foretager et HTTP request med metode DELETE. Requestet foretages til det specifikke element, som skal slettes. Derfor id i url'en.

   Hvis DELETE requestet er succesfuldt, kaldes updatePostsGrid, der sørger for at alle posts hentes på ny og vises i DOM'en.

2. updatePost kaldes af updateClicked.

   updatePost laver et objekt med de værdier (props), der skal opdateres. Objektet konverteres til JSON og sendes med i body'en på et PUT Request.

   Requestet foretages til det specifikke element, som skal opdateres. Derfor id i url'en.

   Hvis PUT requestet er succesfuldt, kaldes updatePostsGrid, der sørger for at alle posts hentes på ny og vises i DOM'en.

1. createPostClicked event tilføjes Create-knappen, der ses i header.

   createPostClicked genererer værdier, som burde komme fra en formular. Men for at vi kan træne, “hard-coder” og genererer vi nogle værdier.

2. createPost kaldes med de genererede værdier for title, body og image.

   createPost laver et nyt postobjektet med title, body og image. Det konverteres til JSON. Dernæst foretages et HTTP POST request med JSON i requestets body.

   Hvis POST requestet er succesfuldt, kaldes updatePostsGrid, der sørger for at alle posts hentes på ny og vises i DOM’en.

```
function initApp() {
    updatePostsGrid(); // update the grid of posts: get and show all posts
    updateUsersGrid(); // update the grid of users: get and show all users

    // event listener for create new post button
    document.querySelector("#btn-create-post").addEventListener("click", createPostClicked);
}

// ============== events ============== //

function createPostClicked() {
    const randomNumber = Math.floor(Math.random() * 100 + 1);
    const title = `My Post Title Number ${randomNumber}`;
    const body = "Quo deleniti praesentium dicta non quod aut est molestias molestias et offici
    const image =
        "https://plus.unsplash.com/premium_photo-1675330628475-b4e0e2a3c4a7?ixlib=rb-4.0.3&ixid
    // call createPost with "hard-coded" values - tbd: values from a form
    createPost(title, body, image);
}


// Create a new post - HTTP Method: POST
async function createPost(title, body, image) {
    const newPost = { title, body, image }; // create new post object
    const json = JSON.stringify(newPost); // convert the JS object to JSON string
    // POST fetch request with JSON in the body
    const response = await fetch(`${endpoint}/posts.json`, { method: "POST", body: json });
    // check if response is ok - if the response is successful
    if (response.ok) {
        console.log("New post succesfully added to Firebase 🔥");
        updatePostsGrid(); // update the post grid to display all posts and the new post
    }
}
```

1

2

Fetch, JSON, CRUD & REST

Slides:
https://cederdorff.github.io/dat-js/slides/Fetch-JSON-CRUD-og-REST.pdf

Web Development

KØBENHAVNS ERHVERVSAKADEMI