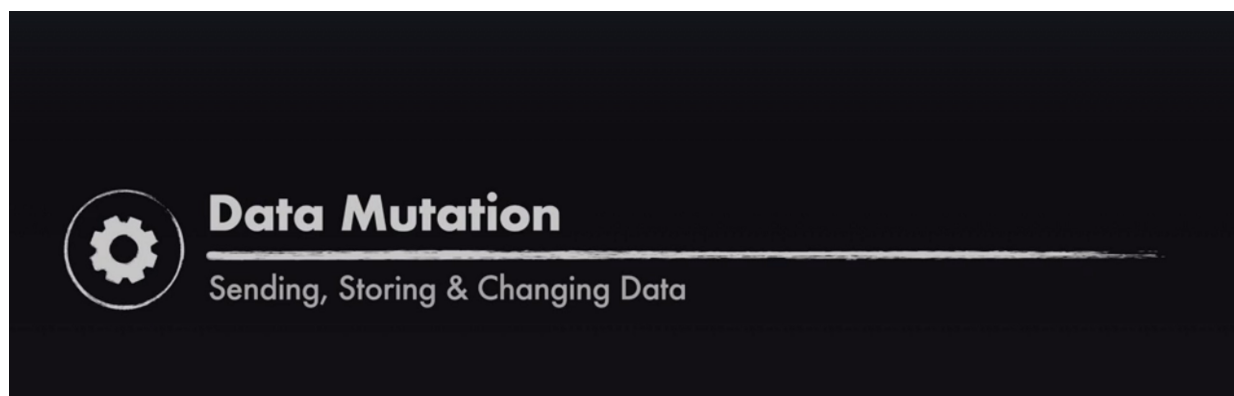
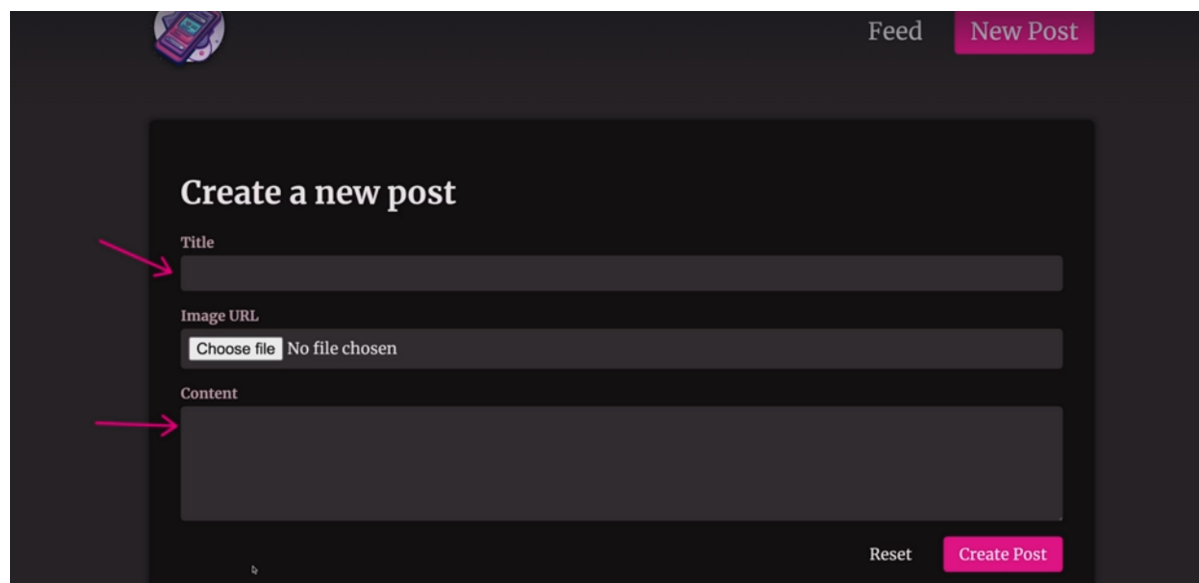


## Section 6: Mutating Data - Deep Dive

14 September 2024 11:28



We want to store this form data to the database.

A screenshot of a web application's "Create a new post" form. The form is set against a dark background. At the top left of the form area is a small profile picture icon. The title "Create a new post" is centered at the top of the form. Below the title are three input fields: a "Title" text input, an "Image URL" input with a "Choose file" button and "No file chosen" text, and a "Content" text area. Two red arrows point to the "Title" and "Content" fields. At the bottom right of the form are "Reset" and "Create Post" buttons. The "Create Post" button is highlighted in a bright pink color. Above the form, in the top right corner of the page, are "Feed" and "New Post" navigation links, with "New Post" also highlighted in pink.

```
> .next
> app
> assets
> components
✓ lib
JS format.js
JS posts.js
> node_modules
> public
{} jsconfig.json
JS next.config.mjs
{} package-lock.json
{} package.json
≡ posts.db

50 export async function getPosts(maxNumber) {
57   const stmt = db.prepare(`
62     GROUP BY posts.id
63     ORDER BY createdAt DESC
64     ${limitClause}`);
65
66   await new Promise((resolve) => setTimeout(resolve, 1000));
67   return maxNumber ? stmt.all(maxNumber) : stmt.all();
68 }
69
70 export async function storePost(post) {
71   const stmt = db.prepare(`
72     INSERT INTO posts (image_url, title, content, user_id)
73     VALUES (?, ?, ?, ?)`);
74   await new Promise((resolve) => setTimeout(resolve, 1000));
75   return stmt.run(post.imageUrl, post.title, post.content, post
76
77   which does indeed include code
```

Options

## There Are Multiple Data Mutation Strategies

### Option 1

#### Standalone Backend

A separate backend receives requests from the NextJS app and stores / mutates all data

#### Separate Server

POST, PATCH  
etc. requests

#### NextJS App

**Problem:** Unnecessary complexity,  
unnecessary extra server

### Option 2

#### Integrated NextJS App

The NextJS app itself contains the data mutation code and reaches out directly to the data stores

#### NextJS App

Renders pages & handles form  
submissions / data mutation

**Advantage:** Reduced complexity, single  
codebase & server

react@18.2.0

Overview

Hooks

Components

APIs

Directives

'use client'

'use server'

react-dom@18.2.0

Hooks

Components

APIs

Client APIs

Server APIs

## Server Actions in forms

The most common use case of Server Actions will be calling server functions that mutate data. On the browser, the **HTML form element** is the traditional approach for a user to submit a mutation. With React Server Components, React introduces first-class support for Server Actions in **forms**.

Here is a form that allows a user to request a username.

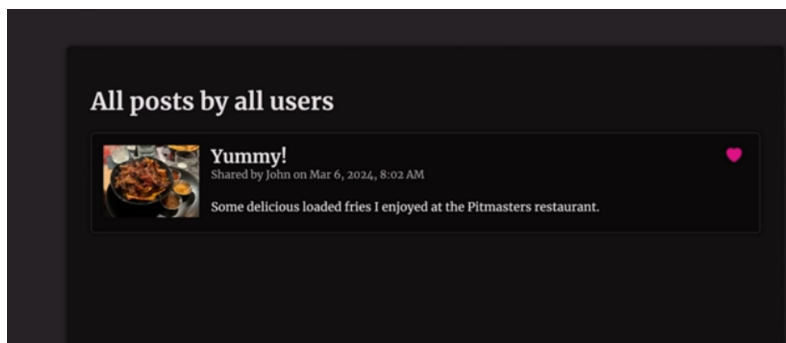
```
// App.js

async function requestUsername(formData) {
  'use server';
  const username = formData.get('username');
  // ...
}

export default function App() {
  return (
    <form action={requestUsername}>
      <input type="text" name="username" />
      <button type="submit">Request</button>
    </form>
  );
}
```

which is not the case in your standard React project

## Performing Optimistic Updates With NextJS



Here in this application,  
we would provide a better user experience  
if we would immediately update this like status  
even before it has been processed in the database.  
And that's a concept or a pattern  
called optimistic updating,  
which in the end simply means that we're optimistic  
that our update will work, and therefore we wanna show  
the new state immediately to the user,  
and perform the update behind the scenes.  
And then only if that update would fail  
or something like this, we wanna roll back the update.

From <<https://www.udemy.com/course/nextjs-react-the-complete-guide/learn/lecture/43340832#announcements>>