Jake and Nates:

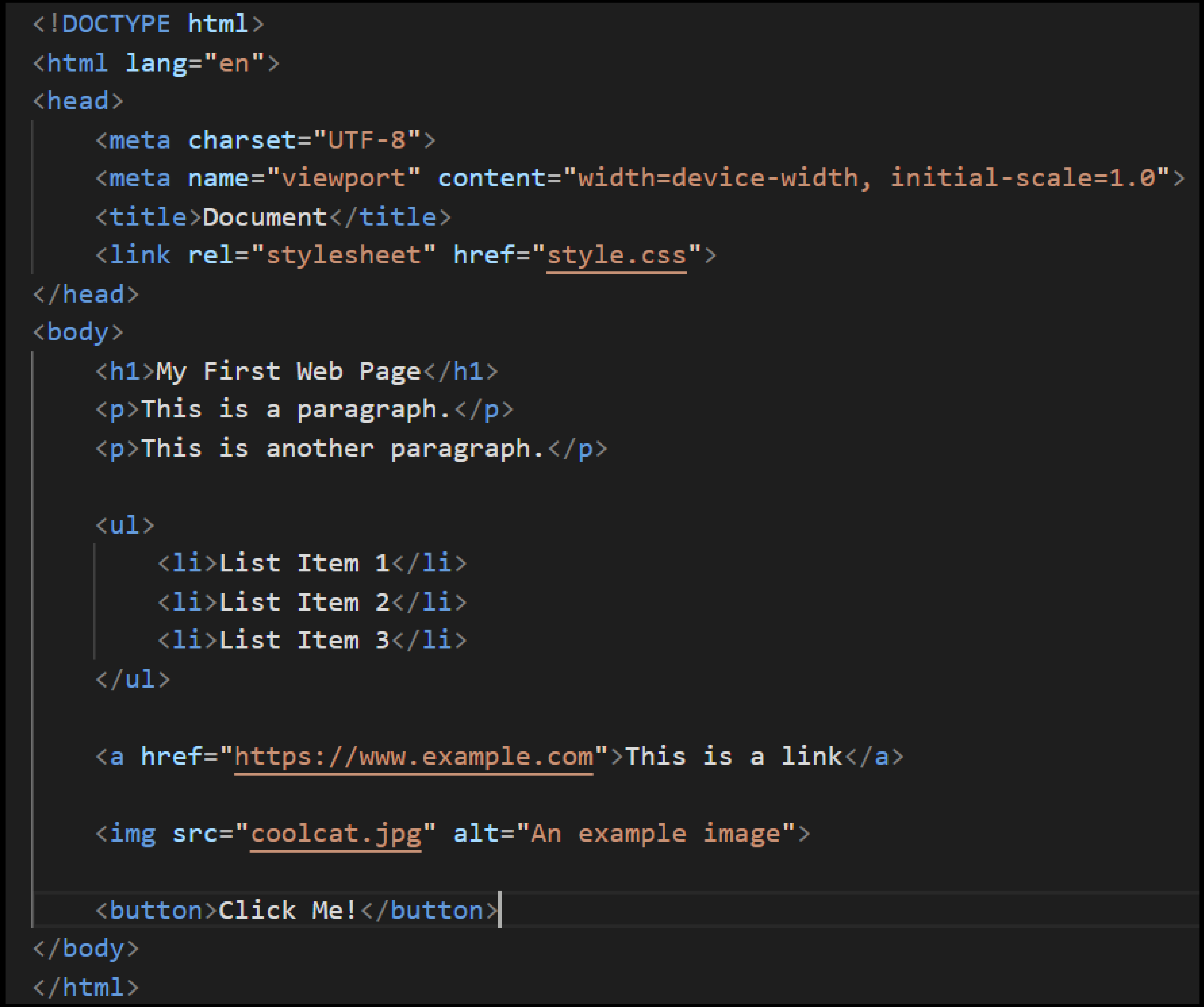
Web Dev Workshop

What is Web Dev?

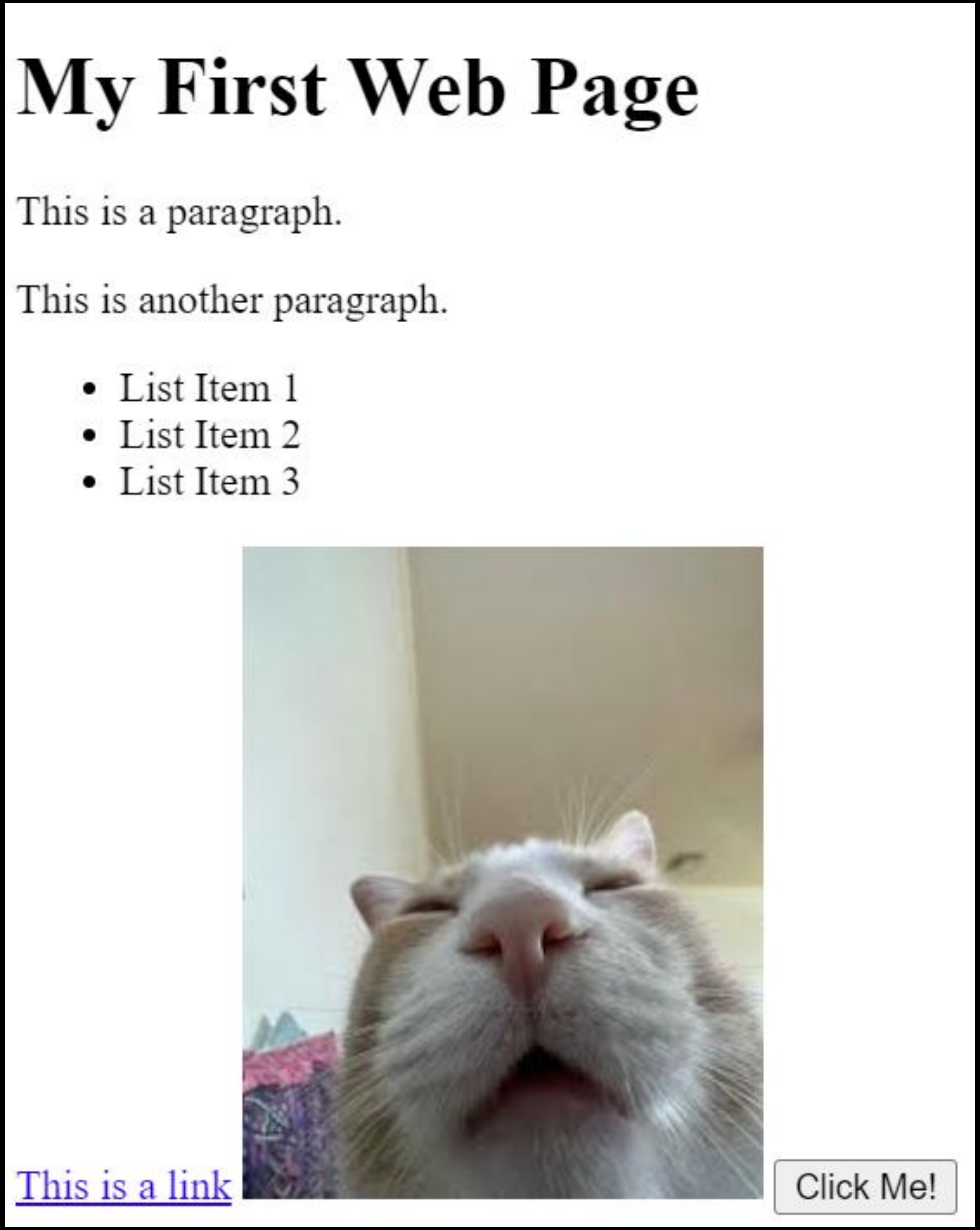
- Frontend Development:
  - Visuals and interactive elements the user can see
  - HTML, CSS, Javascript
- Backend Development:
  - Serverside logic and application functionality
  - PHP, Python, Go, SQL
  - Many open source/hosted databases exist: Supabase, Pocketbase, Firebase

Frontend Basics

- HTML (Hyper Text Markup Language)



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <h1>My First Web Page</h1>
    <p>This is a paragraph.</p>
    <p>This is another paragraph.</p>

    <ul>
        <li>List Item 1</li>
        <li>List Item 2</li>
        <li>List Item 3</li>
    </ul>

    <a href="https://www.example.com">This is a link</a>

    <img src="coolcat.jpg" alt="An example image">

    <button>Click Me!</button>
</body>
</html>
```



My First Web Page

This is a paragraph.

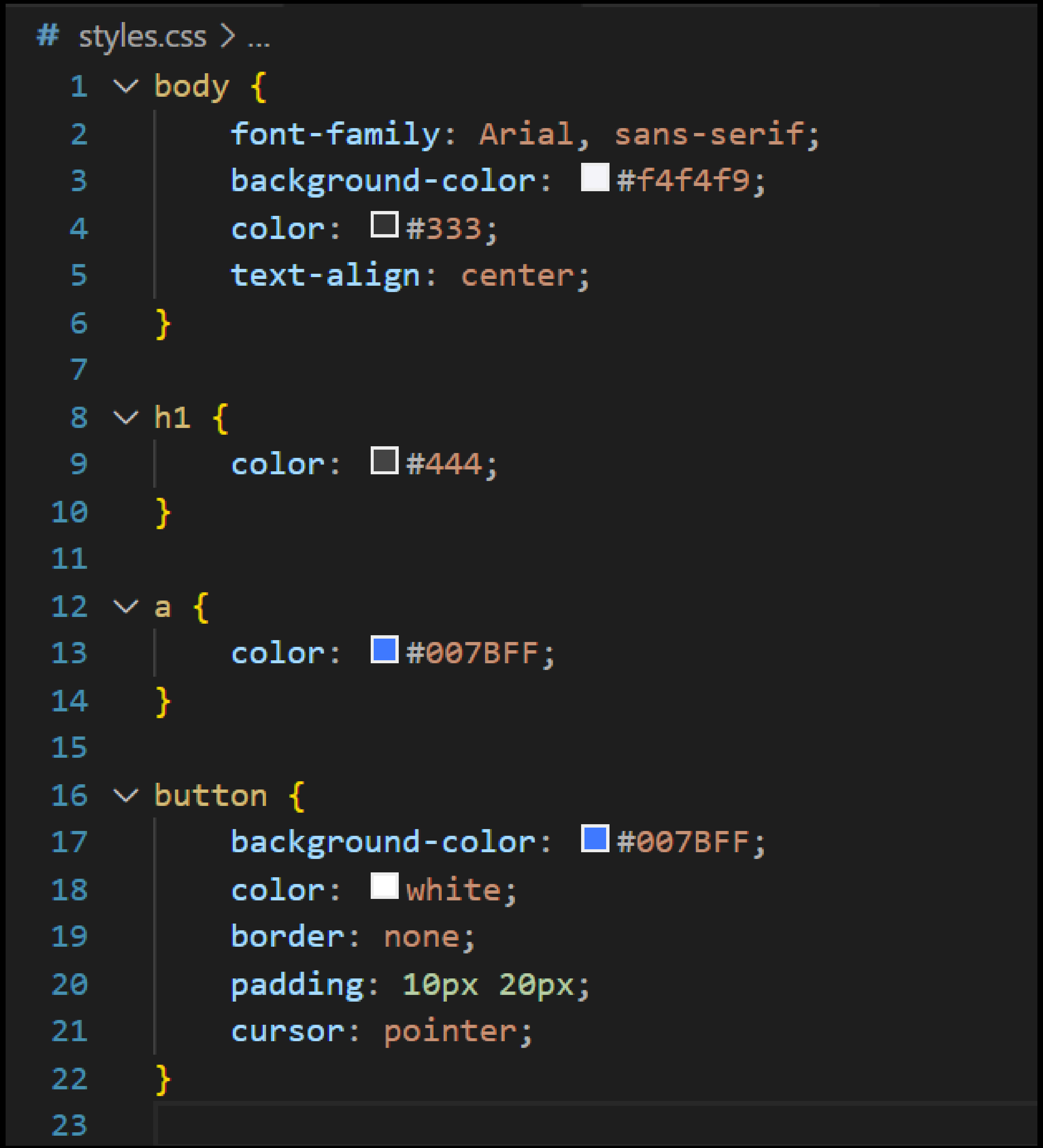
This is another paragraph.

- List Item 1
- List Item 2
- List Item 3

This is a link

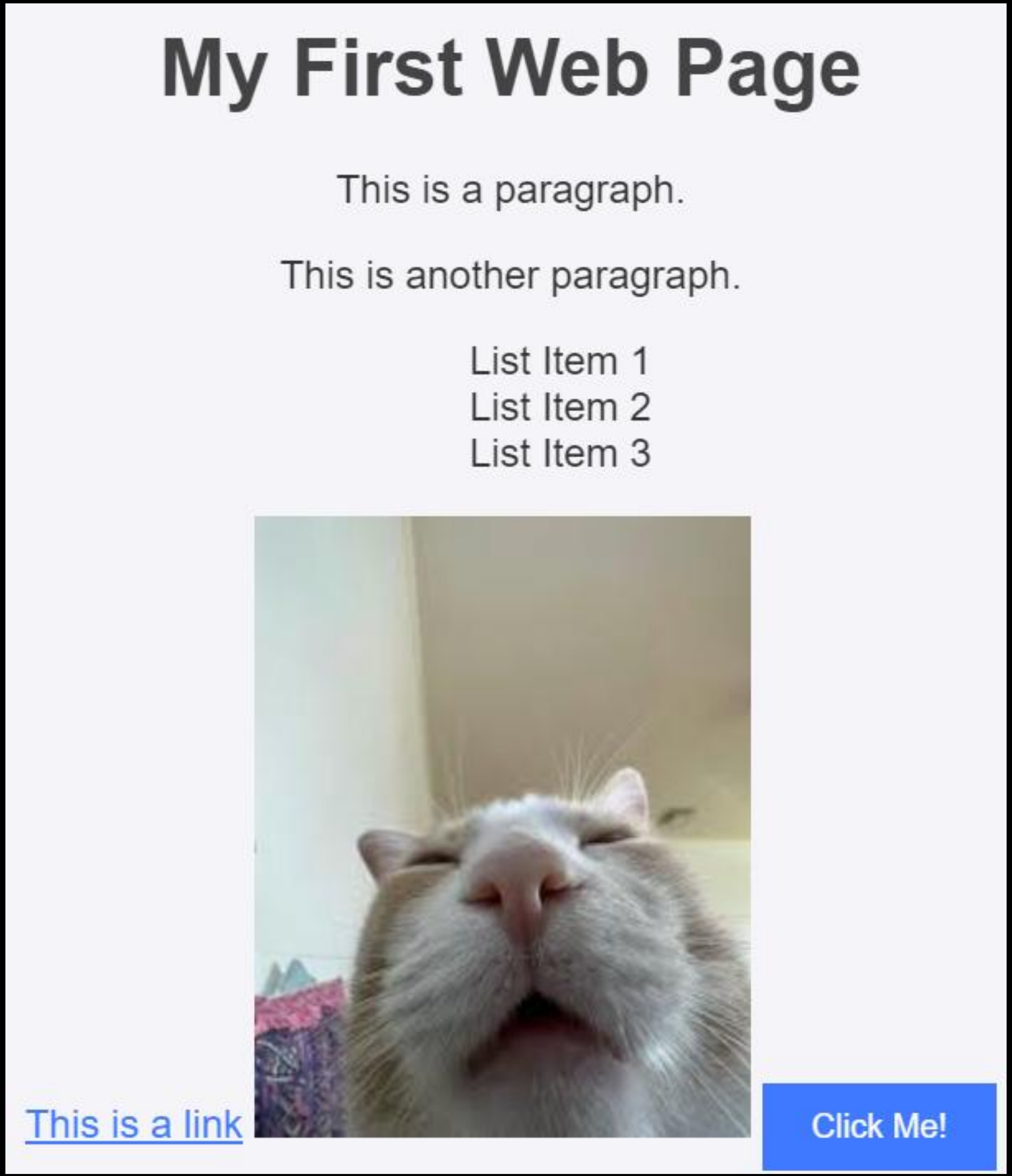
Frontend Basics

- CSS (Cascading Style Sheets)



# styles.css > ...

```css
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f9;
    color: #333;
    text-align: center;
}

h1 {
    color: #444;
}

a {
    color: #007BFF;
}

button {
    background-color: #007BFF;
    color: white;
    border: none;
    padding: 10px 20px;
    cursor: pointer;
}

```



My First Web Page

This is a paragraph.

This is another paragraph.

List Item 1
List Item 2
List Item 3

This is a link

Frontend Frameworks

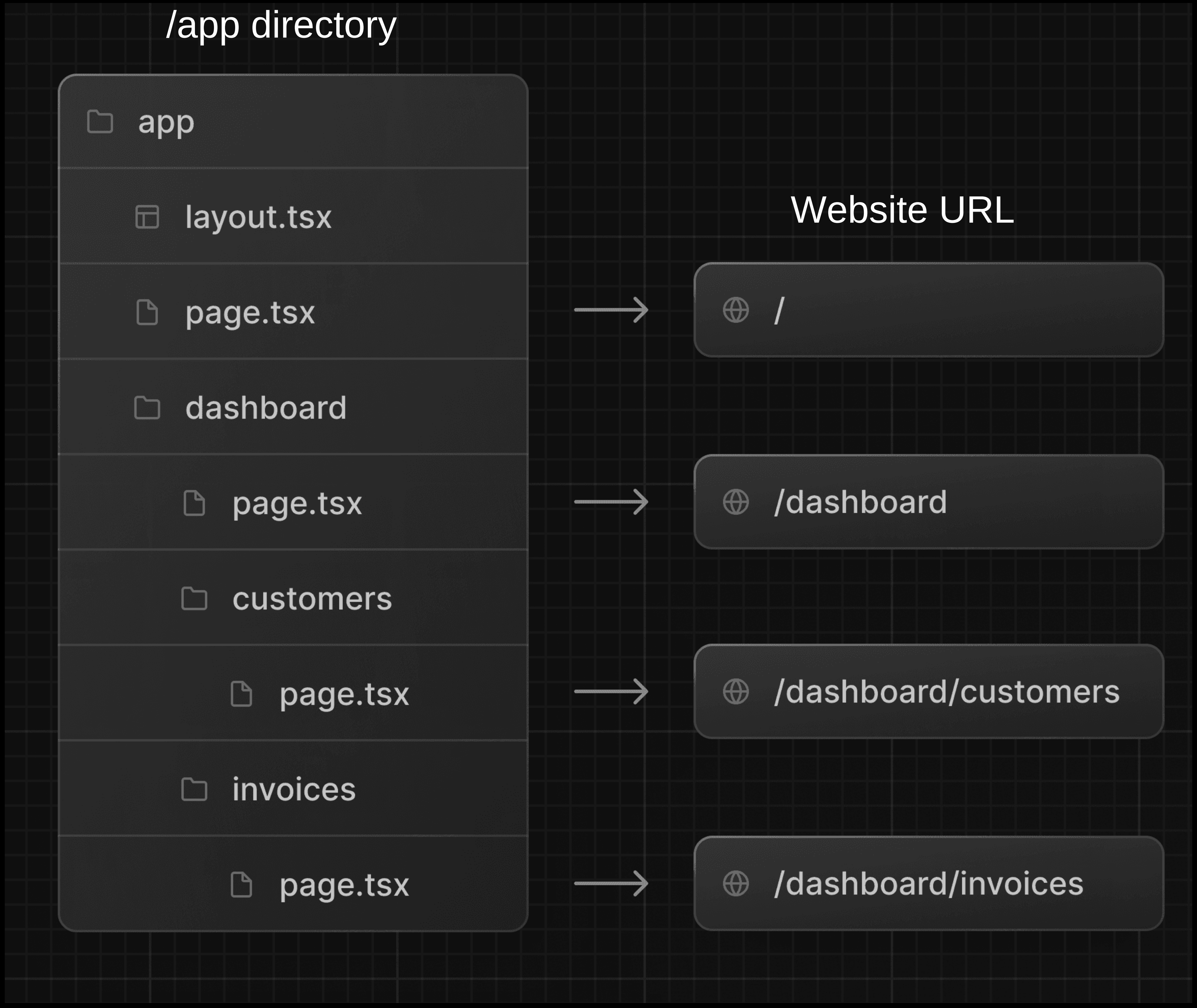
- Provide pre-built tools and structures to development
- There are many common frontend frameworks to choose from: React, Angular, Svelte, etc.
- Chosen based on project needs, performance, and developer experience
- Today we will use Next.js during the demo

What is Next.js?

Next.js is a powerful React framework that makes it easy to build fast, modern web applications. It offers features like server-side rendering, static site generation, and built-in routing. With TypeScript and Tailwind support, it’s perfect for building clean, customizable websites like personal portfolios, and it deploys seamlessly with platforms like Vercel.

Next App Router

- Each folder in app/ becomes a route. app/settings/page.js is https://example.com/settings
- Use layout.js for shared ui in folder below its location (footers, headers, navbars, etc)
- Server vs Client components: Components are rendered on the server unless ‘use client’ is denoted at the top of a page. Has to be client to use many react features
- Built-in file conventions like loading.tsx and error.tsx



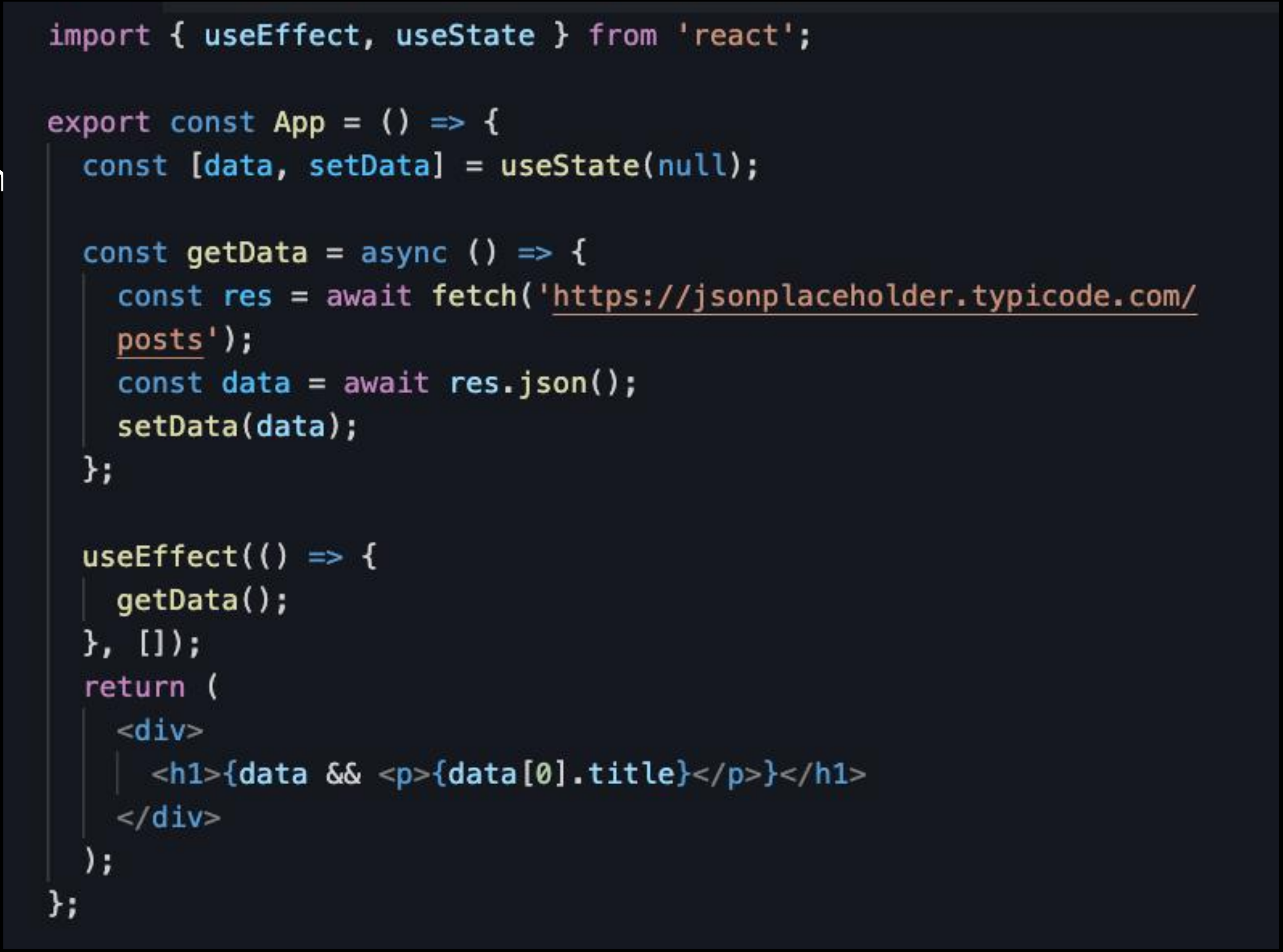
Next Pages

- This is a default home page that would show if you navigated to example.com/
- The page returns a react component which looks a lot like HTML

```
// app/page.js
export default function HomePage()
{
return (
<div>
<h1>Welcome to My Next.js App</h1>
<p>This is the home page.</p>
</div>
);
}
```

Next Pages

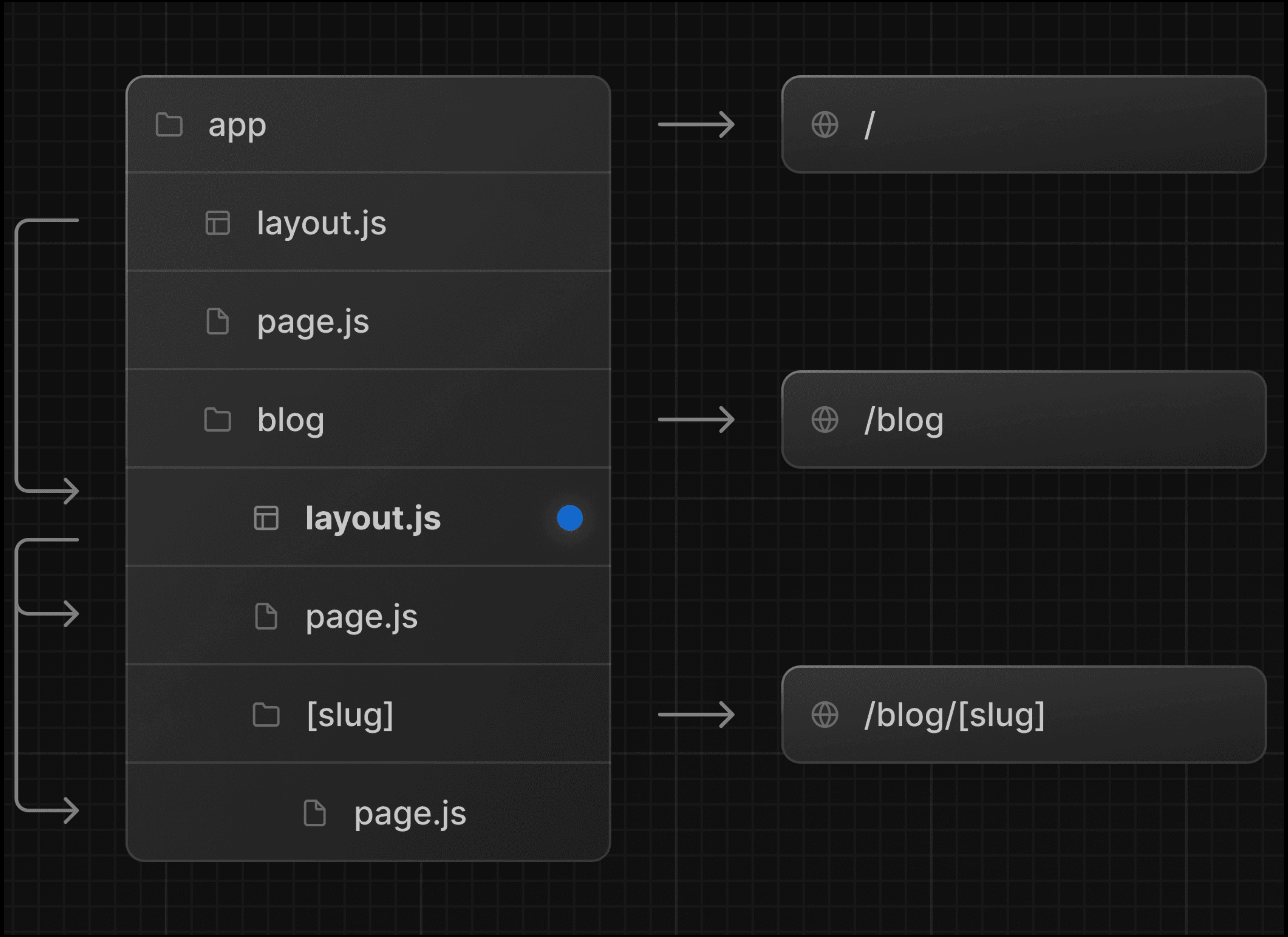
- This component is an example on how you would fetch data from an external source.
- There is an example component that fetch's coding resources from an api for the demo that is optional to implement



```
import { useEffect, useState } from 'react';

export const App = () => {
  const [data, setData] = useState(null);

  const getData = async () => {
    const res = await fetch('https://jsonplaceholder.typicode.com/
    posts');
    const data = await res.json();
    setData(data);
  };

  useEffect(() => {
    getData();
  }, []);
  return (
    <div>
      <h1>{data && <p>{data[0].title}</p>}</h1>
    </div>
  );
};
```

Next Layouts

- What It Is: A way to define a consistent structure for your website's pages in Next.js
- Why It Matters: Helps you reuse common elements (like headers, footers, menus) without rewriting code
- How It Works: You create layout files that wrap around your individual page content



Next Link/Navigation

- Two ways for client pages, using <Link> component. Or useRouter
- <Link> is a built-in component that extends the HTML <a> tag to provide prefetching and client-side navigation between routes. It is the primary and recommended way to navigate between routes in Next.js.
- The useRouter hook allows you to programmatically change routes from Client Components.

```
import Link from 'next/link'

export default function Page() {
  return <Link href="/dashboard">Dashboard</Link>
}
```

```
'use client'

import { useRouter } from 'next/navigation'

export default function Page() {
  const router = useRouter()

  return (
    <button type="button" onClick={() => router.push('/dashboard')}>
      Dashboard
    </button>
  )
}
```

Backend Basics

- Generally includes server-side logic, databases, and authentication
- There are many backend frameworks as well to help with development: Django, Ruby on Rails, Flask, Next.js
- Databases: Supabase, Pocketbase, Firebase, MySQL, MongoDB
- Can also include the implementation of APIs to communicate with frontend

Backend Basics

pocketbase.io

Pocketbase:
- Built on SQLite and Go
- Includes many ways for auth
- Self hosted (free to setup on azure with github student offer)

Supabase:
- Hosted for you on the cloud
- Free and paid plans
- Built on Postgres
- Authentication, instant APIs, Edge Functions, Realtime subscriptions, Storage, and Vector embeddings

supabase.com

Backend in Next.js

- Store API keys and secrets in environment variables
- Use Next.js API routes or server-side rendering to keep database logic secure
- Separate database queries from UI components for maintainability

api/users.js

```
import { createClient } from '@supabase/supabase-js';

const supabaseUrl = process.env.NEXT_PUBLIC_SUPABASE_URL;
const supabaseKey = process.env.SUPABASE_SERVICE_ROLE_KEY;
const supabase = createClient(supabaseUrl, supabaseKey);

export default async function handler(req, res) {
  const { data, error } = await supabase.from('users').select('*');
  if (error) return res.status(500).json({ error: error.message });
  res.status(200).json({ users: data });
}
```

In a client component/page

```
const [users, setUsers] = useState([]);

useEffect(() => {
  async function fetchUsers() {
    const res = await fetch('/api/users');
    const { users } = await res.json();
    setUsers(users);
  }
  fetchUsers();
}, []);
```

DEMO TIME:

tinyurl.com/53vwk6wc

Using AI

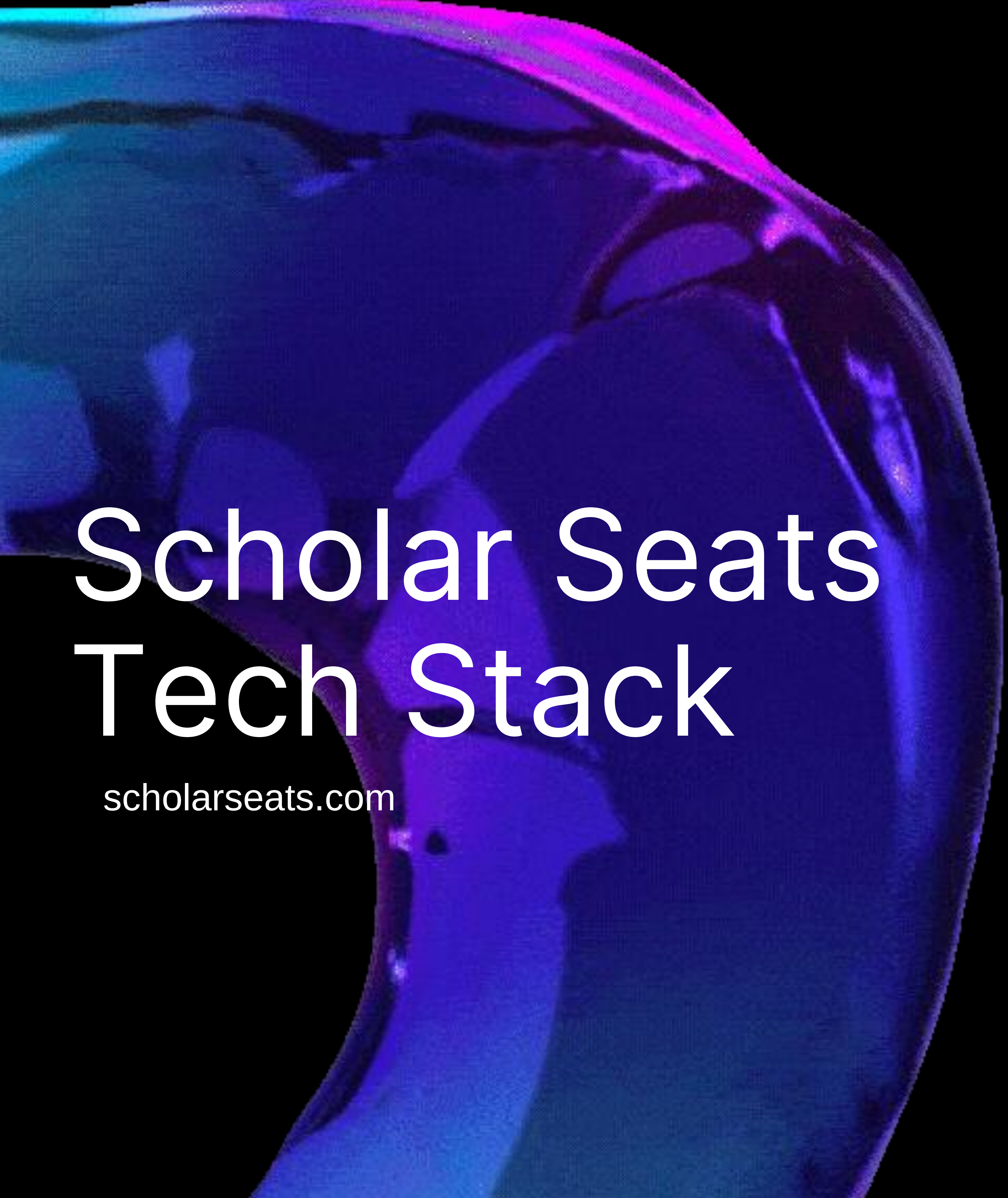
TAKE ADVANTAGE OF AI. IT IS A VALUABLE LEARNING TOOL

- AI can help you streamline your development process by giving you new ideas, helping you find bugs, and completing tedious tasks

- DO NOT expect AI to do everything for you. You will still need to understand the fundamentals well to create a good final product

- There are many useful AI tools out there with free plans: Cursor, Copilot, ChatGPT, V0.dev, Lovable.dev

- (apply for the GitHub student developer pack, you get many free tools, like copilot/github pro) education.github.com/pack



Scholar Seats Tech Stack

scholarseats.com

Next.js
Reach framework for front/backend

Pocketbase
For database and authentication

Vercel
hosting, domain, analytics

Idea to Product (Our Experience)

- Coming up with an idea can sometimes be the hardest part of the development process. Using the [product] for [niche] method can be helpful for coming up with ideas (e.g. Discord could be thought of as Slack for gamers)

- After a bit of back and forth we decided to use Next.js as our framework because of its easy hosting and built in features (also we just wanted to learn something new). We chose Pocketbase as our backend database because of its simple integration and because it was free for us (bc github student dev pack)

Idea to Product (Our Experience)

- During development of Scholar Seats we used AI tools like cursor and v0.dev to assist with getting the initial layout of the site. We also used ChatGPT to help us brainstorm initial ideas with layout and functionality.

- The most difficult problems we encountered were authentication/authorization implementation and figuring out how to market our website to new people.

Hackathon Tips

- Don’t spend all of your time deciding on which frameworks/languages to use. Just pick one you want to learn and move on to actually building something.

- You don’t have to build something revolutionary and that’s ok. Hackathons are typically only 1-2 days long and are meant to be a fun learning experience so don’t be discouraged if you can’t come up with the next big thing.

- It’s ok to take breaks! Taking breaks every once in a while is proven to help you from getting frustrated and burnt out. We’ve heard countless stories of people being stuck on something only to leave for a bit and come back and solve the issue instantly!