9 - Create Next App

00:12

Create Next.js App for UI

1. Create a Next.js App

Create a directory for the web client (aka website).

> We may want to also have a mobile app in the future, so we will call this directory `yt-web-client`

> instead of `ui`.

```
mkdir yt-web-client
cd yt-web-client
```

Create a new Next.js app in the current directory:

```
npx create-next-app@latest .
```

As of Next.js 13.4, we will answer the prompts as follows:

```
✔ Would you like to use TypeScript?Yes
✔ Would you like to use ESLint? Yes
✔ Would you like to use Tailwind CSS? No
✔ Would you like to use `src/` directory? No
✔ Would you like to use App Router? Yes
✔ Would you like to customize the default
import alias? No
```

2. Go over the Next.js App

The `package.json` shows the dependencies that were installed as well as the `npm` scripts available to us:

```
"scripts": {
  "dev": "next dev",
  "build": "next build",
  "start": "next start",
  "lint": "next lint"
},
"dependencies": {
  "@types/node": "20.4.1",
  "@types/react": "18.2.14",
  "@types/react-dom": "18.2.6",
  "eslint": "8.44.0",
  "eslint-config-next": "13.4.9",
  "next": "13.4.9",
  "react": "18.2.0",
  "react-dom": "18.2.0",
  "typescript": "5.1.6"
}
```

The `tsconfig.json` serves a similar purpose to the `tsconfig.json` in our `video-processing-service`.

The `eslintrc.json` is the configuration file for ESLint. ESLint is a tool for identifying and reporting on patterns found in ECMAScript/JavaScript code. It is used here to enforce code quality and consistency.

2.1 App Directory

The `app` directory is a new feature of Next.js, introduced in version 13.

The `favicon.ico` is the icon that appears in the browser tab. We will replace this with a **youtube logo you can download here**.

The `global.css` file is where we can add global styles.

The `layout.tsx` is the entry point for our app. Notice that this file imports the `global.css` file, and those styles are implicitly applied to all pages.
We can change the title of our app by changing the `title` prop of the `metadata` object. We will set this to `Youtube`, and update the description to `Youtube Clone`.
The `body` element renders all of the pages in our app (notice the `{children}` inside).

The routing is handled by Next.js by using file naming conventions.

For example, the `page.tsx` file is the entry point for the `/` route. This file imports some styles from `page.module.css` and applies it via the `className` attribute.

The reason we are not using `class` like we would in HTML is because this is a TypeScript file, and `class` is a reserved keyword in TypeScript. This is based on `jsx` which is a JavaScript variation of HTML. But since we're using TypeScript, our file extension is `.tsx` instead of `.jsx`.

In the video we delete all of the styles within `global.css` and `page.module.css`, and update our page.tsx to look like this:

```
import styles from './page.module.css'

export default function Home() {
  return (
```

Copy

Mark Lesson Complete

```
{styles.code}>app/page.tsx</code>
        </p>
      </div>
    </main>
  )
}
```

3. Run the Next.js App

To run the app, we can use the `npm run dev` command:

```
npm run dev
```

This will start the development server on port 3000.