Basic Routes

23 min

Up to this point, we've created an /app folder and added a page.tsx file to the folder. page.tsx is one of the many *reserved files* in Next.js. Recall that creating a folder does not *automatically* create a path that users can visit. We must add a page.tsx file to the folder and default export a React component to inform the App Router that the folder (path segment) is *accessible*.

Besides page.tsx, Next.js uses a layout.tsx named file to define a *shared* UI across any nested path segments. To define the UI, we default export a React component that accepts a prop called children of type ReactNode like so:

```
import { ReactNode } from "react";

// React component in layout.tsx
function MyLayout({ children } : { children: ReactNode }) {  // destructured props
  return (
    <div>
      <p>Nested UI's will always see me above them!</p>
      <section>
        {children}
      </section>
    </div>
  )
}


// Note: default export is required
export default MyLayout;
```

Due to layout.tsx being a shared UI, Next.js *requires* every application to have at least 1 layout.tsx in the topmost segment known as the root layout. This root layout *must* contain the <html> and <body> elements. An important aspect of layout.tsx is that they maintain their state across any nested segment navigation and do not re-render.

We've discussed a lot about nested segments, but how do we create one? To create a nested segment, we create a new folder (segment) inside another folder (segment). For example, let's look at the following folder structure:

```
 ├— app
|    ├— settings
|    |    ├— page.tsx
|    |    ├— billing
```

```
|   |   |   ├── page.tsx
|   |   ├── info
|   |   |   ├── MyComponent.tsx
|   |   ├── layout.tsx
```

The above folder structure:

- Contains the required, shared root layout.
- Contains two nested folders within /app, /settings, and /info.
- Contains an accessible, nested path (/settings/billing) by nesting the /settings and /billing segments.
- Contains an accessible path (/settings) by using the /settings segment.
- Contains an in-accessible path /info because its folder does *not* contain a page.tsx file.

Now, we'll practice creating basic and nested routes in the following checkpoints. For the rest of this lesson, you will be practicing what you've learned to complete the Codecademy article reader application.

Instructions

1. Checkpoint 1 Passed

1.

Let's begin by defining our root layout in our article reader Next.js application.

In /app, create a new file called layout.tsx and add the following code:

```
import type { Metadata } from 'next'
import { ReactNode } from 'react'
import '../globals.css'
import Footer from '../components/Footer'
import Navigation from '../components/Navigation'
export const metadata: Metadata = {
  title: 'Article Reader',
  description: 'Application to read codecademy articles',
}
```

Hint

Recall that every Next.js application must have at least one root layout in the top-most folder.

Make sure to create a reserved file called layout.tsx.

2. Checkpoint 2 Passed

2.

Next, default export a React component called RootLayout that receives a children prop and returns the <html> and <body> elements in RootLayout.

Hint

Make sure to default export a React component that receives a ReactNode type children prop. The component should return the <html> and <body> elements.

3. Checkpoint 3 Passed

3.

Next, render the following elements within <body>:

- <Navigation>
- children within a <main> element
- <Footer>

Hint

Make sure to return the children prop within the <main> element.

4. Checkpoint 4 Passed

4.

With the root layout in place, let's add an About page to our application.

Begin by creating a page.tsx file in the /about path.

Hint

Make sure to make your path folder accessible using a reserved file.

5. Checkpoint 5 Passed

5.

Finally, create an accessible view in /about that renders the following code:

```
<p>
  This application contains snippets from various Codecademy articles. You can browse all our articles and read them in their entirety <a
href="https://www.codecademy.com/articles">here</a>.
</p>
```

Run npm run dev. Once "Ready" appears, click the refresh button in the workspace browser.

If successful, you should be able to see the content by starting the dev server and navigating to localhost:4001/about.

Note: There will still be limited functionality in the application.

Hint

A path segment is made accessible when it contains a page.tsx file and default exports a React component.

Enter the command npm run dev inside the terminal window.

The window can be refreshed when it displays “Ready.”

```
> dev
> next dev -p 4001

  ▲ Next.js 14.0.1
  - Local:        http://localhost:4001

 ✓ Ready in 1156ms
```

layout.tsx

```
import type { Metadata } from 'next'
import { ReactNode } from 'react'
import '../globals.css'
import Footer from '../components/Footer'
import Navigation from '../components/Navigation'
import UrlBar from '../components/UrlBar/UrlBar'

export const metadata: Metadata = {
  title: 'Articler Reader',
  description: 'Application to read codecademy articles',
}

export default function RootLayout({
  children,
}: {
  children: ReactNode
```

```
}) {
  return (
    <html lang="en">
      <body>
        <Navigation/>
        <main>
          {children}
        </main>
        <Footer/>
      </body>
    </html>
  )
}
```

app/about/page.tsx

```
export default function AboutPage() {
  return (
    <p>
      This application contains snippets from various Codecademy articles. You can browse all our articles and read them in their entirety <a href="https://www.codecademy.com/articles">here</a>.
    </p>
  )
}
```