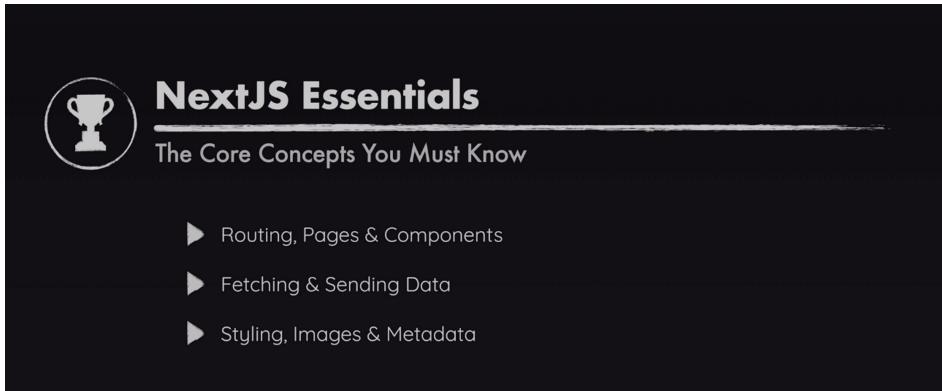
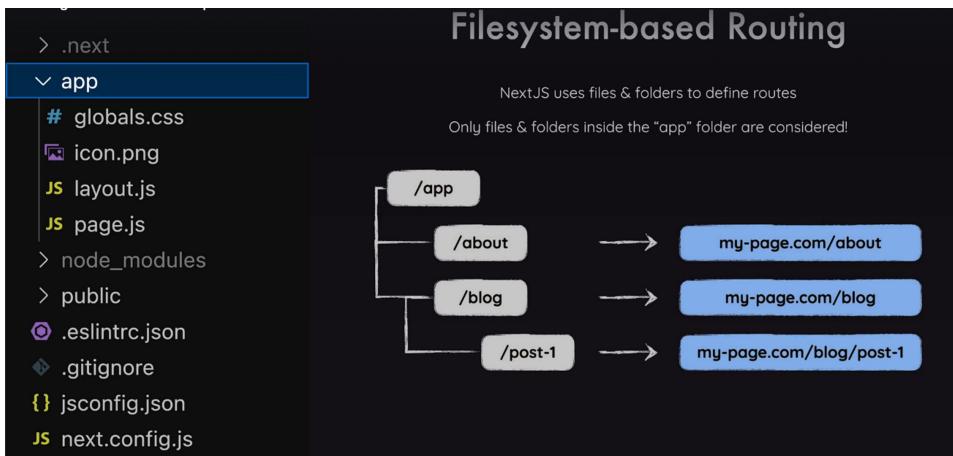


Section 3: NextJS Essentials (App Router)

08 September 2024 14:04



Understanding File-based Routing & React Server Components



A file named page.js simply tells Next.JS that it simply should render a page.

A screenshot of a code editor showing the content of page.js. The code defines a default export function Home() that returns a main component containing an image and a h1 header. A red arrow points from the bottom left towards the page.js file in the sidebar.

This is a so-called Server Component.

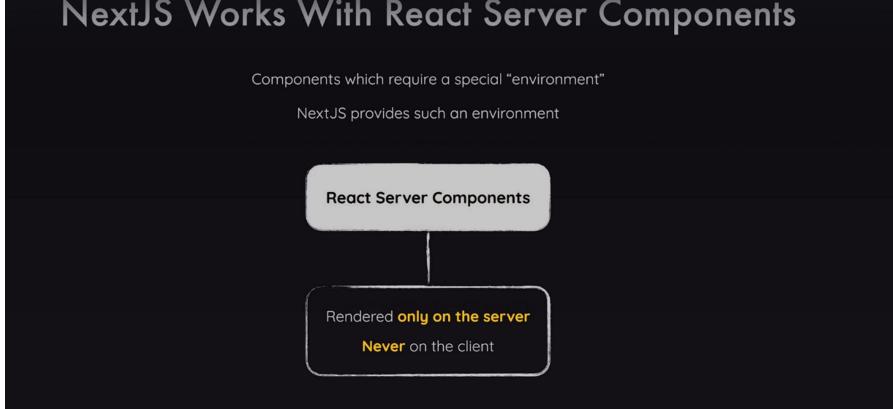
```
1, export default function Home() {  
2  
3,   console.log("Executing....");  
4  
5,   return (  
6,     <main>  
7,         
8,       <h1>Welcome to this NextJS Course!</h1>  
9,       <p>🔥 Lets's get started!</p>  
10,      </main>  
11,    );  
12}
```

→ Hence, it's a sever component, that is why in the browser, console log message won't appear.

```
✓ Compiled in 105ms (248 modules)  
  
✓ Compiled in 71ms (248 modules)  
Executing....  
✓ Compiled in 102ms (248 modules)  
Executing....  
✓ Compiled in 124ms (248 modules)  
Executing....
```

you can see this message on the vs code terminal (backend)

NextJS Works With React Server Components



Adding Another Route via the File System

The screenshot shows the file structure of a Next.js project. On the left, the file system tree includes .next, app (with about and page.js), node_modules, public, .eslintrc.json, and .gitignore. The right side features a title "Filenames Matter!" and a list of reserved filenames and their purposes:

- page.js → Define page content
- layout.js → Define wrapper around pages
- not-found.js → Define “Not Found” fallback page
- error.js → Define “Error” fallback page

At the bottom, it says "And others — covered later!"

The screenshot displays the content of a Home.js file. The code uses the `Link` component from the `next/link` package instead of standard `a` tags. The code is as follows:

```
export default function Home() {
  <p>💡 Lets's get started!</p>

  /* it's no Longer a single page application if we use <a></a>
   */
  {/* <p><a href="/about " >About Us</a></p> */}

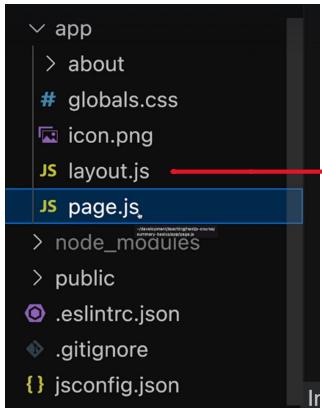
  <p><Link href="/about " >About Us</Link></p>
  </main>
}

//localhost:3000/
```

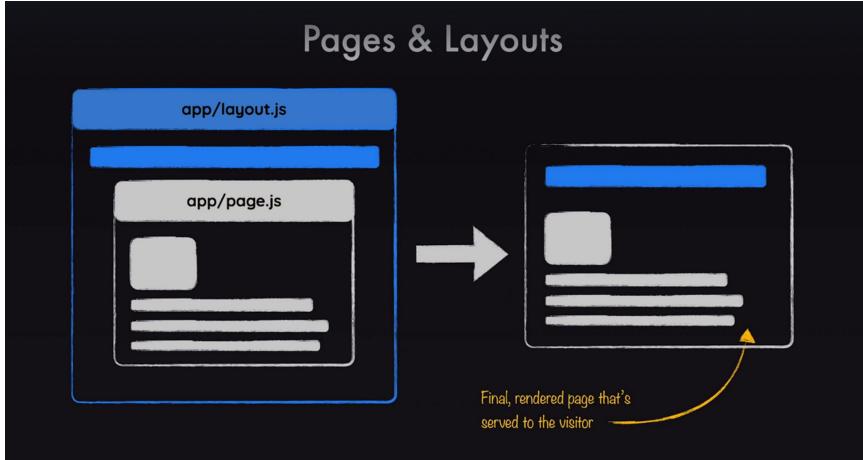
If we use `<a>`; each time it gets the data from the server and shows to us, in this way, our page reloads and we lost the single page application benefit.

But if we use LINK tag, it still loads the page from the server, but it'll then be sent to the client, and there it'll be handled by client-side javascript code to update what we see on the screen.

Working with Pages & Layouts



Where page.js file defines the content of a page, the layout.js file defines the shell around one or more pages.



Every Next.Js project needs at least one root layout.js file.

Reserved Filenames

As you already learned, there are some reserved filenames when working with NextJS.

Important: These filenames are only reserved when creating them inside of the app/ folder (or any subfolder). Outside of the app/ folder, these filenames are not treated in any special way.

Here's a list of reserved filenames in NextJS - you'll, of course, learn about the important ones throughout this section:

page.js => Create a new page (e.g., app/about/page.js creates a <your-domain>/about page)

layout.js => Create a new layout that wraps sibling and nested pages

not-found.js => Fallback page for "Not Found" errors (thrown by sibling or nested pages or layouts)

error.js => Fallback page for other errors (thrown by sibling pages or nested pages or layouts)

loading.js => Fallback page which is shown whilst sibling or nested pages (or layouts) are fetching data

route.js => Allows you to create an API route (i.e., a page which does NOT return JSX code but instead data, e.g., in the JSON format)

You also find a list with all supported filenames & detailed explanations in the official docs: <https://nextjs.org/docs/app/api-reference/file-conventions>

Exercise

Create three routes

/meals
/meals/share
/community

Optimizing Images with the NextJS Image Component

The screenshot shows the NextLevel Food application interface. On the left, there's a sidebar with a logo, 'NEXTLEVEL FOOD' text, and links for 'Meals', 'Share Meal', and 'Community'. The main content area has a dark background with the text 'Time to get started!'. On the right, the browser's developer tools are open, showing the DOM structure of the header. A red arrow points from the 'Meals' link in the sidebar towards the developer tools.

```
<Image src={logofImg} alt="A plate with food on it" />
NextLevel Food
```

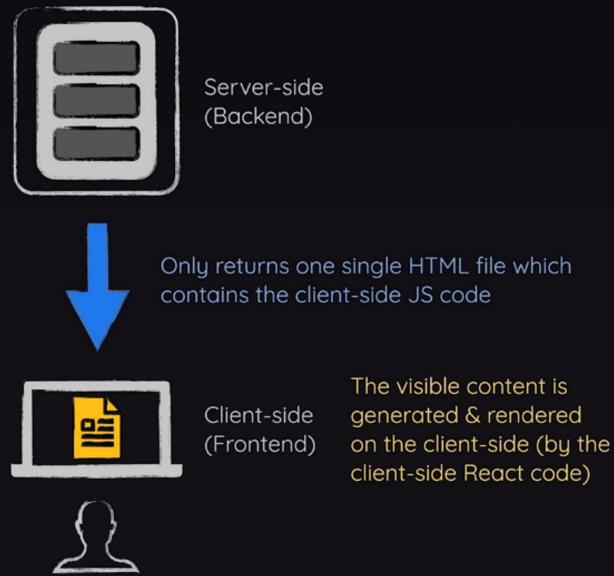
React Server Components vs Client Components - When To Use What



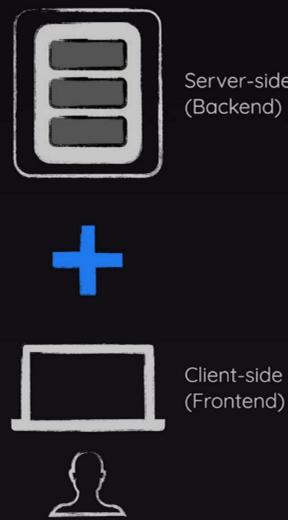
Now that's something that's super important in NextJS. NextJS knows Server components, React Server components and Client Components. And actually that's technically not just NextJS,

instead, React itself has this differentiation though in most React apps, in all those vanilla React apps which you are building with help of npx create React app or with help of Vite, you are using client components out of the box.

Vanilla React Apps Render On The Client



With NextJS, You Build Fullstack Applications

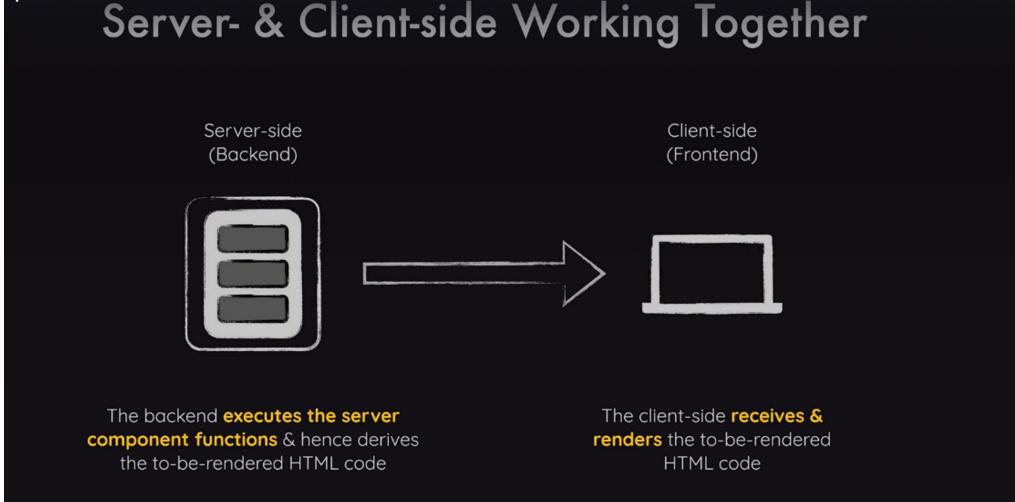


Because in those projects, React.js is a pure client side library, running code in the browser on the client.

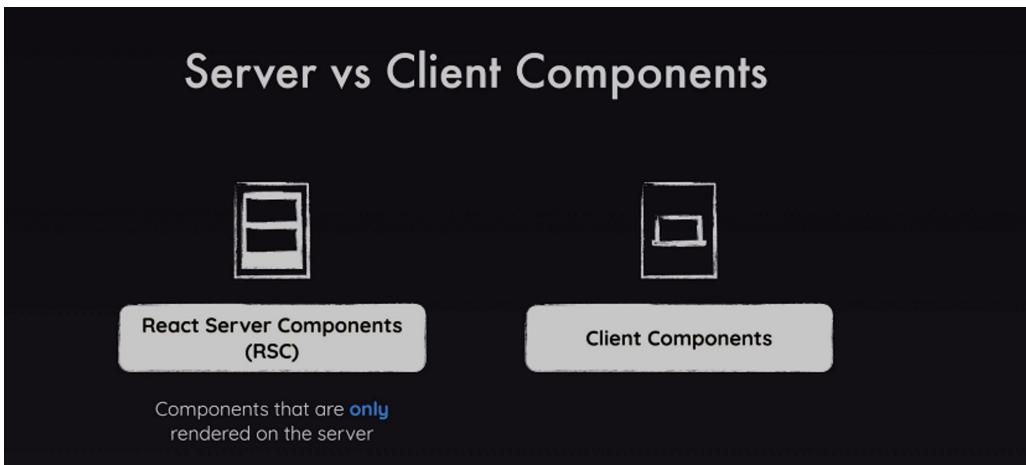
With NextJS, that changes because NextJS is a full stack framework. It has a backend, not just a front end,

and therefore code also executes on that backend when working with NextJS.

Server- & Client-side Working Together

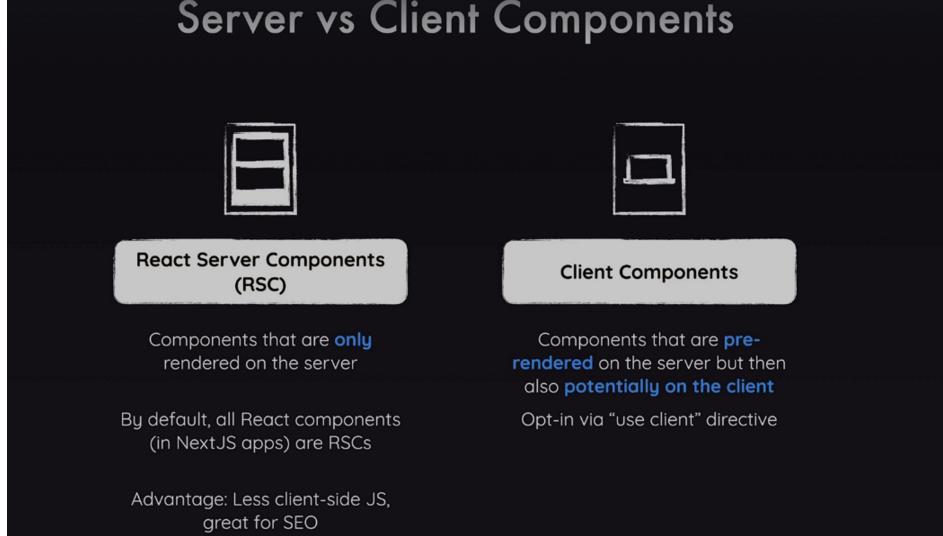


By default, all those React components you have in your NextJS project, no matter if they're pages, layouts or standard components are only rendered on the Server. That's why they are called React Server Components.



Now as mentioned, that's actually technically a feature built into React, but it must be unlocked, you could say, with a certain behind the scenes build process and structure, that's not part of most React projects. But it is unlocked and it is the standard in NextJS projects.

Server vs Client Components



Hooks/Events are not available on the server side component.

```
1   <ImageGrid padding={1} columns={3} colors="black white">
2     { images.map((image, index) =>
3       <Image key={index} alt={image.alt} />
4     );
5   }
6
7   export default function ImageSlideshow() {
8     const [currentImageIndex, setCurrentImageIndex] = useState(0);
9
10    useEffect(() => {
11      const interval = setInterval(() => {
12        setCurrentImageIndex((prevIndex) =>
13          prevIndex < images.length - 1 ? prevIndex + 1 : 0
14        );
15      }, 5000);
16
17      return () => clearInterval(interval);
18    }, []);
19
20    return (
21      <ImageGrid>
```

Suspense is a component that is provided by React that allows you to handle loading states and show fallback content until some data or resource has been loaded.