

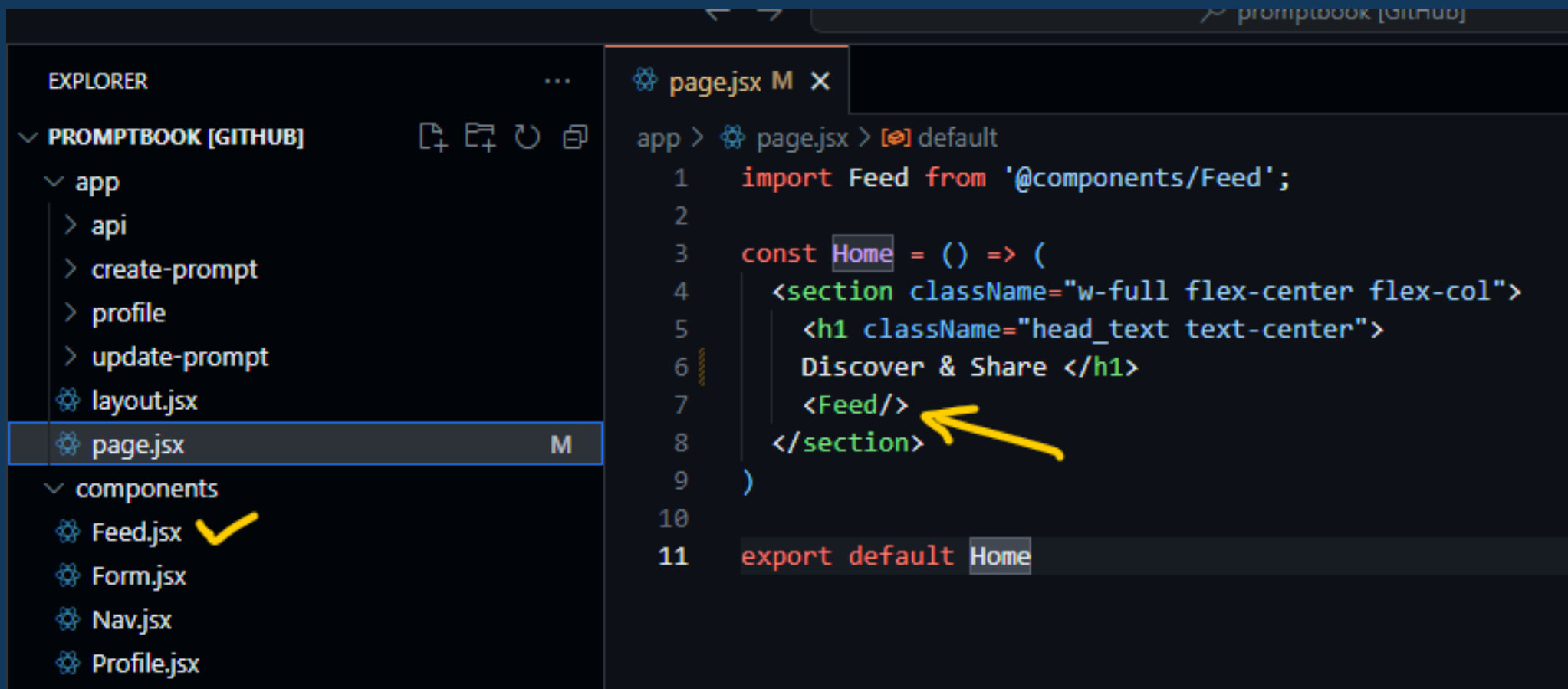


# Exploring Components In NextJS.

This post explores the concept of components in Next.js, highlighting their similarities and differences compared to React.js components:

SWIPE





```
1 import Feed from '@components/Feed';
2
3 const Home = () => (
4   <section className="w-full flex-center flex-col">
5     <h1 className="head_text text-center">
6       Discover & Share </h1>
7     <Feed/>
8   </section>
9 )
10
11 export default Home
```

# Building Blocks of Success: Components in Next.js

This introductory slide emphasizes the importance of components as reusable UI elements, as shown in picture that component Feed can be reused in homepage of our app.

KEEP SWIPING

The image shows two code snippets side-by-side, with a yellow arrow pointing from the left one to the right one. The left snippet is Next.js code for a page component, using an arrow function and returning a JSX element. The right snippet is React.js code for a component, using a function and returning a JSX element. Both snippets use the same JSX syntax for rendering an 

# element.

```
app > page.jsx > ...
1  const Home = () => (
2    <>
3      <h1>Home Page of Next</h1>
4    </>
5  )
6
7  export default Home
```

```
JS Home.js > ...
1  import React from 'react';
2
3  function Home(){
4
5    return(
6      <>
7        <h1>Home Of React</h1>
8      </>
9    )
10 };
11 export default Home
```

# Familiar Ground: Similarities with React.js Components

Next.js components share core principles with React.js components. Both utilize JSX syntax, lifecycle methods (if needed), and return a UI element. Hence the main difference between (React and Next)'s UI components is framework's structure.

KEEP SWIPING

```
page.jsx M X
app > page.jsx > getStaticProps
1  const Home = ({ posts }) => {
2    // Render the list of posts
3  };
4
5  export async function getStaticProps() {
6    // Fetch data from an API
7    const response = await fetch('https://api.example.com');
8    const posts = await response.json();
9
10   // Return the data as props
11   return {
12     props: { posts },
13   };
14 }
```

```
page.jsx M X
app > page.jsx > getServerSideProps
1  const HomePage = ({ posts }) => {
2    // Render the list of posts
3  };
4
5  export async function getServerSideProps() {
6    // Fetch data from an API
7    const response = await fetch('https://api.example.com');
8    const posts = await response.json();
9
10   // Return the data as props
11   return {
12     props: { posts },
13   };
14 }
```

Client side data fetching

Server side data fetching

# Next.js Exclusives: Data Fetching Power

Next.js components offer unique functionalities like `getStaticProps` and `getServerSideProps` for data fetching at build time or on the server, respectively. Where react itself is not directly capable of server-side rendering (SSR) like Next.js

ONE MORE

```
Products.jsx A X
components > Products.jsx > ...
1  import Card from '@components/Card';
2
3  const products = [
4    { id: 1, title: 'Product 1' },
5    // other products
6  ]
7
8  function ProductsPage() {
9    return (
10     <div className="products-container">
11       {products.map((product) => (
12         → <Card key={product.id} title={product.title} />
13       ))}
14     </div>
15   )}
16   export default ProductsPage;
17
```



# Pro Tip: Leverage Reusability!

The power of components lies in reusability. Create well-defined components in Next.js to keep your code organized, maintainable, and efficient.

For example you can see component Card is being reused in a map function to show card multiple time for every separate product.

COMMENT BELOW