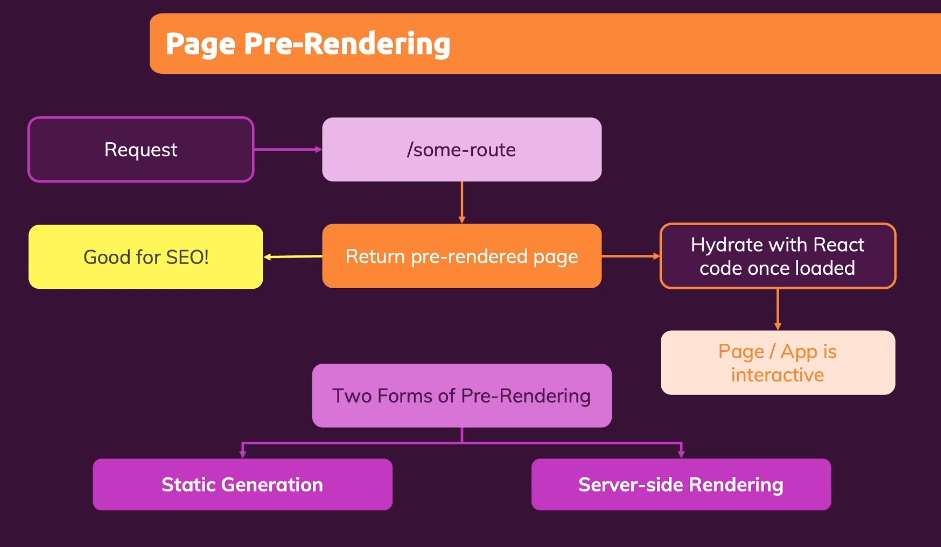
When a page loads, with just react app, source code will not have any details resulting in poor SEO. When using Next.js we can pre render a page. So when page loads source code contains some details



While having pre rendered page, we can also have react features.

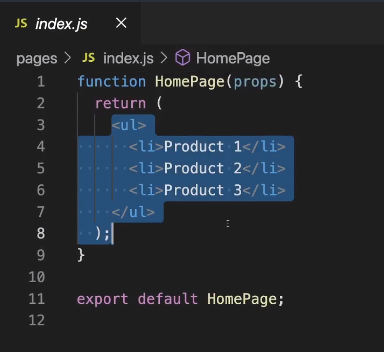
Static Generation: Pages are created with some data while doing a build.

Server-side Rendering: Pages are created with data when request is made.

**Static Generation with "getStaticProps”** (name matters)

Whatever code that is written inside this will only be executed by backend. Data will not be sent to client side. So we can also write security keys or DB keys inside this method. “window” object will not be available inside this.

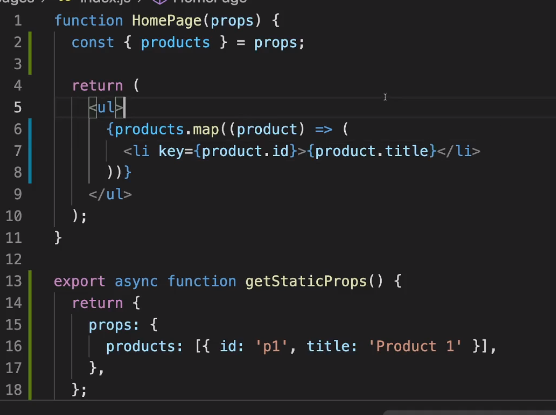


Next.js pre-renders by default.

See here since code is hard coded, next.js pre-renders content.

Using getStaticProps



Here in getStaticProps() we are using fs module, which is available only in Node environment and not in client evn. Here we are fetching the data from /data/dummy-backend.json path and sending that to HomePage component as props.

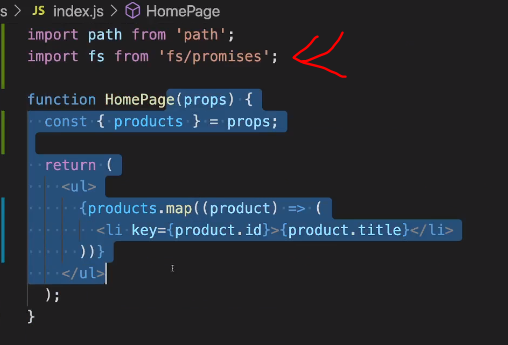
export async getStaticProps() {

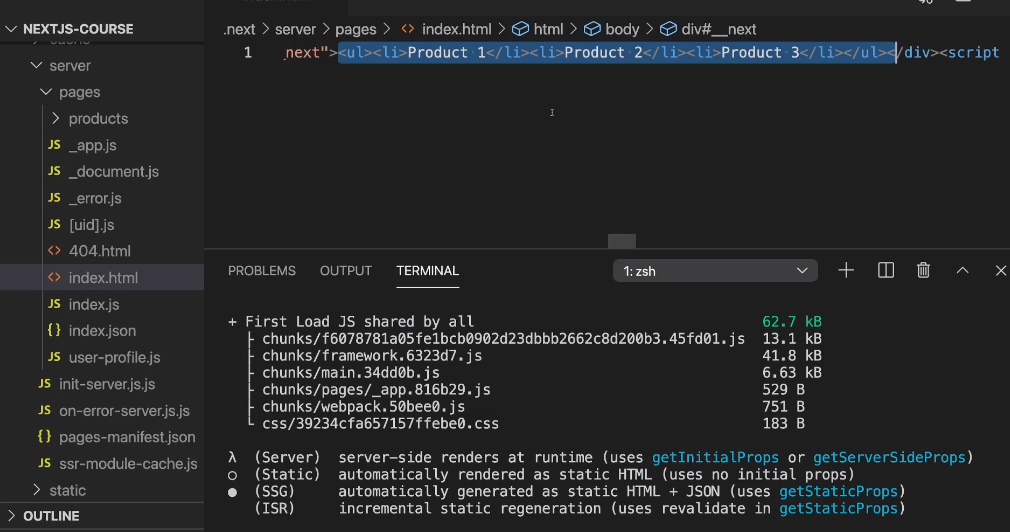
return {

//whatever is returned here will be sent as props

}

}



Once build is done. We can see page generated by next.js contains that details already.

npm run build 🡪 to do a build

npm start 🡪 to start production ready app

npm run dev 🡪 to start in dev mode

**Incremental Static Generation (ISR)**

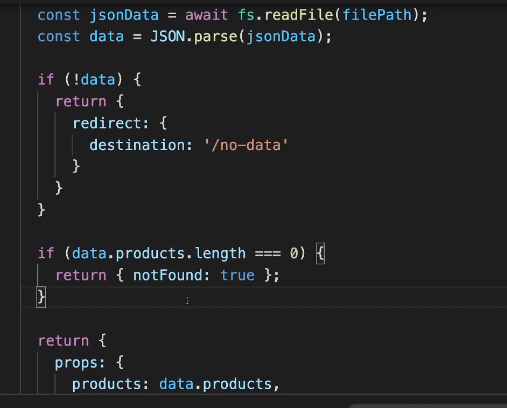
****

“revalidate” is the time in seconds after which the page is re-rendered again. We have to send a request for that page to rebuild again.

In dev mode, it will be build after every request, but in production it will after that X seconds.

Before 10 seconds, same page will be rendered. But after 10 secs, when new request is made new page is built and that page sent back to client. (In production only)

Couple of other parameters that we can pass as props to the components are



Here if data is not fetched, then we are returning

Redirect: { destination: ‘/no-data’ }

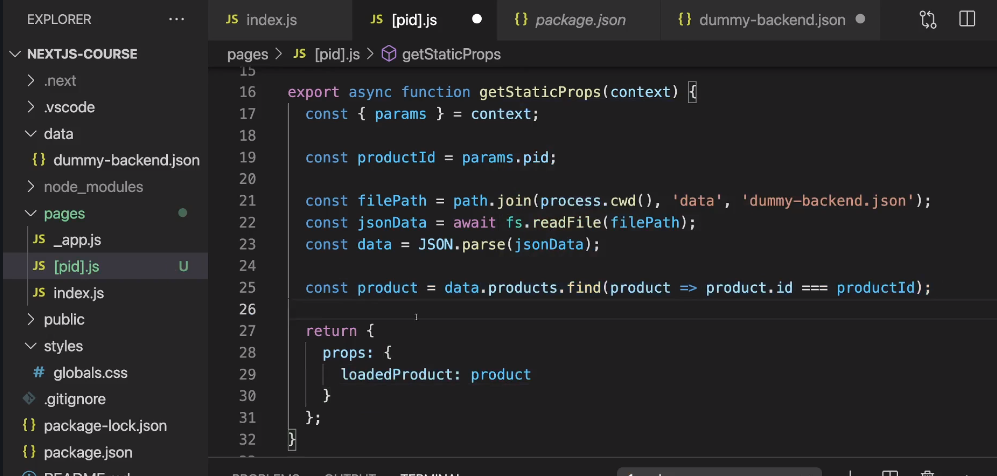
This will redirect to that page when data is not available.

If data is empty we can return a 404 page

“notFound” should be returned as “true”. Then 404 page will be shown instead of component.

“redirect” and “noFound” are 2 important return parameters

**Working with dynamic parameters**

****

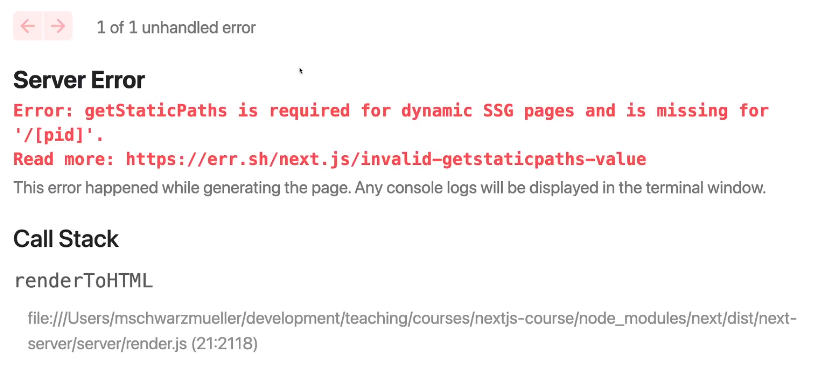
The above error is because, when getStaticProps is mentioned in a file, it makes sure that the page is re-rendered. But since it is a dynamic path, next.js doesn’t know how many pages it should render.

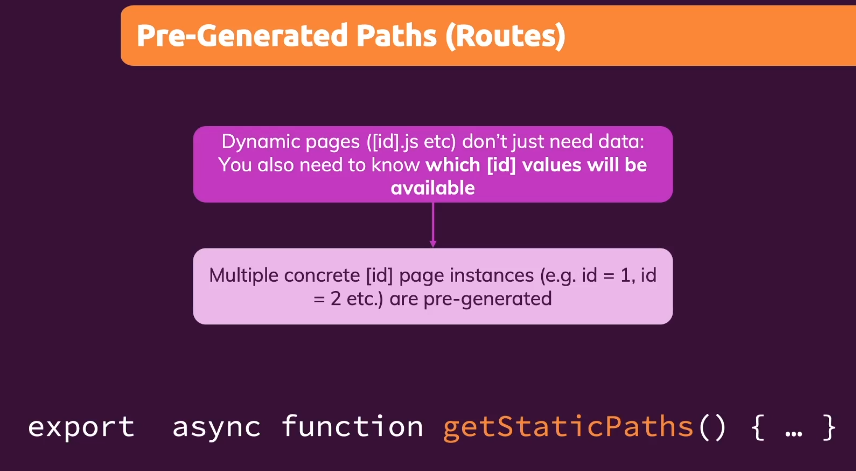
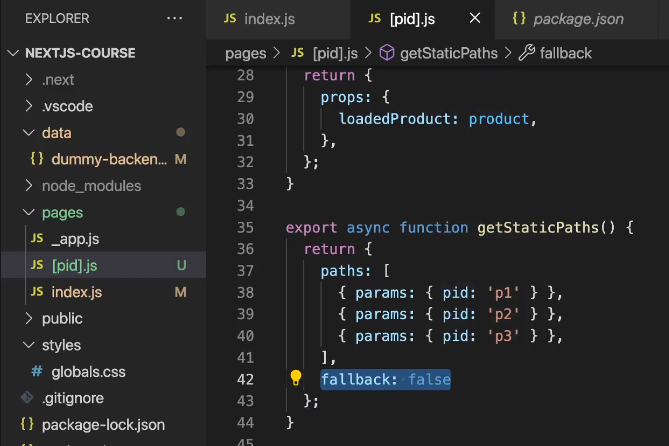
I here if we want to pre-render the page with a particular param then use “context” parameter provided inside getStaticProps()

“params” and “context” are names provided by next.js, they should not be changed.

Depending upon the value we can pre fetch the values and pre-render the page..

We get below error, for the left side code.



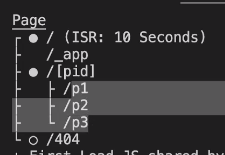
 

Here we are pre-defining the paths.

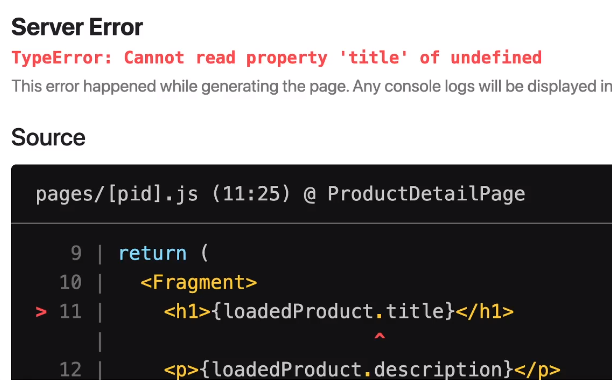
Pages with params: p1,p2,p3 will be pre-generated during build time.

fallback should be set to false.

After building these pages are built.

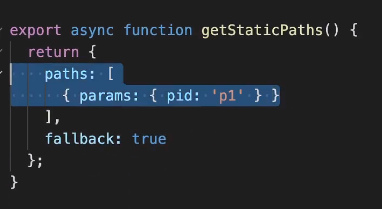


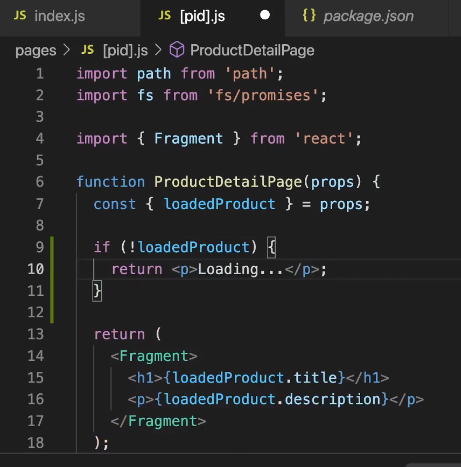
**“fallback” importance**



When fallBack is given as true, then we are telling next.js that there are other possible values that we can give to [pid]. Therefore we can render infinite number of pages.

This will work only if we are going to that page via link. If we directly go to that page by giving in URL, it will throw an error.

****

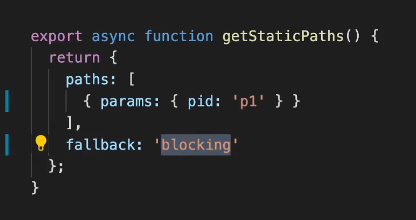


\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*IMPORTANT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Adding lines 9 to 11 will resolve the above issue.

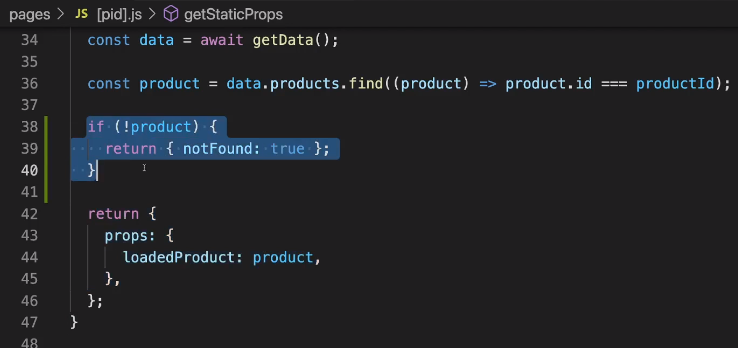
When pages are loaded directly by giving path in the URL, then it takes some time to get the details, till then Loading… will be shown. After that props will have the details and page will be shown

**Alternative Way**

By giving “fallaback” to “blocking”, we can remove that check. Now we will only show that page when the page is completely loaded. i.e. it will be rendered on server and sent back here.

This will take some time(more than previous method).



**NOTE:**

1. React hooks will not work inside “getServerSideProps” and “getStaticProps” and “getStaticPages” methods.

2. We cannot combine “getServerSideProps” and “getStaticProps”

3. We can combine client side data fetching with either “getServerSideProps” or “getStaticProps”

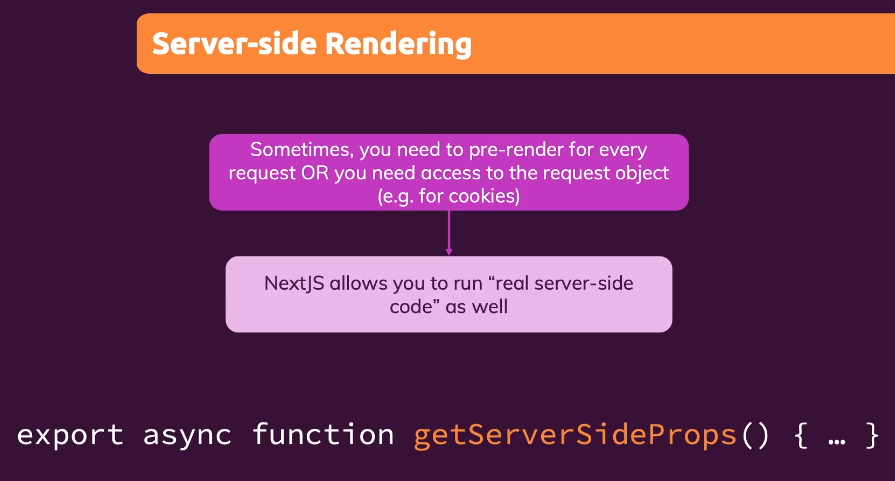
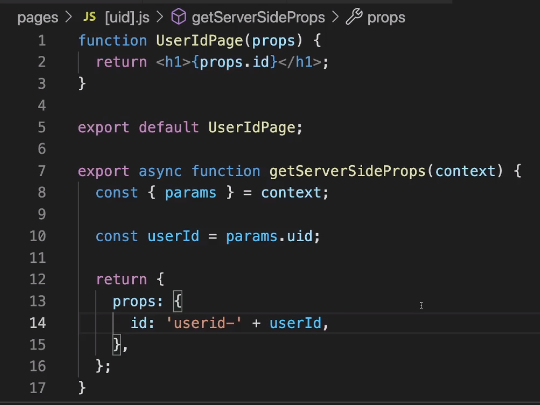
If we enter an url “http://lh3000/p4” then we return parameter “notFound: true”

So that if data is not there it will show 404 error page.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Server Side Rendering**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

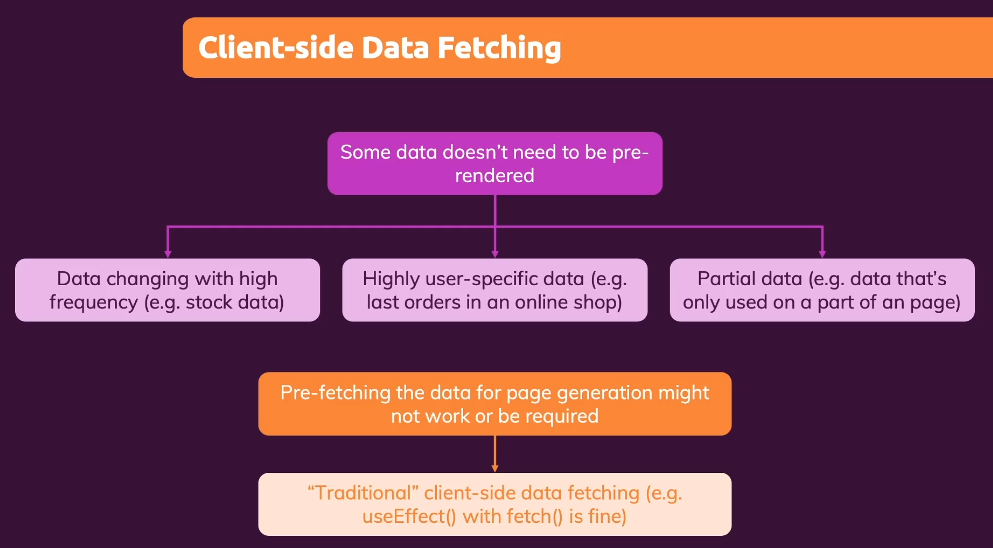
 

return values will be same as “getStaticProps”, except redirect will not be accepted.

The page will be rendered only once request is made.

“context” will contain “req” and “res” objects apart from params.

**Client side data fetching**



This is just like “fetch”, but contains lot of other features. It is like “Axios”.

This is normal data fetching. Just like in react, we use useEffect hook and use fetch to get some data and useState to set state and update the component.

**“useSWR” hook**

>npm install swr