⟶ In next js there is a by default routing enabled so once you create any file then you can just type the filename in url and you will get the page.

⟶ if you want to make a nested directories then you can create a folder and then you must have a `index.js` file.

⟶ you can create an error page by making a `404.js` file

⟶ if you want to redirect the user by clicking on any button or anchor tag then you can use `Link` which is by default provided by nextjs. For that you have to import link

⟶ if you want to enable dynamic routing then you can use `useRouter()` hook and then write like this :

```
const router = useRouter();
const urlText = router.query.filename // Without []
return (
    <p>you have entered {urlText}</p>
)
```

⟹ Note : the filename must be between `[]` for example : `[blogs].js`

⟶ if you want to redirect the user onClick then you can use `useRouter()` hook with the `push` method.

Code will be like this :

```
function Error() {

 const router = useRouter();
 const redirHome = () ⇒ {
    router.push("/home");
 };

 return (
 <>
     <h1>Page not found!</h1>
     <p>Chala jaa!</p>
     <Link href="/">back to home</Link>
     <button onClick={redirHome}>Click me</button>
 </>
 );

}
export default Error;
```

Redirect user after some seconds

⟶ If you want to redirect the user after some seconds once he visit the error page then you can use `useEffect()` hook and `setTimeout` to implement this.

code :

```
useEffect(() => {
     setTimeout(() => {
        router.push("/");
     }, 3000);
}, []);
```

⟶ This code will redirect the user after 3 seconds if he visits the error page

---

CSS

⟶ If you want to apply the css globally to all the components then you can use the `globals.css` file which you have to import only in `_app.js`

⟶ But if you want to make seperate css files for each component then the css file name must be like `[component_name].module.css`

NOTE : filename can be anything but it's preferred to start with `component_name` but `module.css` must be there after filename

After making the file, you have to import it like this :

```
// suppose we are making css file for home module
import styles from '../styles/home.module.css';
```

⟶ And then if you want to apply the classes of that css file then you can't directly specify the classname,so for that you have to write like `styles.[classname]`

Example :

```
<div className={styles.container}>This is a container</div>
```

⟹ But what if there are multiple classses in one element?

⟶ Then you can use template literals :

```
<div className={`${styles.container} ${styles.navbar}`}>This is
```

Inline css

⟶ If you want to use inline css in nextjs then you have to use `style` attribute and then pass the css in the form of object

Example :

```
<div style={{color:"green"}}>This is a div </div>
```

⟹ There is a one more way to use inline css which is using `<style>` tags with `jsx` attribute

Example :

```
<style jsx>
    {`
        div{
            color:red;
        }
    `}
</style>
```

⟶ In this type of inline css you can target by the tag name also which you can't do in other methods.

Images

⟶ There are 2 types of images : 1)Internal 2)external

Internal images

⟶ If you have images in your local machine and you want to display it on nextjs app then the image must be in `public` folder and there is a one component in next js which is `Image` for images

Example :

```
import Image from 'next/image'
```

⟹ Some important props :

You must need to specify the `src` `width` and `height` props in your `Image` component and if you don't want to specify `width and height` then you can use `layout="fill"`

⟶ and your `src` will start from `public` folder.

External images

⟶ If you want to show the external images which are coming from the different server then you have to make some configurations in `next.config.js` file

Like this :

```
images:{
    domains:["images.pixels.com","unsplash.com"]
}
```

and then you can use external images with `src` prop.

---

working with API

⟶ If you want to fetch some details from api then you can use `getStaticProps` function provided by nextjs

Example :

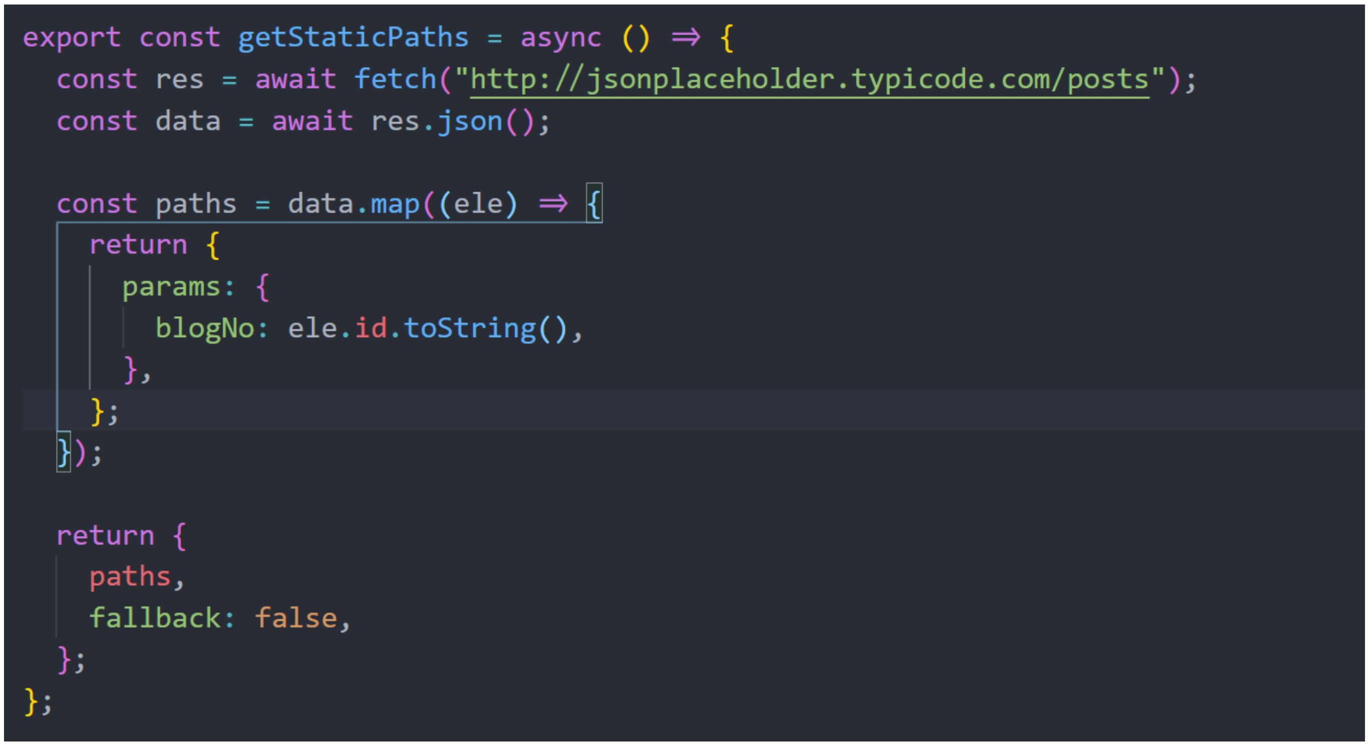
```
export const getStaticProps = async () ⇒ {

  const res = await fetch(`http://jsonplaceholder.typicode.com/po
  const data = await res.json();
  return {
      props: {
          data: data,
      },
  };
};
```

⟶ you must pass the `props` parameter while returning the data in this function.

⟹ If you want user to redirect on the details of the specific product by api then you can use `getStaticPaths` which will return the `id` from which we can fetch the data for specific object.

Example :



```
export const getStaticPaths = async () ⇒ {
  const res = await fetch("http://jsonplaceholder.typicode.com/posts");
  const data = await res.json();

  const paths = data.map((ele) ⇒ {
    return {
      params: {
        blogNo: ele.id.toString(),
      },
    };
  });

  return {
    paths,
    fallback: false,
  };
};
```

⟶ Here we must need to specify 2 params `paths` and `fallback`. so for getting paths we will map through the data and store the id in `blogNo` which is the filename in `[]`(it must be same otherwise it will not work) insie `params` property.

⟹ And one more thing is that you have to convert that id into string using `toString()` function.

⟶ Currently we don't need `fallback` so we will specify `false`

⟶ Here also we need to make `useStaticProps` function to fetch the data by id but here we will pass one prop which is coming from `useStaticPaths` which is `context`.

Now we will destructure it.

```
const id = context.params.blogNo;
```

```
export const getStaticProps = async (context) => {
  const id = context.params.blogNo;
  const res = await fetch(`http://jsonplaceholder.typicode.com/posts/${id}`);
  const data = await res.json();
  return {
    props: {
      data: data,
    },
  };
};
```

⟶ Now you can specify this `id` in `fetch` function.

and at last you will get the `data` as prop in component function so just display it on dom using properties like `data.id` , `data.title` and `data.body`

```
const BlogNo = ({ data }) => {
  // const router = useRouter();
  // const urlText = router.query.blogNo; //to get the text from url
  return (
    <>
      <Navbar />
      <div>
        <h2>{data.title}</h2>
        <p>{data.body}</p>
      </div>
    </>
  );
};

export default BlogNo;
```

---

app.js

⟶ If you want to show the navbar on all the components then you can use the built in file `_app.js` to add Navbar component into it.

⟶ You just have to make one more component and for example it's `Layout.js`. After that you have to do like this in `_app.js`

```
import "../styles/globals.css";
import Layout from "./Components/Layout";

function MyApp({ Component, pageProps }) {
 return (
     <>
         <Layout>
            <Component {...pageProps} />
         </Layout>
     </>
 );
}

export default MyApp;
```

⟶ And then add this code in `Layout` component

```
import React from "react";
import Navbar from "./Navbar";

const Layout = ({ children }) => {
 return (
     <>
         <Navbar />
         <main>{children}</main>
     </>
 );
};

export default Layout;
```

$\longrightarrow$ And now navbar is added to all the components. So you don't need to add `<Navbar/>` in every component

$\Longrightarrow$ Here, whole `_app.js` is in `children`