NEXT JS

- React framework
- Framework – Library is small and framework is whole package with more features and file based routing(pages/index.js)
  No need to use routing like react
  React is only a library
- For production- production ready apps(directly host to server)
- Fast
- Build hybrid apps – that run on all OS
- Automatic static optimization- static and dynamic are one
- High SEO (Search Engine optimisation) bcoz of server side rendering nd static site generator
- High SEO means people can see that site more on top so more traffic
- In react app when we see view page source that we can only see div id=”root” so it shows no content therefore its SEO is less but in Next we can see the whole html content
- Better UX
- CSS for each pages
- Image optimisation (next/image)
- API Route(pages/api/user.js)
- TSX
- Fast refresh

Requirements:

- Node JS
- VS code

Create 1st app- npx create-next-app appname

To run- yarn dev or npm run dev

Server on localhost 3000

Folder structure:

- pages/api/hello.js
- pages/_app.js – entrypoint of app
- pages/index.js
- no index.html in public unlike react
- styles folder- for css

- gitignore- those folders which are not important
- imgs in public folder
- _app.js takes Component and pageProps automatically and returns the component that is made in index.js. That's why we are able to see the content on browser without index.html (that was there in React) file also

_app.js

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

export default MyApp
```

index.js

```
import Image from 'next/image'
const index = () => {
  return (
    <>
    <p align="center">
    <h1>Hello World</h1>
    <Image src="/hello.jpg" alt="hello" layout='fill'/>
    </p>



    </>

  )
}

export default index
```

- **If we don't have index.js file, 404 error is shown**
- `layout='fill'` - for displaying img in full screen
- **File based routing-** to access a file in pages folder, only use the file url in browser Eg: to see the content in pages/home.js: url in browser will be localhost:3000/home
- **If we add a file in a folder in pages folder-** Eg: pages/blog/index.js: to access this url is localhost:3000/blog But if we change the index.js file name to any other we cannot access it(404 error)

Nested file based routing:

- Pages/blog/blog1 – url will be localhost:3000/blog/blog1
- Pages/blog/index.js- url will be localhost:3000/blog
- If url is wrong it shows 404

Dynamic Routing:

- By making a single file with [] we can do dynamic routing
- Eg: to access blog500 inside blog, we cannot make 500 blog files, so for that we can do dynamic routing
- Eg: we make a file [pgno].js in blog folder. To access this, url on browser will be localhost:3000/blog/anything. Anything means we can write anything here.

```
import {useRouter} from "next/router"

const pageno = () => {
    const router=useRouter();
    const pg=router.query.pageno;
  return (
    <>
    <h1>My {pg} content</h1>
    </>
  )
}

export default pageno
```

- We use useRouter hook to display the same msg as user enters in url

Creating NavBar:

```
const index = () => {
  return (
    <>
    <nav>
      <ul>
        <li>
          <a href="/home">Home </a>
        </li>
        <li>
          <a href="/about">About </a>
        </li>
        <li>
          <a href="/contact">Contact </a>
        </li>
      </ul>
    </nav>
    </>
  )
}

export default index
```

- As we are using <a> (as it is html tag), browser refreshes whenever we route to any page. To overcome refreshing, we use <Link> and its there in next/link

```
import Link from 'next/link'
const index = () => {
  return (
    <>
    <nav>
      <ul>
        <li>
```

```
            <Link href="/home">
            <a >Home </a>
            </Link>
          </li>
          <li>
            <Link href="/about">
            <a >About </a>
            </Link>
          </li>
          <li>
            <Link href="/contact">
            <a >Contact </a>
            </Link>
          </li>
        </ul>
      </nav>
      </>
    )
  }

  export default index
```

Customising 404 error pg: By creating a 404.js file in pages folder, if we change the name from 404.js to any other, then default 404 error msg will be shown

Go back to home page automatically after 5 seconds from 404 page: use useEffect and userRouter hook

```
import Link from 'next/link'
import { useEffect } from 'react'
import { useRouter } from 'next/router'
const error = () => {
  useEffect(() => {
    setTimeout(()=>{
      router.push("/")
    },5000)

  }, [])
  const router=useRouter()

  return (
    <>
    <nav>
      <ul>
        <li>
          <Link href="/home">
          <a >Home </a>
          </Link>
        </li>
        <li>
          <Link href="/about">
          <a >About </a>
          </Link>
        </li>
        <li>
          <Link href="/contact">
          <a >Contact </a>
          </Link>
```

```
        </li>
      </ul>
    </nav>
    <h1>This is error</h1>
    </>
  )
}

export default error
```

Components in Next Js :

- same as React
- Eg: if we want Navbar in all pages, we create a Navbar component and use it in all files by importing it and using it by <Navbar/>

CSS in NextJS:

- 3 types of CSS: Internal, external and inline
- For external CSS, change in globals.css file and import it in pages/_app.js. This style will be applicable to all pages
- For styles in some particular files, make a css file with convention name.module.css and import that in that file. Name here means name of file on which css is to be applied

  index.js:

```
import Link from 'next/link'
import style from '../styles/index.module.css'
const index = () => {
  return (
    <>
    <nav className='navs'>
      <ul className={style.ul}>
        <li>
          <Link href="/home">
          <a >Home </a>
          </Link>
        </li>
        <li>
          <Link href="/about">
          <a >About </a>
          </Link>
        </li>
        <li>
          <Link href="/contact">
          <a >Contact </a>
          </Link>
        </li>
      </ul>
    </nav>
    </>
  )
}
```

```
•
•   export default index
```

index.module.css:

```
.ul{
    color:blue
}
```

- For component level or file level CSS(Internal CSS), if we have more than 1 style

```
<a className={`${style.home} ${style.home_white}`}>Home </a>
```

- Inline CSS: by using style property Eg: <h1 style={{color:"green"}}>Hello</h1>
- Internal or component level CSS by Styled JSX: use style jsx tag and {``}

```
•    <h1>contact</h1>
•       <p className='apa'>whatsapp at 9811683711</p>
•         <style jsx>
•           {`h1{
•             color:blue
•           }
•             .apa{
•               color:pink
•             }`
•           }
•           </style>
```

Images in Next JS:

- **External and internal images**
- **Internal images:** Place img in public folder and use Image component from next/image with src, height, width,alt attributes or layout='fill'.

```
•   import Navbar from "../components/Navbar"
•   import Image from "next/image"
•   const home = () => {
•     return (
•       <>
•       <Navbar/>
•       <p align="center">
•       <h1>Home page</h1>
•       <Image src="/hello.jpg" height="300" width="400"></Image>
•       </p>
•       </>
•     )
•   }
•
•   export default home
```

- **External images:** Copy the url of img and paste it in src attribute and mention its hostname in next.config.js. After mentioning domain in next.config.js, we need to restart our app.

```
import Navbar from "../components/Navbar"
```

```
import Image from "next/image"
const about = () => {
  return (
    <>
    <Navbar/>
    <h1>about</h1>
    <Image
    src

="https://www.google.com/imgres?imgurl=https%3A%2F%2Fst.depositphotos.com%2F1038076%2F4870%2Fi%2F600%2Fdepositphotos_48701865-stock-photo-about-us.jpg&imgrefurl=https%3A%2F%2Fdepositphotos.com%2Fstock-photos%2Fabout.html&tbnid=B38uZ_ilEdvBZM&vet=12ahUKEwiq1eqGoI72AhVgyKACHev-AJcQMygAegUIARDbAQ..i&docid=xqO9mMU403FscM&w=600&h=436&q=about%20image&ved=2ahUKEwiq1eqGoI72AhVgyKACHev-AJcQMygAegUIARDbAQ"
    alt="about" height="400" width="400"></Image>
    </>
  )
}

export default about
```

next.config.js:

```
/** @type {import('next').NextConfig} */
const nextConfig = {
  reactStrictMode: true,
  images:{
    domains:["www.google.com"]
  }
}

module.exports = nextConfig
```

- To kill processes running on multiple ports: npx kill-port 3000 3001

Head component and how to increase SEO

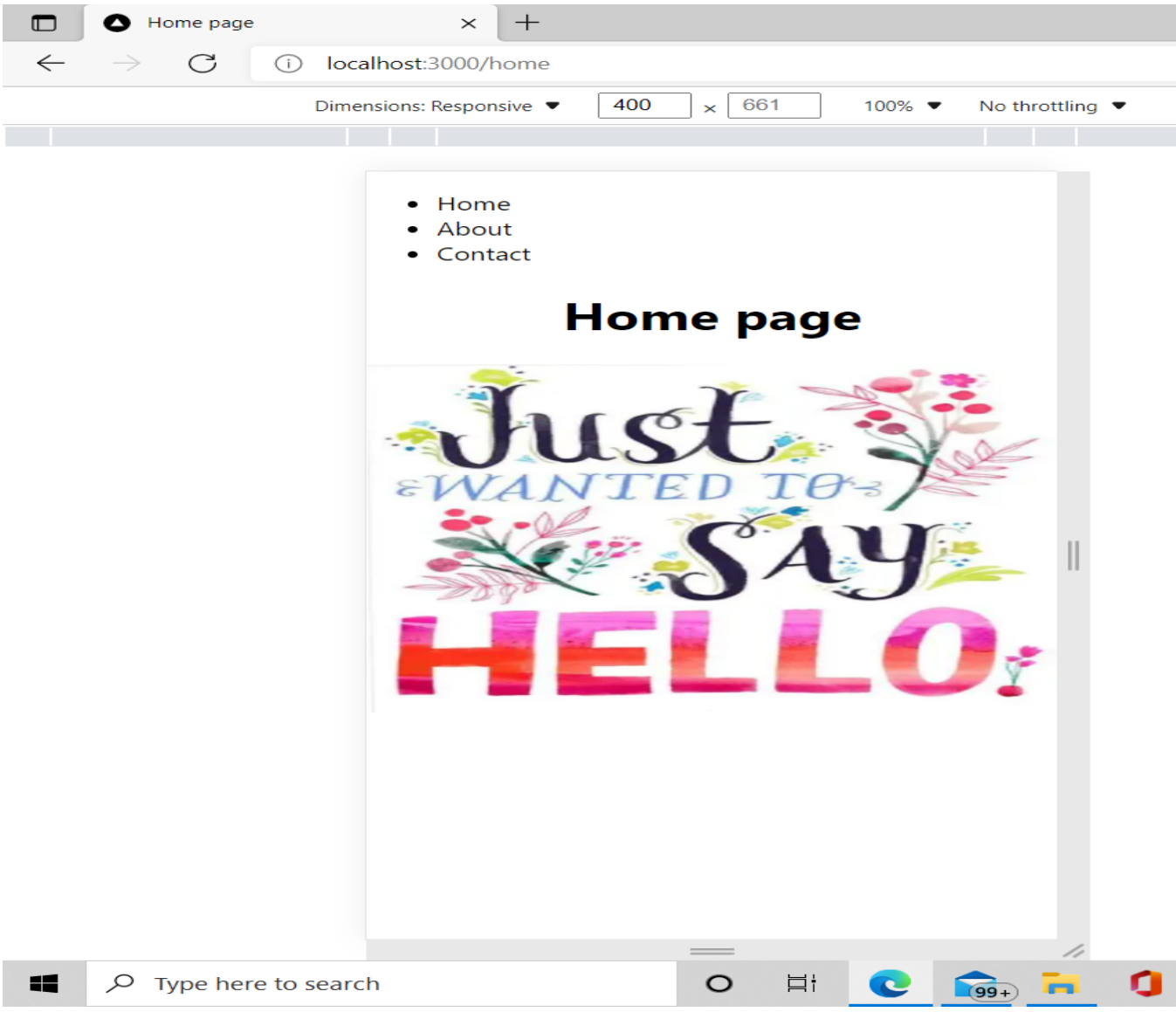
- In react, we can change the title of app from index.html. But in next js we donot have index.html file.
- To change the title of app, we have Head component inside next/head. So to change the title, we can make a title tag inside Head component
- To increase the SEO, add meta tags that tells page description, keywords, author of the document, and viewport settings. Metadata will not be displayed on the page, but is machine parsable.

```
import Navbar from "../components/Navbar"
import Image from "next/image"
import Head from "next/head"
const home = () => {
  return (
    <>
    <Head>
      <title>Home page</title>
      <meta charset="UTF-8"/>
    <meta name="description" content="Next JS"/>
    <meta name="keywords" content="HTML, CSS, JavaScript, Next js"/>
    <meta name="author" content="Aparna"/>
    <meta name="viewport" content="width=device-width, initial-scale=1.0"></meta>
```

```
      </Head>
      <Navbar/>
      <p align="center">
      <h1>Home page</h1>
      <Image src="/hello.jpg" height="300" width="400"></Image>
      </p>
      </>
    )
  }

  export default home
```

On localhost, instead of localhost:3000, we see Home page as title and meta tags increase the SEO



Data fetching from API

- Use getStaticProps function that uses async await to have fetch api and converts in json. Then to display each element we use loop(map).

- If we donot return props it gives error. So getStaticProps always return some props.

```
import Navbar from "../../components/Navbar"
export const getStaticProps=async()=>{
  const res=await fetch('https://jsonplaceholder.typicode.com/posts')
  const data=await res.json()
  return{
    props:{
      data
    }
  }
}
const index = ({data}) => {
    return (
      <>
      <Navbar/>
      {data.map((curElem)=>{
        return(
          <div key={curElem.id}>
          <h3>{curElem.id}</h3>
          <h3>{curElem.title}</h3>
          <h6>{curElem.body}</h6>
          </div>
        )
      })}
      </>
    )
  }

  export default index
```

- To get first 5 records only : use slice method that takes 2 args: first is the starting index and last index. (0,5) means it displays from 0 to 4

```
{data.slice(0,5).map((curElem)=>{
```