

Curso de Nextjs

Diego Saavedra

Jul 12, 2024

Table of contents

1	Bienvenido	4
1.1	¿De qué trata este curso?	4
1.2	¿Para quién es este curso?	4
1.3	¿Cómo contribuir?	5
I	Unidad 1: Introducción a Nextjs	6
2	Introducción	7
3	¿Qué es Next.js y por qué aprenderlo si quieres ser frontend senior?	9
3.1	Introducción a Next.js y sus beneficios.	9
3.2	Comparación con otros frameworks.	9
3.3	Casos de uso y ejemplos de proyectos exitosos.	10
3.4	Conclusión.	10
4	Arquitectura de un proyecto de Next.js	11
4.1	Estructura de carpetas y archivos.	11
4.2	Configuración inicial de un proyecto.	12
4.3	Revisión de un proyecto ejemplo.	12
5	Herramientas y stack utilizado en el curso	14
5.1	Herramientas necesarias para el curso.	14
5.2	Configuración del entorno de desarrollo.	14
5.3	Instalación y configuración de dependencias.	14
5.4	Iniciar el servidor local.	15
5.5	Acceder a la aplicación en el navegador.	15
6	Cómo crear rutas en Next.js	16
6.1	Rutas básicas.	16
6.2	Rutas dinámicas.	16
6.3	Nested routes.	16
6.4	Rutas con parámetros.	16
6.5	Rutas con query strings.	16
6.6	Rutas con rutas anidadas.	17
6.7	Rutas con rutas anidadas y parámetros.	17
6.8	Rutas con rutas anidadas y query strings.	17
6.9	Conclusión.	17
6.10	Ejercicios.	17

7	Cómo crear Layout en Next.js	18
7.1	Definición de layouts.	18
7.1.1	Componente Layout:	18
7.1.2	Componente Index :	19
7.1.3	Componente Layout específico:	19
7.1.4	Componente About :	20
7.2	Uso de layouts globales y específicos.	20
7.2.1	Layout global:	20
7.2.2	Layout específico:	21
7.3	Conclusión.	21
7.4	Ejercicios.	21
7.5	Implementación de layouts en un proyecto.	22
8	Cómo funciona la navegación en Next.js	23
8.1	Link component.	23
8.2	Uso de Router.	23
8.3	Navegación programática.	24
8.4	Conclusión.	24
8.5	Ejercicio.	24
9	Manejo de parámetros en rutas en Next.js	25
9.1	Parámetros de ruta.	25
9.2	Parámetros de consulta (query params).	25
9.3	Uso de useRouter para acceder a los parámetros.	26
9.4	Conclusión.	27
9.5	Ejercicios.	27
10	React Server Components en Next.js: notación “use Client”	28
10.1	Introducción a React Server Components.	28
10.1.1	¿Qué es use Client?	28
10.2	Uso de la notación “use Client”.	28
10.3	Ventajas de la notación “use Client”.	29
10.4	Ejemplos prácticos.	29
10.5	Conclusión.	30
10.6	Ejercicios.	30
11	Manejo de estilos y archivos estáticos en Next.js	31
11.1	Estilos en Next.js	31
11.1.1	CSS global	31
11.2	CSS Modules en Next.js	32
11.2.1	CSS Modules	32
11.2.2	Scoped CSS en Next.js	32
11.2.3	Creación y uso de CSS Modules.	32
11.3	Styled JSX en Next.js	33
11.3.1	Styled JSX	33
11.3.2	Ventajas de Styled JSX	34
11.4	Conclusión	34
11.5	Ejercicios	34

1 Bienvenido



Figure 1.1: Typescript

¡Bienvenido al Curso de Typescript!

En este curso, exploraremos todo, desde los fundamentos hasta las aplicaciones prácticas.

1.1 ¿De qué trata este curso?

Este curso es una introducción a TypeScript, un lenguaje de programación de código abierto desarrollado y mantenido por Microsoft. TypeScript es un superconjunto de JavaScript que agrega tipado estático opcional y otras características avanzadas a JavaScript.

En este curso, aprenderá los conceptos básicos de TypeScript, incluidos los tipos de datos, las funciones, las clases, los módulos y mucho más. También explorará cómo TypeScript se puede utilizar para crear aplicaciones web modernas y escalables.

Este curso es ideal para principiantes y aquellos con poca o ninguna experiencia en programación. Si eres un estudiante curioso, un profesional que busca cambiar de carrera o simplemente alguien que quiere aprender TypeScript, este curso es para ti.

1.2 ¿Para quién es este curso?

Este curso es para cualquier persona interesada en aprender TypeScript, incluidos:

- Estudiantes que deseen aprender un nuevo lenguaje de programación.
- Profesionales que buscan mejorar sus habilidades de desarrollo web.
- Desarrolladores que deseen aprender TypeScript para crear aplicaciones web modernas y escalables.
- Cualquiera que quiera aprender un lenguaje de programación de código abierto y de alto rendimiento.
- Cualquiera que quiera aprender TypeScript para mejorar su carrera profesional.

- Cualquiera que quiera aprender TypeScript para crear aplicaciones web modernas y escalables.

1.3 ¿Cómo contribuir?

Valoramos su contribución a este curso. Si encuentra algún error, desea sugerir mejoras o agregar contenido adicional, me encantaría saber de usted.

Puede contribuir a través del repositorio en línea, donde puede compartir sus comentarios y sugerencias.

Juntos, podemos mejorar continuamente este recurso educativo para beneficiar a la comunidad de estudiantes y entusiastas de la programación.

Este ebook ha sido creado con el objetivo de proporcionar acceso gratuito y universal al conocimiento.

Estará disponible en línea para cualquier persona, sin importar su ubicación o circunstancias, para acceder y aprender a su propio ritmo.

Puede descargarlo en formato PDF, Epub o verlo en línea en cualquier momento y lugar.

¡Gracias por su interés en este curso y espero que disfrute aprendiendo TypeScript!

Part I

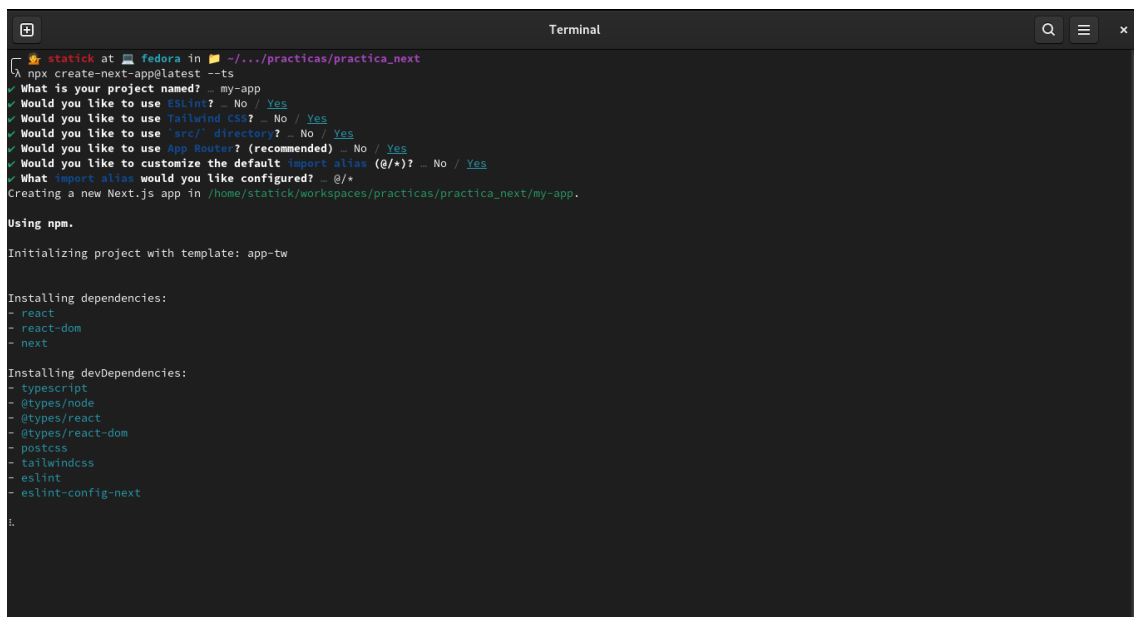
Unidad 1: Introducción a Nextjs

2 Introducción

En este ebook se presentan los conceptos básicos acerca del framework Next JS, el cual es un framework de React que permite la creación de aplicaciones web de forma sencilla y rápida. A lo largo de este ebook se presentarán los conceptos básicos de Next JS, así como ejemplos de su uso.

Lo primero que se debe hacer es instalar Next JS en su computadora. Para ello, se debe tener instalado Node JS en su computadora. Si no lo tiene instalado, puede descargarlo desde la página oficial de Node JS. Una vez que tenga Node JS instalado, puede instalar Next JS utilizando el siguiente comando:

```
npx create-next-app@latest --ts
```



```
statick at fedora in ~/.../practicas/practica_next
$ npx create-next-app@latest --ts
✔ What is your project named?  my-app
✔ Would you like to use eslint?  No / Yes
✔ Would you like to use Tailwind CSS?  No / Yes
✔ Would you like to use 'src/' directory?  No / Yes
✔ Would you like to use App Router? (recommended)  No / Yes
✔ Would you like to customize the default import alias (@/*)?  No / Yes
✔ What import alias would you like configured?  @/*
Creating a new Next.js app in /home/statick/workspaces/practicas/practica_next/my-app.

Using npm.

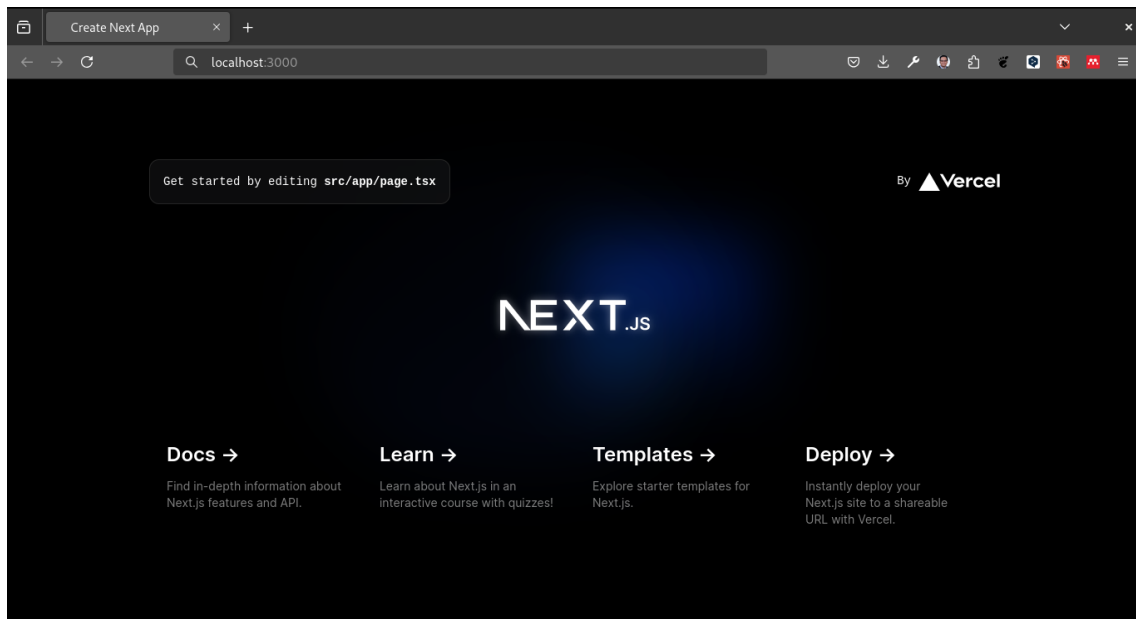
Initializing project with template: app-tw

Installing dependencies:
- react
- react-dom
- next

Installing devDependencies:
- typescript
- @types/node
- @types/react
- @types/react-dom
- postcss
- tailwindcss
- eslint
- eslint-config-next
b.
```

Este comando creará una nueva aplicación de Next JS en su computadora. Una vez que la aplicación se haya creado, puede ejecutarla utilizando el siguiente comando:

```
npm run dev
```



Este comando iniciará un servidor local en su computadora y podrá ver la aplicación en su navegador web. A partir de aquí, puede comenzar a desarrollar su aplicación utilizando Next JS.

En los siguientes capítulos se presentarán los conceptos básicos de Next JS, así como ejemplos de su uso. Espero que este ebook le sea de utilidad y le ayude a comprender mejor el framework Next JS.

3 ¿Qué es Next.js y por qué aprenderlo si quieres ser frontend senior?

Next.js es un framework de React que permite la creación de aplicaciones web de forma sencilla y rápida. A lo largo de este ebook se presentarán los conceptos básicos de Next.js, así como ejemplos de su uso.

3.1 Introducción a Next.js y sus beneficios.

Para introducirnos en Next.js, primero debemos entender qué es un framework. Un framework es un conjunto de herramientas y librerías que facilitan el desarrollo de aplicaciones web. En el caso de Next.js, es un framework de React que nos permite crear aplicaciones web de forma sencilla y rápida.

Next.js nos ofrece una serie de beneficios que lo hacen una excelente opción para el desarrollo de aplicaciones web:

- **Rendimiento:** Next.js nos ofrece un rendimiento óptimo gracias a su capacidad de renderizado en el servidor y en el cliente. Esto nos permite crear aplicaciones web rápidas y eficientes.
- **SEO:** Next.js nos ofrece una serie de herramientas que nos permiten optimizar nuestras aplicaciones web para los motores de búsqueda. Esto nos ayuda a mejorar el posicionamiento de nuestras aplicaciones en los resultados de búsqueda.
- **Escalabilidad:** Next.js nos ofrece una arquitectura escalable que nos permite crear aplicaciones web de cualquier tamaño. Esto nos permite crear aplicaciones web que puedan crecer con el tiempo y adaptarse a las necesidades de nuestros usuarios.
- **Facilidad de uso:** Next.js nos ofrece una serie de herramientas y librerías que nos facilitan el desarrollo de aplicaciones web. Esto nos permite crear aplicaciones web de forma sencilla y rápida, sin necesidad de tener un conocimiento profundo de React.

3.2 Comparación con otros frameworks.

Podemos comparar Next.js con otros frameworks de React, como Create React App. A continuación, se presentan algunas diferencias entre Next.js y Vite:

- **Rendimiento:** Next.js ofrece un rendimiento óptimo gracias a su capacidad de renderizado en el servidor y en el cliente. Vite, por otro lado, ofrece un rendimiento óptimo gracias a su capacidad de renderizado en el cliente.

- **SEO:** Next.js nos ofrece una serie de herramientas que nos permiten optimizar nuestras aplicaciones web para los motores de búsqueda. Vite, por otro lado, nos ofrece una serie de herramientas que nos permiten optimizar nuestras aplicaciones web para los motores de búsqueda.
- **Escalabilidad:** Next.js nos ofrece una arquitectura escalable que nos permite crear aplicaciones web de cualquier tamaño. Vite, por otro lado, nos ofrece una arquitectura escalable que nos permite crear aplicaciones web de cualquier tamaño.
- **Facilidad de uso:** Next.js nos ofrece una serie de herramientas y librerías que nos facilitan el desarrollo de aplicaciones web. Vite, por otro lado, nos ofrece una serie de herramientas y librerías que nos facilitan el desarrollo de aplicaciones web.

3.3 Casos de uso y ejemplos de proyectos exitosos.

Next.js es utilizado por una gran cantidad de empresas y desarrolladores en todo el mundo. Algunos ejemplos de proyectos exitosos que utilizan Next.js son:

- **Vercel:** Vercel es una plataforma de desarrollo y alojamiento de aplicaciones web que utiliza Next.js como su framework principal. Vercel nos ofrece una serie de herramientas y servicios que nos permiten crear aplicaciones web de forma sencilla y rápida.
- **Spotify:** Spotify es una plataforma de streaming de música que utiliza Next.js para su aplicación web. Spotify nos ofrece una experiencia de usuario rápida y eficiente gracias a Next.js.
- **Netflix:** Netflix es una plataforma de streaming de películas y series que utiliza Next.js para su aplicación web. Netflix nos ofrece una experiencia de usuario rápida y eficiente gracias a Next.js.

3.4 Conclusión.

Next.js es un framework de React que nos permite crear aplicaciones web de forma sencilla y rápida. A lo largo de este ebook se presentarán los conceptos básicos de Next.js, así como ejemplos de su uso. Espero que este ebook le sea de utilidad y le ayude a comprender mejor el framework Next.js.

4 Arquitectura de un proyecto de Next.js

En este capítulo se presentará la arquitectura de un proyecto de Next.js, así como las carpetas y archivos que conforman un proyecto

4.1 Estructura de carpetas y archivos.

La estructura de un proyecto de Next.js se compone de las siguientes carpetas y archivos:

- **pages:** En esta carpeta se encuentran las páginas de la aplicación. Cada archivo en esta carpeta representa una página de la aplicación. Por ejemplo, si se crea un archivo `index.js` en esta carpeta, se creará una página de inicio en la aplicación.
- **public:** En esta carpeta se encuentran los archivos estáticos de la aplicación, como imágenes, estilos y scripts. Estos archivos se pueden acceder directamente desde la URL de la aplicación.
- **styles:** En esta carpeta se encuentran los estilos globales de la aplicación. Estos estilos se aplican a toda la aplicación y se pueden importar en cualquier archivo de la aplicación.
- **components:** En esta carpeta se encuentran los componentes de la aplicación. Estos componentes se pueden reutilizar en diferentes partes de la aplicación.
- **lib:** En esta carpeta se encuentran las librerías y utilidades de la aplicación. Estas librerías se pueden importar en cualquier archivo de la aplicación.
- **api:** En esta carpeta se encuentran los endpoints de la API de la aplicación. Estos endpoints se pueden acceder desde la URL de la aplicación.
- **config:** En esta carpeta se encuentran los archivos de configuración de la aplicación. Estos archivos se pueden utilizar para configurar diferentes aspectos de la aplicación.
- **test:** En esta carpeta se encuentran los archivos de pruebas de la aplicación. Estos archivos se pueden utilizar para probar diferentes aspectos de la aplicación.
- **.env:** En este archivo se encuentran las variables de entorno de la aplicación. Estas variables se pueden utilizar para configurar diferentes aspectos de la aplicación.

4.2 Configuración inicial de un proyecto.

Para realizar la configuración inicial de un proyecto de Next.js, se deben seguir los siguientes pasos:

1. Crear un nuevo proyecto de Next.js utilizando el siguiente comando:

```
npx create-next-app@latest --ts
```

2. Instalar las dependencias del proyecto utilizando el siguiente comando:

```
npm install
```

3. Iniciar el servidor local utilizando el siguiente comando:

```
npm run dev
```

4. Acceder a la aplicación en el navegador web utilizando la URL `http://localhost:3000`.

Con estos pasos, se habrá creado un nuevo proyecto de Next.js y se podrá comenzar a desarrollar la aplicación.

4.3 Revisión de un proyecto ejemplo.

A continuación se presenta un ejemplo de la estructura de un proyecto de Next.js:

```
my-next-app/  
  pages/  
    index.tsx  
    about.tsx  
    contact.tsx  
  public/  
    images/  
      logo.png  
    styles/  
      main.css  
  styles/  
    global.css  
  components/  
    header.tsx  
    footer.tsx  
    button.tsx  
  lib/  
    api.ts  
    utils.ts  
  api/  
    users.ts
```

```
    posts.ts
  config/
    app.config.ts
  test/
    app.test.ts
  .env
```

En este ejemplo, se puede observar la estructura de un proyecto de Next.js, así como las carpetas y archivos que conforman el proyecto

5 Herramientas y stack utilizado en el curso

En este curso utilizaremos las siguientes herramientas y tecnologías:

5.1 Herramientas necesarias para el curso.

Para poder seguir este curso, necesitarás tener instaladas las siguientes herramientas en tu computadora:

- **Node.js:** Es un entorno de ejecución para JavaScript que nos permite ejecutar código JavaScript en el servidor. Puedes descargar Node.js desde la página oficial de Node.js.
- **npm:** Es el gestor de paquetes de Node.js que nos permite instalar y gestionar las dependencias de nuestros proyectos. npm viene incluido con Node.js, por lo que no es necesario instalarlo por separado.
- **Visual Studio Code:** Es un editor de código fuente desarrollado por Microsoft que nos permite escribir y editar código de forma sencilla. Puedes descargar Visual Studio Code desde la página oficial de Visual Studio Code.
- **Git:** Es un sistema de control de versiones que nos permite gestionar el código fuente de nuestros proyectos. Puedes descargar Git desde la página oficial de Git.

5.2 Configuración del entorno de desarrollo.

Creemos un nuevo proyecto de Next.js utilizando el siguiente comando:

```
npx create-next-app@latest --ts
```

5.3 Instalación y configuración de dependencias.

Instalamos las dependencias del proyecto utilizando el siguiente comando:

```
npm install
```

5.4 Iniciar el servidor local.

Iniciamos el servidor local utilizando el siguiente comando:

```
npm run dev
```

5.5 Acceder a la aplicación en el navegador.

Accedemos a la aplicación en el navegador web utilizando la URL `http://localhost:3000`.

6 Cómo crear rutas en Next.js

En este capítulo se presentarán los conceptos básicos de cómo crear rutas en Next.js. A lo largo de este capítulo se presentarán los siguientes temas:

6.1 Rutas básicas.

Las rutas en Next.js se crean utilizando la carpeta **pages**. Cada archivo en esta carpeta representa una ruta en la aplicación. Por ejemplo, si se crea un archivo **index.js** en esta carpeta, se creará una ruta de inicio en la aplicación.

6.2 Rutas dinámicas.

Las rutas dinámicas en Next.js se crean utilizando corchetes `[]` en el nombre del archivo. Por ejemplo, si se crea un archivo `[id].js` en esta carpeta, se creará una ruta dinámica en la aplicación.

6.3 Nested routes.

Las rutas anidadas en Next.js se crean utilizando la carpeta **pages** y subcarpetas. Por ejemplo, si se crea una carpeta **blog** y un archivo `[slug].js` en esta carpeta, se creará una ruta anidada en la aplicación.

6.4 Rutas con parámetros.

Las rutas con parámetros en Next.js se crean utilizando corchetes `[]` en la ruta. Por ejemplo, si se crea una ruta `/blog/[slug]` en la aplicación, se creará una ruta con parámetros en la aplicación.

6.5 Rutas con query strings.

Las rutas con query strings en Next.js se crean utilizando el signo de interrogación `?` en la ruta. Por ejemplo, si se crea una ruta `/blog?slug=hello-world` en la aplicación, se creará una ruta con query strings en la aplicación.⁹

6.6 Rutas con rutas anidadas.

Las rutas con rutas anidadas en Next.js se crean utilizando la carpeta **pages** y subcarpetas. Por ejemplo, si se crea una carpeta **blog** y un archivo **[slug].js** en esta carpeta, se creará una ruta anidada en la aplicación.

6.7 Rutas con rutas anidadas y parámetros.

Las rutas con rutas anidadas y parámetros en Next.js se crean utilizando la carpeta **pages** y subcarpetas. Por ejemplo, si se crea una carpeta **blog** y un archivo **[slug].js** en esta carpeta, se creará una ruta anidada con parámetros en la aplicación.

6.8 Rutas con rutas anidadas y query strings.

Las rutas con rutas anidadas y query strings en Next.js se crean utilizando la carpeta **pages** y subcarpetas. Por ejemplo, si se crea una carpeta **blog** y un archivo **[slug].js** en esta carpeta, se creará una ruta anidada con query strings en la aplicación.

6.9 Conclusión.

En este capítulo se presentaron los conceptos básicos de cómo crear rutas en Next.js. A lo largo de este capítulo se presentaron los siguientes temas: rutas básicas, rutas dinámicas, rutas anidadas, rutas con parámetros, rutas con query strings, rutas con rutas anidadas, rutas con rutas anidadas y parámetros, y rutas con rutas anidadas y query strings. Espero que este capítulo le sea de utilidad y le ayude a comprender mejor cómo crear rutas en Next.js.

6.10 Ejercicios.

1. Crea una ruta básica en Next.js.
2. Crea una ruta dinámica en Next.js.
3. Crea una ruta anidada en Next.js.
4. Crea una ruta con parámetros en Next.js.
5. Crea una ruta con query strings en Next.js.
6. Crea una ruta con rutas anidadas en Next.js.
7. Crea una ruta con rutas anidadas y parámetros en Next.js.
8. Crea una ruta con rutas anidadas y query strings en Next.js.

Espero que estos ejercicios le sean de utilidad y le ayuden a practicar cómo crear rutas en Next.js.

7 Cómo crear Layout en Next.js

En este capítulo se presentarán los conceptos básicos de cómo crear Layouts en Next.js. A lo largo de este capítulo se presentarán los siguientes temas:

7.1 Definición de layouts.

Para definir un layout en Next.js, se debe crear un componente de React que contenga la estructura del layout. Por ejemplo, si se desea crear un layout con un encabezado y un pie de página, se puede crear un componente **Layout** que contenga estos elementos.

7.1.1 Componente Layout:

```
import Head from 'next/head'

const Layout = ({ children }) => {
  return (
    <div>
      <Head>
        <title>My Next.js App</title>
      </Head>
      <header>
        <h1>Header</h1>
      </header>
      <main>
        {children}
      </main>
      <footer>
        <p>Footer</p>
      </footer>
    </div>
  )
}

export default Layout
```

En el componente **Layout**, se define la estructura del layout con un encabezado, un pie de página y un contenedor principal para el contenido de la página. El componente recibe como prop **children** el contenido de la página que se renderizará en el contenedor principal.

7.1.2 Componente Index:

```
import Layout from '../components/Layout'

const Index = () => {
  return (
    <Layout>
      <h1>Home Page</h1>
      <p>Welcome to my Next.js App</p>
    </Layout>
  )
}

export default Index
```

En el componente **Index**, se importa el componente **Layout** y se renderiza el contenido de la página dentro del layout. De esta forma, se crea una estructura de layout que se puede reutilizar en diferentes páginas de la aplicación.

7.1.3 Componente Layout específico:

```
import Head from 'next/head'

const Layout = ({ title, children }) => {
  return (
    <div>
      <Head>
        <title>{title}</title>
      </Head>
      <header>
        <h1>Header</h1>
      </header>
      <main>
        {children}
      </main>
      <footer>
        <p>Footer</p>
      </footer>
    </div>
  )
}

export default Layout
```

En el componente **Layout**, se define un prop **title** que permite personalizar el título de la página. De esta forma, se puede crear un layout específico para cada página de la aplicación.

7.1.4 Componente About:

```
import Layout from '../components/Layout'

const About = () => {
  return (
    <Layout title="About Page">
      <h1>About Page</h1>
      <p>Learn more about my Next.js App</p>
    </Layout>
  )
}

export default About
```

En el componente **About**, se importa el componente **Layout** y se renderiza el contenido de la página dentro del layout. Se utiliza el prop **title** para personalizar el título de la página. De esta forma, se crea un layout específico para la página **About**.

7.2 Uso de layouts globales y específicos.

En Next.js, se pueden crear layouts globales que se aplican a todas las páginas de la aplicación, así como layouts específicos que se aplican a páginas específicas de la aplicación. De esta forma, se puede personalizar el diseño de cada página de la aplicación de forma independiente.

7.2.1 Layout global:

Para crear un layout global en Next.js, se puede definir un componente **Layout** en un archivo separado y importarlo en todas las páginas de la aplicación. De esta forma, se aplica el mismo diseño a todas las páginas de la aplicación.

Ejemplo:

```
import Layout from '../components/Layout'

const MyApp = ({ Component, pageProps }) => {
  return (
    <Layout>
      <Component {...pageProps} />
    </Layout>
  )
}
```

```

    </Layout>
  )
}

export default MyApp

```

7.2.2 Layout específico:

Para crear un layout específico en Next.js, se puede definir un componente **Layout** en un archivo separado y personalizarlo con props específicas para cada página. De esta forma, se puede crear un diseño único para cada página de la aplicación.

Ejemplo:

```

import Layout from '../components/Layout'

const About = () => {
  return (
    <Layout title="About Page">
      <h1>About Page</h1>
      <p>Learn more about my Next.js App</p>
    </Layout>
  )
}

export default About

```

En este ejemplo, se crea un layout específico para la página **About** con un título personalizado. De esta forma, se puede personalizar el diseño de cada página de la aplicación de forma independiente.

7.3 Conclusión.

En este capítulo se presentaron los conceptos básicos de cómo crear Layouts en Next.js. A lo largo de este capítulo se presentaron los siguientes temas: definición de layouts, uso de layouts globales y específicos. Espero que este capítulo le sea de utilidad y le ayude a comprender mejor cómo crear Layouts en Next.js.

7.4 Ejercicios.

1. Crea un layout con un encabezado y un pie de página en Next.js.
2. Crea un layout con un título personalizado en Next.js.
3. Crea un layout global que se aplique a todas las páginas de la aplicación.

4. Crea un layout específico que se aplique a una página específica de la aplicación.
5. Crea un layout con un menú de navegación en Next.js.
6. Crea un layout con un formulario de contacto en Next.js.

Espero que estos ejercicios le sean de utilidad y le ayuden a practicar cómo crear Layouts en Next.js.

7.5 Implementación de layouts en un proyecto.

8 Cómo funciona la navegación en Next.js

La navegación en Next.js se realiza a través de la librería `next/router`. Esta librería nos permite navegar entre las diferentes páginas de nuestra aplicación de forma programática o mediante enlaces.

8.1 Link component.

El componente **Link** de Next.js nos permite crear enlaces entre las diferentes páginas de nuestra aplicación. Este componente se utiliza de la siguiente forma:

```
import Link from 'next/link'

const Index = () => {
  return (
    <div>
      <h1>Home Page</h1>
      <Link href="/about">
        <a>About Page</a>
      </Link>
    </div>
  )
}

export default Index
```

En el ejemplo anterior, se crea un enlace a la página **About** utilizando el componente **Link**. Al hacer clic en el enlace, se navegará a la página **About** de forma rápida y sin recargar la página.

8.2 Uso de Router.

La librería `next/router` nos permite navegar entre las diferentes páginas de nuestra aplicación de forma programática. Para utilizar esta librería, se debe importar el objeto **Router** de la siguiente forma:

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

const About = () => {
  const router = useRouter()

  const handleClick = () => {
    router.push('/')
  }

  return (
    <div>
      <h1>About Page</h1>
      <button onClick={handleClick}>Go to Home Page</button>
    </div>
  )
}

export default About
```

En el ejemplo anterior, se importa el objeto **Router** de la librería **next/router** y se utiliza el método **push** para navegar a la página de inicio al hacer clic en un botón.

8.3 Navegación programática.

La navegación programática nos permite navegar entre las diferentes páginas de nuestra aplicación de forma dinámica. Esto nos permite crear experiencias de usuario más interactivas y personalizadas.

8.4 Conclusión.

La navegación en Next.js se realiza a través de la librería **next/router** y el componente **Link**. Estas herramientas nos permiten crear enlaces entre las diferentes páginas de nuestra aplicación y navegar de forma programática. Conocer cómo funciona la navegación en Next.js es fundamental para crear aplicaciones web modernas y eficientes.

8.5 Ejercicio.

Crea un enlace en la página de inicio que te lleve a la página **About** y un botón en la página **About** que te lleve a la página de inicio. Utiliza tanto el componente **Link** como la librería **next/router** para realizar la navegación entre las páginas de la aplicación.

9 Manejo de parámetros en rutas en Next.js

En este capítulo se presentará cómo manejar parámetros en las rutas de Next.js. A lo largo de este capítulo se presentarán los siguientes temas:

9.1 Parámetros de ruta.

Los parámetros de ruta en Next.js se utilizan para pasar información a una ruta a través de la URL. Por ejemplo, si se desea mostrar el detalle de un producto en una tienda en línea, se puede utilizar un parámetro de ruta para pasar el ID del producto a la ruta.

Ejemplo:

```
// pages/product/[id].js
import { useRouter } from 'next/router'

const Product = () => {
  const router = useRouter()
  const { id } = router.query

  return (
    <div>
      <h1>Product Detail</h1>
      <p>Product ID: {id}</p>
    </div>
  )
}

export default Product
```

En el ejemplo anterior, se crea una ruta dinámica `/product/[id]` que recibe un parámetro `id` a través de la URL. Al acceder a la ruta `/product/123`, se mostrará el detalle del producto con el ID `123`.

9.2 Parámetros de consulta (query params).

Los parámetros de consulta en Next.js se utilizan para pasar información a una ruta a través de la URL utilizando el signo de interrogación `?`. Por ejemplo, si se desea filtrar una lista de productos por categoría, se puede utilizar un parámetro de consulta para pasar la categoría a la ruta.

Ejemplo:

```
// pages/products.js
import { useRouter } from 'next/router'

const Products = () => {
  const router = useRouter()
  const { category } = router.query

  return (
    <div>
      <h1>Products</h1>
      <p>Category: {category}</p>
    </div>
  )
}

export default Products
```

En el ejemplo anterior, se crea una ruta **/products** que recibe un parámetro de consulta **category** a través de la URL. Al acceder a la ruta **/products?category=electronics**, se mostrarán los productos de la categoría **electronics**.

9.3 Uso de useRouter para acceder a los parámetros.

La librería **next/router** proporciona el hook **useRouter** para acceder a los parámetros de ruta y de consulta en una página de Next.js. Este hook nos permite obtener los parámetros de la URL y utilizarlos en la página.

Ejemplo:

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

const Product = () => {
  const router = useRouter()
  const { id } = router.query

  return (
    <div>
      <h1>Product Detail</h1>
      <p>Product ID: {id}</p>
    </div>
  )
}

export default Product
```

En el ejemplo anterior, se importa el hook **useRouter** de la librería **next/router** y se utiliza para acceder al parámetro **id** de la URL. Este parámetro se puede utilizar en la página para mostrar el detalle del producto.

9.4 Conclusión.

En este capítulo se presentó cómo manejar parámetros en las rutas de Next.js. A lo largo de este capítulo se presentaron los siguientes temas: parámetros de ruta, parámetros de consulta (query params) y uso de **useRouter** para acceder a los parámetros. Espero que este capítulo le sea de utilidad y le ayude a comprender mejor cómo manejar parámetros en las rutas de Next.js.

9.5 Ejercicios.

1. Crea una ruta dinámica en Next.js que reciba un parámetro de ruta.
2. Crea una ruta en Next.js que reciba un parámetro de consulta y muestre el valor en la página.
3. Utiliza el hook **useRouter** para acceder a los parámetros de ruta y de consulta en una página de Next.js.
4. Crea una aplicación en Next.js que utilice parámetros de ruta y de consulta en diferentes páginas.
5. Experimenta con diferentes formas de pasar parámetros a las rutas en Next.js y observa cómo se comporta la aplicación.
6. Investiga cómo validar los parámetros de ruta y de consulta en Next.js y aplica la validación en tu aplicación.
7. Comparte tus experiencias y aprendizajes sobre cómo manejar parámetros en las rutas de Next.js con tus compañeros de clase.
8. Investiga cómo manejar parámetros en las rutas de Next.js utilizando la librería **next/router** y comparte tus hallazgos con tus compañeros de clase.

10 React Server Components en Next.js: notación “use Client”

En este capítulo se presentarán los conceptos básicos de cómo utilizar React Server Components en Next.js. A lo largo de este capítulo se presentarán los siguientes temas:

10.1 Introducción a React Server Components.

React Server Components es una nueva característica de React que permite renderizar componentes en el servidor y enviar solo los datos necesarios al cliente. Esto mejora el rendimiento de la aplicación al reducir la cantidad de datos que se envían al cliente.

10.1.1 ¿Qué es use Client?

La notación “use Client” en React Server Components se utiliza para indicar que un componente se renderizará en el cliente en lugar de en el servidor. Esto permite que el componente se actualice de forma dinámica en el cliente sin tener que volver a renderizarlo en el servidor.

10.2 Uso de la notación “use Client”.

La notación “use Client” se utiliza en un componente de React Server Components para indicar que el componente se renderizará en el cliente. Por ejemplo:

```
import { useClient } from 'react-server-components'

const MyComponent = () => {
  const data = useClient(fetchData)

  return (
    <div>
      <h1>{data.title}</h1>
      <p>{data.content}</p>
    </div>
  )
}

export default MyComponent
```

En el ejemplo anterior, el componente **MyComponent** utiliza la notación “use Client” para indicar que se renderizará en el cliente. El componente llama a la función **fetchData** para obtener los datos necesarios y los muestra en la interfaz de usuario.

10.3 Ventajas de la notación “use Client”.

La notación “use Client” en React Server Components ofrece las siguientes ventajas:

- Permite renderizar componentes en el cliente de forma dinámica.
- Reduce la cantidad de datos que se envían al cliente.
- Mejora el rendimiento de la aplicación al evitar renderizaciones innecesarias en el servidor.

10.4 Ejemplos prácticos.

1. Crea un componente de React Server Components que utilice la notación “use Client”.

```
import { useClient } from 'react-server-components'

const MyComponent = () => {
  const data = useClient(fetchData)

  return (
    <div>
      <h1>{data.title}</h1>
      <p>{data.content}</p>
    </div>
  )
}

export default MyComponent
```

2. Utiliza la notación “use Client” en un componente de React Server Components para renderizar datos dinámicamente en el cliente.

```
import { useClient } from 'react-server-components'

const MyComponent = () => {
  const data = useClient(fetchData)

  return (
    <div>
      <h1>{data.title}</h1>
      <p>{data.content}</p>
    </div>
  )
}
```

```
)  
}  
  
export default MyComponent
```

10.5 Conclusión.

En este capítulo se presentaron los conceptos básicos de cómo utilizar React Server Components en Next.js. A lo largo de este capítulo se presentaron los siguientes temas: introducción a React Server Components, notación “use Client”, uso de la notación “use Client” y ventajas de la notación “use Client”. Espero que este capítulo le sea de utilidad y le ayude a comprender mejor cómo utilizar React Server Components en Next.js.

10.6 Ejercicios.

1. Crea un componente de React Server Components que utilice la notación “use Client”.
2. Utiliza la notación “use Client” en un componente de React Server Components para renderizar datos dinámicamente en el cliente.
3. Crea una aplicación en Next.js que utilice React Server Components y la notación “use Client” en diferentes componentes.

Espero que estos ejercicios le sean de utilidad y le ayuden a practicar cómo utilizar React Server Components en Next.js.

11 Manejo de estilos y archivos estáticos en Next.js

En este capítulo se presentarán los conceptos básicos de cómo manejar estilos y archivos estáticos en Next.js. A lo largo de este capítulo se presentarán los siguientes temas:

11.1 Estilos en Next.js

Los estilos en Next.js se pueden manejar de diferentes formas, como CSS global, CSS Modules y Styled JSX. Cada una de estas formas tiene sus propias ventajas y desventajas, y se pueden utilizar según las necesidades del proyecto.

11.1.1 CSS global

El CSS global en Next.js se puede utilizar para aplicar estilos a toda la aplicación. Para utilizar CSS global en Next.js, se puede crear un archivo **styles.css** en la carpeta **public** y enlazarlo en el archivo ****_app.js****.

```
/* styles.css */

body {
  font-family: 'Arial', sans-serif;
  background-color: #f0f0f0;
}
```

```
// _app.js
import '../public/styles.css'

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

export default MyApp
```

11.2 CSS Modules en Next.js

11.2.1 CSS Modules

Los CSS Modules en Next.js permiten crear estilos locales para cada componente. Para utilizar CSS Modules en Next.js, se puede crear un archivo **styles.module.css** en la carpeta del componente y importarlo en el archivo del componente.

```
/* Button.module.css */
```

```
.button {  
  background-color: #007bff;  
  color: #fff;  
  padding: 10px 20px;  
  border: none;  
  border-radius: 5px;  
}
```

```
// Button.js
```

```
import styles from './Button.module.css'
```

```
const Button = () => {  
  return <button className={styles.button}>Click me</button>  
}
```

```
export default Button
```

11.2.2 Scoped CSS en Next.js

En Next.js, los estilos se aplican de forma local por defecto, lo que significa que los estilos de un componente no afectan a otros componentes. Esto se conoce como scoped CSS y ayuda a evitar conflictos de estilos entre componentes.

11.2.3 Creación y uso de CSS Modules.

Para crear y utilizar CSS Modules en Next.js, se deben seguir los siguientes pasos:

1. Crear un archivo **styles.module.css** en la carpeta del componente.
2. Definir los estilos en el archivo **styles.module.css** utilizando la sintaxis de CSS.
3. Importar los estilos en el archivo del componente y utilizar la clase generada por CSS Modules.


```
/* styles.module.css */
```

```
.button {  
  background-color: #007bff;  
  color: #fff;  
  padding: 10px 20px;  
  border: none;  
  border-radius: 5px;  
}
```

```
// Button.js
```

```
import styles from './styles.module.css'
```

```
const Button = () => {  
  return <button className={styles.button}>Click me</button>  
}
```

```
export default Button
```

11.3 Styled JSX en Next.js

11.3.1 Styled JSX

Styled JSX en Next.js permite escribir estilos en línea dentro de los componentes. Para utilizar Styled JSX en Next.js, se puede utilizar la etiqueta **style** y definir los estilos dentro de ella.

```
// Button.js
```

```
const Button = () => {  
  return (  
    <button>  
      Click me  
      <style jsx>{`  
        button {  
          background-color: #007bff;  
          color: #fff;  
          padding: 10px 20px;  
          border: none;  
          border-radius: 5px;  
        }  
      `}</style>  
    </button>  
  )  
}
```

```
export default Button
```

11.3.2 Ventajas de Styled JSX

Las ventajas de Styled JSX en Next.js son las siguientes:

- Permite escribir estilos en línea de forma sencilla.
- Permite utilizar variables y funciones de JavaScript en los estilos.
- Permite aplicar estilos de forma local a un componente.

11.4 Conclusión

En este capítulo se presentaron los conceptos básicos de cómo manejar estilos y archivos estáticos en Next.js. A lo largo de este capítulo se presentaron las diferentes formas de manejar estilos en Next.js, como CSS global, CSS Modules y Styled JSX. Espero que este capítulo le sea de utilidad y le ayude a comprender mejor cómo manejar estilos y archivos estáticos en Next.js.

11.5 Ejercicios

1. Crea un archivo **styles.css** en la carpeta **public** y enlázalo en el archivo ****_app.js**** para aplicar estilos globales a la aplicación.
2. Crea un archivo **styles.module.css** en la carpeta de un componente y utilízalo para aplicar estilos locales al componente.
3. Utiliza Styled JSX en un componente de Next.js para aplicar estilos en línea al componente.

Espero que estos ejercicios le sean de utilidad y le ayuden a practicar cómo manejar estilos y archivos estáticos en Next.js.