ATELIER 2 : ROUTAGE NEXT JS

Introduction

Next.js est un Framework React livré avec toutes les fonctionnalités dont vous avez besoin pour la production. Next.js active le routage dans votre application à l'aide du routage basé sur le système de fichiers.

Next.js utilise le système de fichiers pour activer le routage dans l'application. Next traite automatiquement tous les fichiers du répertoire app avec les extensions .js, .jsx, .ts ou .tsx comme une route. Une page dans Next.js est un composant React qui a une route basée sur son nom de fichier.

Par défaut, les composants dans le répertoire **/app** sont des composants serveur React. Il s'agit d'une optimisation des performances et permet de les adopter facilement. Cependant, vous pouvez également utiliser les composants client.

Les composants du serveur React sont récupérés et rendus sur le serveur, tandis que les composants du client React sont récupérés et rendus sur le client. Si votre composant inclut l'interactivité, comme l'un des hooks du cycle de vie, faites-en un composant côté client, alors que si ce n'est pas le cas, faites-en un composant serveur.

Un composant devient un composant client lors de l'utilisation de la directive **"use client"** en haut du fichier (avant toute importation).

Ces composants (et fichiers) peuvent être placés n'importe où dans votre application. Ils n'ont pas besoin d'être dans app/, en particulier.

Quand utiliser les composants serveur ou client ?

Pour simplifier la décision entre les composants serveur et client, nous vous recommandons d'utiliser les composants serveur (par défaut dans le répertoire de l'application) jusqu'à ce que vous ayez besoin d'utiliser un composant client.

Ce tableau résume les différents cas d'utilisation des composants serveur et client :

What do you need to do?	Server Component	Client Component
Fetch data. Learn more.	✅	⚠️
Access backend resources (directly)	✅	❌
Keep sensitive information on the server (access tokens, API keys, etc)	✅	❌
Keep large dependencies on the server / Reduce client-side JavaScript	✅	❌
Add interactivity and event listeners (`onClick()`, `onChange()`, etc)	❌	✅
Use State and Lifecycle Effects (`useState()`, `useReducer()`, `useEffect()`, etc)	❌	✅
Use browser-only APIs	❌	✅
Use custom hooks that depend on state, effects, or browser-only APIs	❌	✅
Use React Class components	❌	✅

Par exemple, vous voudrez peut-être un répertoire components/ pour les composants de type client. Cependant, le répertoire app/ vous permet de Co localiser des composants avec des pages. C'est que dans le répertoire **/app** les dossiers sont utilisés pour définir des **routes**. Une route est un chemin unique de dossiers imbriqués, suivant la hiérarchie du dossier racine jusqu'au dossier feuille final.

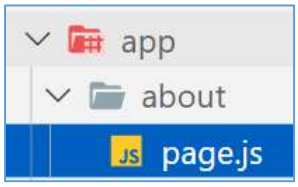
Exemple :

Les fichiers sont utilisés pour créer une interface utilisateur qui s'affiche pour le segment de route.

Chaque dossier d'une route représente un segment de route. Chaque segment de route est mappé à un segment correspondant dans un chemin d'URL.

Créer les pages

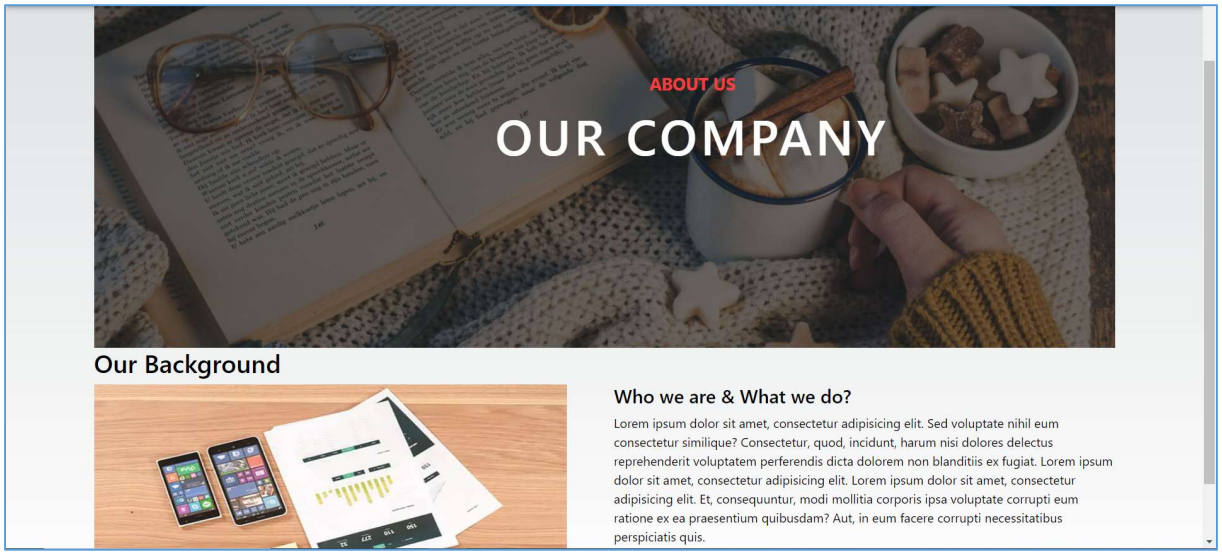
Créer le dossier about sous /app puis y créer le ficher page.js




```
import Image from 'next/image'

export default function About() {
  return (
<div className="container">
<div className="banner-item-02">
          <div className="text-content">
            <h4>About us</h4>
            <h2>Our company</h2>
          </div>
      </div>
        <div className="row">
          <div className="col-md-12">
            <div>
              <h2>Our Background</h2>
            </div>
          </div>
          <div className="col-md-6">
            <div>
              <Image src="/images/feature-image.jpg" alt="" width="600"
height="300" priority />
            </div>
          </div>
          <div className="col-md-6">
            <div>
              <h4>Who we are &amp;amp; What we do?</h4>
              <div>
                Lorem ipsum dolor sit amet, consectetur adipisicing elit.
                Sed voluptate nihil eum consectetur similique?
                Consectetur, quod, incidunt, harum nisi dolores delectus
reprehenderit voluptatem perferendis dicta dolorem non blanditiis ex fugiat.
Lorem ipsum dolor sit amet, consectetur adipisicing elit.
                Lorem ipsum dolor sit amet, consectetur adipisicing elit. Et,
consequuntur, modi mollitia corporis ipsa voluptate corrupti eum ratione ex ea
```

```
praesentium quibusdam? Aut, in eum facere corrupti necessitatibus perspiciatis
quis.
            </div>
          </div>
        </div>
      </div>
    </div>
    )
    }
```

http://localhost:3000/about



Sous le dossier components créer le fichier Navbar.js

Références :

https://mui.com/material-ui/react-app-bar/

https://beta.nextjs.org/docs/routing/linking-and-navigating

Tous les composants clients sont marqués par 'use client'

Le routeur Next.js utilise un routage centré sur le serveur avec une navigation côté client. Il prend en charge les états de chargement instantanés et le rendu simultané. Cela signifie que la navigation maintient l'état côté client, évite les re-rendus coûteux, est interruptible et ne provoque pas de conditions de concurrence.

Il existe deux façons de naviguer entre les itinéraires :

- **Composant <Link>**

<Link> est un composant React qui étend l'élément HTML <a> pour fournir une prélecture et une navigation côté client entre les routes. C'est le principal moyen de naviguer entre les itinéraires dans Next.js

- **useRouter Hook**

Le hook useRouter vous permet de modifier les routes à l'intérieur des composants clients.

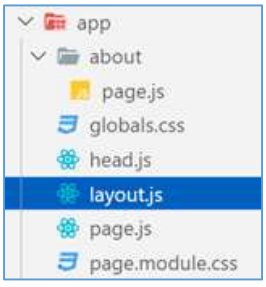
useRouter fournit des méthodes telles que push(), refresh(), etc.

Pour utiliser useRouter, importez-le depuis next/navigation et appelez le hook dans votre composant client :

```
"use client";

import { useEffect,useState } from 'react';

import { useRouter } from 'next/navigation';

import AppBar from '@mui/material/AppBar';
import Box from '@mui/material/Box';
import Toolbar from '@mui/material/Toolbar';
import Typography from '@mui/material/Typography';
import Button from '@mui/material/Button';
import AllOutIcon from '@mui/icons-material/AllOut';
import HomeIcon from '@mui/icons-material/Home';

function Navbar () {

    const router = useRouter();

  const [onTop, setOnTop] = useState(true);

  useEffect(() => {
    window.addEventListener('scroll', handleScroll);
  });

  const handleScroll = () => {
    if(window.pageYOffset === 0) {
      setOnTop(true);
    } else {
      setOnTop(false);
    }
  }

    return (
  <>
```

```
  <Box sx={{ flexGrow: 1 }}>
    <AppBar color={onTop ? 'transparent' : 'inherit'}>
          <Toolbar>

            <Typography variant="h6" component="div" sx={{ flexGrow: 1 }}
color="default">
              E-shopping
            </Typography>
 <Button color="inherit" onClick={() => router.push('/')}><HomeIcon
color="secondary"/> Home </Button>

            <Button color="inherit" onClick={() =>
router.push('/about')}><AllOutIcon color="primary"/> About </Button>

          </Toolbar>
        </AppBar>
      </Box>
     <Toolbar />
  </>

    );
}

export default Navbar ;
```

app/layout.js




```
import './globals.css'
import Navbar from "../components/Navbar"

export default function RootLayout({ children }) {
  return (
    <html lang="en">
      {/*
        <head /> will contain the components returned by the nearest parent
        head.jsx. Find out more at https://beta.nextjs.org/docs/api-
reference/file-conventions/head
      */}
```

```
        <head />

        <body>
        <Navbar />
          {children}
        </body>
      </html>
  )
}
```