ATELIER 6 : NESTED LAYOUTS NEXT JS

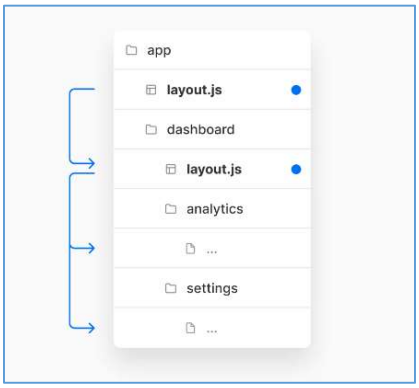
Introduction

Next introduit une option de mise en page imbriquée, ce qui signifie que nous pouvons avoir des mises en page partagées qui rendent certaines parties : nested layouts.

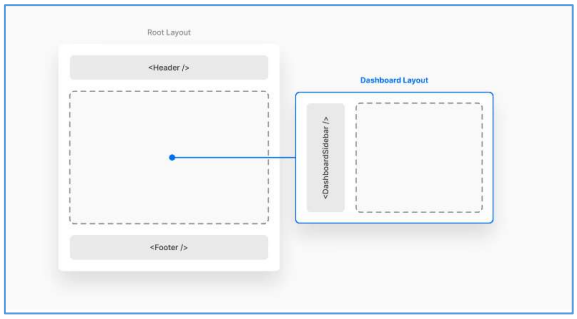
Il n'y a qu'une seule mise en page requise, qui est la mise en page racine : /app/layout.js

En plus du layout global, on peut également créer des layouts individuels. C'est ce qu'on appelle des layouts nested.

L'expérience du développeur dans la création de mises en page peut être améliorée. Il devrait être facile de créer des mises en page qui peuvent être imbriquées, partagées entre les itinéraires et dont l'état est préservé lors de la navigation.



Par exemple, si nous devions combiner les deux mises en page ci-dessus. La mise en page racine (app/layout.js) serait appliquée à la mise en page du tableau de bord, qui s'appliquerait également à tous les segments de route dans dashboard/*.



Créer le dashboard

Sous le dossier components créer le fichier style.css

style.css

```css
.stylediv
{
width: 200px;
height: 50px;
margin: 0 auto;
text-align: center;
align-items: center;
padding-top: 1.5rem;
padding-bottom: 1.5rem;
}
.exemple:hover {
background: #b3b7bc;
}
.stylepop{
position: absolute;
top:200px;
max-width: 100%;
}
.containerst{
background-color: #f2f3f4;
height:870px;
}
```

Sous le dossier components créer le fichier adminSidebar.js


```javascript
"use client" ;
import React from "react";
import { useRouter } from 'next/navigation';

import "./style.css";
import { Paper, Divider, MenuList, MenuItem, Typography } from
'@mui/material';

//https://mui.com/material-ui/material-icons/
//https://htmlcolorcodes.com/fr/

import Person3Icon from '@mui/icons-material/Person3';
import PeopleOutlineIcon from '@mui/icons-material/PeopleOutline';
import ArticleOutlinedIcon from '@mui/icons-material/ArticleOutlined';
import ReceiptLongOutlinedIcon from '@mui/icons-material/ReceiptLongOutlined';
import AssignmentOutlinedIcon from '@mui/icons-material/AssignmentOutlined';
import SettingsIcon from '@mui/icons-material/Settings';
```

```
const AdminSidebar = () => {

    const router = useRouter();

return (
    <div className="containerst">
<Paper className="stylepop" >
<MenuList >
<MenuItem >

</MenuItem>
<Divider />
<MenuItem>
<div onClick={()=>{}}
className="styledIv">
<div>
<Person3Icon sx={{ color: '#1C15F5'}}/>
</div>
<div><Typography sx={{ color: 'gray'}}>User Profile</Typography></div>
</div>
</MenuItem>
<Divider />
<MenuItem>
<div onClick={()=>{}}
className="styledIv">
<div>
<PeopleOutlineIcon sx={{ color: '#FF5733'}}/>
</div>
<div><Typography sx={{ color: 'gray'}}>Clients</Typography></div>
</div>
</MenuItem>
<Divider />
<MenuItem>
<div onClick={()=>router.push('/tableCategories')}
className="styledIv">
<div>
<ArticleOutlinedIcon sx={{ color: '#316610'}}/>
</div>
<div><Typography sx={{ color: 'gray'}}>
Categories</Typography></div>
</div>
</MenuItem>
<Divider />
<MenuItem>
<div onClick={()=>router.push('/tableProducts')}
className="styledIv">
<div>
<ReceiptLongOutlinedIcon sx={{ color: '#991793'}}/>
</div>
```

```
<div><Typography sx={{ color: 'gray'}}>Products</Typography></div>
</div>
</MenuItem>
<Divider />
<MenuItem>
<div onClick={()=>{}}
className="styledIv">
<div>
<AssignmentOutlinedIcon sx={{ color: '#FFC300'}}/>
</div>
<div><Typography sx={{ color: 'gray'}}>Orders</Typography></div>
</div>
</MenuItem>
<Divider />
<MenuItem>
<div onClick={()=>{router.push('/dashboard')}}
className="styledIv">
<div>
<SettingsIcon sx={{ color: '#831C5D'}}/>
</div>
<div><Typography sx={{ color: 'gray'}}>Settings</Typography></div>
</div>
</MenuItem>
<Divider />
</MenuList>
</Paper>
</div>
);
};
export default AdminSidebar;
```

Créer sous /app le dossier dashboard. Puis le page.js

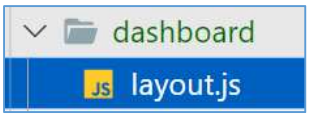

```
import React from "react";
import Image from 'next/image'
const DashboardPage= async ()=> {

    return (
      <div className="container mx-auto shadow p-4">
      <Image src="/images/charts.png" alt="" width="1050" height="300"
priority />
      </div>
```

```
    )
}

export default DashboardPage;
```

Créer sous /app/dashboard le fichier layout.js




```
'use client' ;
import AdminSidebar from "@/components/adminSidebar";

function DasboardLayout ({ children }) {
    return (

      <div className="row">

      <div className="col-md-12 col-lg-2 mb-4 mb-lg-0 pt-4 ">

      <AdminSidebar/>
      </div>
      <div className="col-md-9 col-lg-7 mb-4 mb-lg-0 pt-4 ">
      {children}
      </div>

    </div>



     );
}

export default DasboardLayout;
```

http://localhost:3000/dashboard

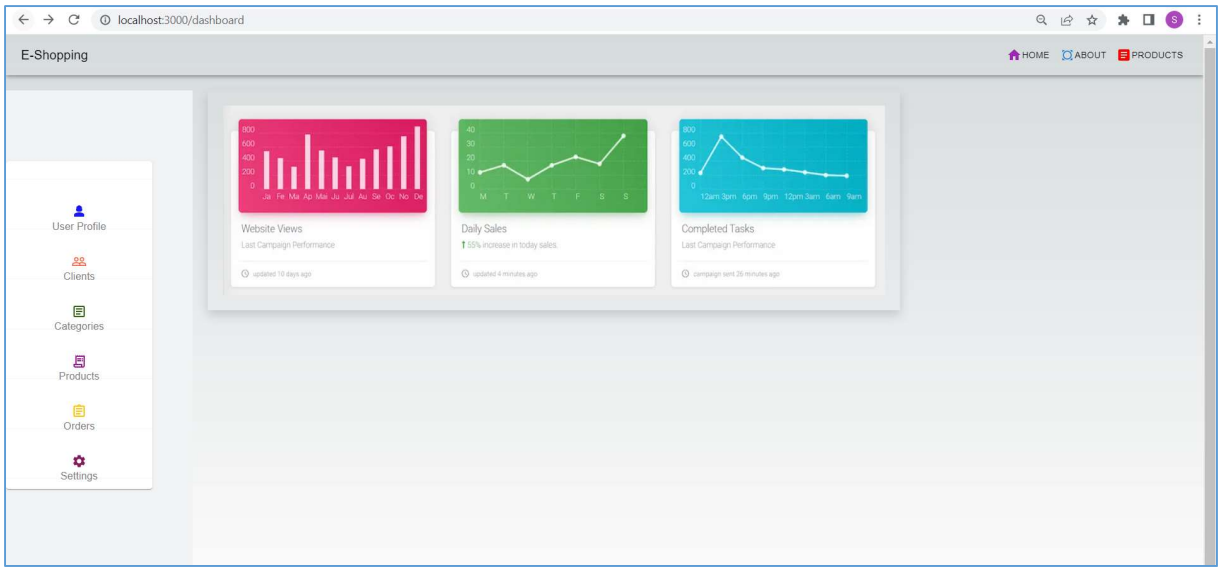


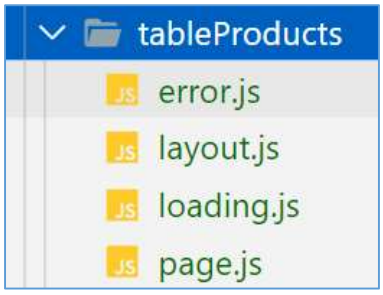
Table des articles

Installer les dépendances de mui datatables

```
npm i mui-datatables --legacy-peer-deps

npm i @mui/styles --legacy-peer-deps
```

sous /app créer le dossier tableProducts ayant ces fichiers



Dans page.js écrire ce code :

```
import React from "react";

import dynamic from 'next/dynamic'

const AffTableProducts = dynamic(() =>
import('@/components/affTableProducts'), {
    loading: () => 'Loading...',  ssr: false,
})

async function getProducts(){
    const res= await fetch('https://api.escuelajs.co/api/v1/products')
    const products = await res.json();
```

```
    return products;
}

const tableProducts= async ()=> {
    const products = await getProducts();

    return (
        <div className="container mx-auto shadow p-4">
            <AffTableProducts products={products} />
        </div>
    )
}

export default tableProducts;
```

Remarque :

Lors du rendu de l'application, il y a une différence entre l'arborescence React qui a été pré-rendu (SSR/SSG) et l'arborescence React qui a été rendue lors du premier rendu dans le navigateur. Le premier rendu s'appelle Hydratation, une fonctionnalité de React.

Cela peut entraîner la désynchronisation de l'arborescence React avec le DOM et entraîner la présence de contenus/attributs inattendus.

Ce qui génère un warning de type :

hydration-error-info.js

En général, ce problème est dû à l'utilisation d'une bibliothèque ou d'un code d'application spécifique qui s'appuie sur quelque chose qui peut différer entre le pré-rendu et le navigateur.

En effet, lors de l'utilisation de composants stylés / émotion, lorsque les bibliothèques css-in-js ne sont pas configurées pour le pré-rendu (SSR/SSG), cela entraînera souvent une inadéquation de l'hydratation. En général, cela signifie que l'application doit suivre l'exemple Next.js pour la bibliothèque.

La solution réside dans la façon de l'appel du composant, il faut faire :

```
import dynamic from 'next/dynamic'

const AffTableProducts = dynamic(() =>
import('@/components/affTableProducts'), {

    ssr: false,

})
```

A la place de

```
import AffTableProducts from '@/components/affTableProducts';
```

ssr: false ça veut dire que pour charger dynamiquement un composant côté client, on utilise l'option ssr pour désactiver le rendu du serveur. Ceci est utile si une dépendance ou un composant externe repose sur des API de navigateur telles que window.

On utilise dynamic dans Next.js pour prendre en charge le chargement paresseux des bibliothèques externes avec les composants import() et React avec next/dynamic. Le chargement différé permet d'améliorer les performances de chargement initiales en diminuant la quantité de JavaScript nécessaire pour afficher la page. Les composants ou les bibliothèques ne sont importés et inclus dans le bundle JavaScript que lorsqu'ils sont utilisés.

Références :

https://nextjs.org/docs/advanced-features/dynamic-import

https://nextjs.org/docs/messages/react-hydration-error

Sous le dossier components créer le fichier


```
"use client" ;
import React from 'react';

import MUIDataTable from "mui-datatables";

const affTableProducts=({products})=>{


 const columns = [
      {
        label: "Title",
        name: "title"
      },
      {
        label: "price",
        name: "price"
      },
      {
        label: "Description",
        name: "description"
      },
      {
        label: "Image",
        name:"images",
        options: {
          customBodyRender : (rowdata) => (
            <img
              style={{ height: 40, width : 60, borderRadius: '10%' }}
```

```
              src= {`${rowdata[0]}`}
              alt=""
            />
          )
          }
        }
          ];

return(

 <>
     {products  && products?.length > 0 ?

            <MUIDataTable
            title="Products List"
            data={products}
            columns={columns}
            />

    :null}
  </>


    )
}
export default affTableProducts;
```

Dans tableProducts/layout.js faire appel au menu side bar.


```
'use client' ;
import AdminSidebar from "@/components/adminSidebar";

function ProductLayout({ children }) {
    return (

      <div className="row">

      <div className="col-md-12 col-lg-2 mb-4 mb-lg-0 pt-4 ">

      <AdminSidebar/>
      </div>
      <div className="col-md-12 col-lg-9 mb-4 mb-lg-0 pt-4 ">
      {children}
```

```
      </div>

    </div>


     );
}

export default ProductLayout;
```