# ATELIER 3 : DATA FETCHING NEXT JS

## Introduction

React et Next.js 13 ont introduit une nouvelle façon de récupérer et de gérer les données dans votre application. Le nouveau système de récupération de données fonctionne dans le répertoire de l'application et repose sur l'API Web **fetch()**.

fetch() est une API Web utilisée pour récupérer des ressources distantes qui renvoie une promesse. React étend la récupération pour fournir une déduplication automatique des requêtes, et Next.js étend l'objet d'options de récupération pour permettre à chaque requête de définir sa propre mise en cache et revalidation.

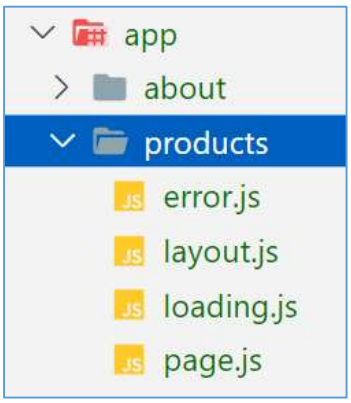
Vous pouvez utiliser **async** et **await** pour récupérer des données dans les composants serveur. async et await permettent un comportement asynchrone, basé sur une promesse ( Promise ).

Une fonction async peut contenir une expression await qui interrompt l'exécution de la fonction asynchrone et attend la résolution de la promesse passée Promise. La fonction asynchrone reprend ensuite puis renvoie la valeur de résolution.

Le mot-clé await est uniquement valide au sein des fonctions asynchrones. Si ce mot-clé est utilisé en dehors du corps d'une fonction asynchrone, cela provoquera une erreur.

## app/products

Créer le dossier products sous /app




Sans lequel on va créer :

- page.js : fichier utilisé pour définir l'interface utilisateur unique d'une route et rendre le chemin accessible au public.
- layout.js : fichier utilisé pour définir l'interface utilisateur partagée sur plusieurs pages. Une mise en page accepte une autre mise en page ou une page comme enfant. Vous pouvez imbriquer des mises en page pour créer des routes imbriquées.
- loading.js : fichier facultatif utilisé pour créer une interface utilisateur de chargement pour une partie spécifique d'une application. Il encapsule automatiquement une page ou une mise en

page enfant, affichant votre composant de chargement immédiatement lors du premier chargement et lors de la navigation entre les routes frères.

- error.js : fichier facultatif utilisé pour isoler les erreurs dans des parties spécifiques d'une application, afficher des informations d'erreur spécifiques et des fonctionnalités pour tenter de récupérer de l'erreur. Il encapsule automatiquement une page ou une mise en page enfant dans une limite d'erreur React.

On peut aussi ajouter :

- template.js : un fichier facultatif, similaire aux mises en page, mais lors de la navigation, une nouvelle instance du composant est montée et l'état n'est pas partagé. Vous pouvez utiliser des modèles pour les cas où vous avez besoin de ce comportement, comme les animations d'entrée/sortie.
- head.js : un fichier facultatif utilisé pour définir le contenu de la balise <head> pour une route donnée.
- not-found.js : fichier facultatif utilisé pour afficher l'interface utilisateur lorsque la fonction notFound est lancée dans un segment de route.


C'est parce que **par défaut tous les composants sous /app sont des server components**. Mais dans ce cas on n'utilise pas de server component pour ce fichier error.js, il s'agit d'un **client-side component**. Pour cela, il faut ajouter **"use client"**

Le fichier error.js devrait être ainsi :

```
"use client";

import { useEffect } from "react";

export default function Error({ error }) {
  useEffect(() => {
    console.log("err => ", error);
  }, [error]);

  return (
     <div className="row mt-5">
     <div className="col-9 mt-5">
       <h2 className="d-flex justify-content-center mt-5">Error! Something
went wrong.</h2>
     </div>
   </div>
  );
}
```

loading.js

```
import React from 'react'

const Loading = () => {
  return (

    <div className="row mt-5">
      <div className="col-9 mt-5">
        <h2 className="d-flex justify-content-center mt-5">Loading ... </h2>
      </div>
    </div>

  )
}

export default Loading
```

layout.js

```
'use client' ;
import styles from '../page.module.css';

function ProductLayout({ children }) {
    return (
    <div className={styles.main}>

     <div className={styles.grid}>
      {children}
     </div>
    </div>

     );
}

export default ProductLayout;
```

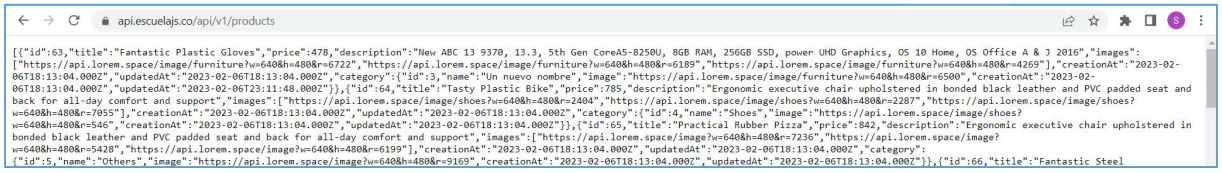
page.js

Dans Next.js 13, tous les composants sont par défaut "serveur" et peuvent être asynchrones. Cela rend la récupération de données beaucoup plus facile et plus flexible. La meilleure chose est que vous pouvez distribuer votre logique de récupération côté serveur à plusieurs endroits et les colocaliser avec des composants qui utilisent les données.

## API online :

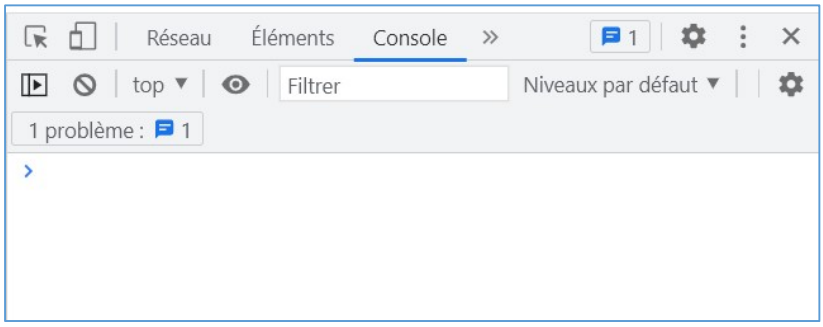
https://fakeapi.platzi.com/

url : https://fakestoreapi.com/products




```
import React from 'react';

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

const ProductsPage = async ()=> {
    const products = await getProducts();
    console.log(products);

    return (
       <>
       Products Page
       </>
     );
}

export default ProductsPage;
```

Dans la console du navigateur on ne trouve rien :

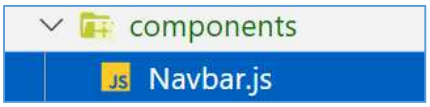



Mais par contre dans la console de l'invite du terminal on peut visualiser les résultats.

```
 {
   id: 165,
   title: 'Rustic Bronze Ball',
   price: 402,
   description: 'The Nagasaki Lander is the trademarked name of several series of Nagasaki sport bikes, that started
with the 1984 ABC800J',
   images: [
     'https://api.lorem.space/image/fashion?w=640&h=480&r=1873',
     'https://api.lorem.space/image/fashion?w=640&h=480&r=1903',
     'https://api.lorem.space/image/fashion?w=640&h=480&r=8915'
   ],
   creationAt: '2023-02-06T18:13:04.000Z',
   updatedAt: '2023-02-06T18:13:04.000Z',
   category: {
     id: 1,
     name: 'Clothes',
     image: 'https://api.lorem.space/image/fashion?w=640&h=480&r=4708',
     creationAt: '2023-02-06T18:13:04.000Z',
     updatedAt: '2023-02-06T18:13:04.000Z'
   }
 },
```

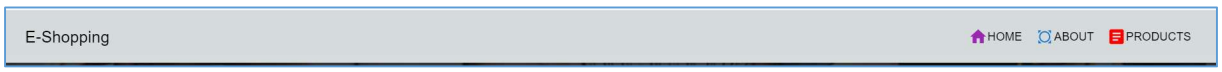
**C'est parce que la page qu'on a créée est un server component** c'est pour cela qu'on a pu visualiser les résultats dans la console du serveur.

Modifier le contenu de Navbar




```
"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';
import ArticleIcon from '@mui/icons-material/Article';

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>

            <Button color="inherit" onClick={() =>
router.push('/products')}><ArticleIcon style={{ color: 'red' }}/> Products
</Button>
          </Toolbar>
        </AppBar>
      </Box>
     <Toolbar />
  </>

    );
}

export default Navbar ;
```



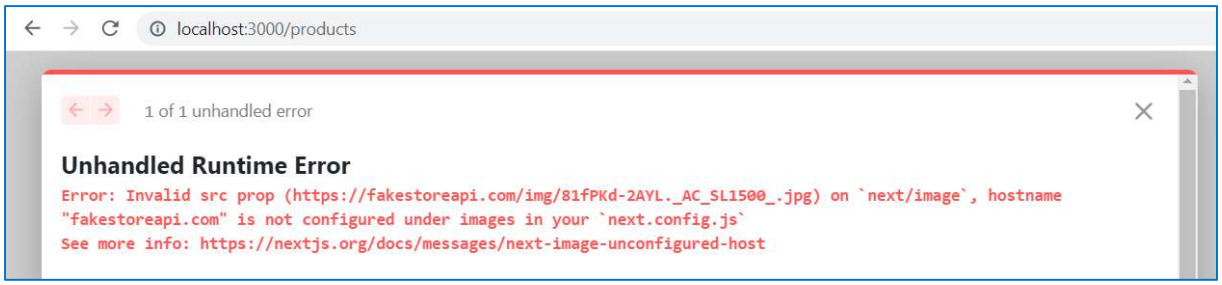

## Afficher les produits sous forme de cards

Modifier le contenu de /app/page.js


```
import React from 'react';
import Link from 'next/link';

async function getProducts(){
    const res= await fetch('https://fakestoreapi.com/products?limit=6')
    const products = await res.json();
    return products;
}

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

    return (
        <section style={{ backgroundColor: "#eee" }}>
        <div className="container py-5">
          <div className="row">
            {products?.map((product) => (
              <div
                key={product?.id}
                className="col-md-12 col-lg-3 mb-4 mb-lg-0 pt-4 "
              >
                <div className="card h-100">
                  <img
                    src={product.image[0]}
                    className="card-img-top p-5"
                    height={320}
                    width={100}
                    alt={product.title}
                  />
                  <div className="card-body">
                    <div className="d-flex justify-content-between">
                      <p className="small">{product.category.name}</p>
                    </div>

                    <div className="d-flex justify-content-between mb-3">
                      <Link href={`/products/${product.id}`}>
                        <h5 className="mb-0">{product.title}</h5>
                      </Link>
                      <h5 className="text-dark mb-0">${product.price}</h5>
                    </div>
                  </div>
```

```
                </div>
              </div>
            ))}
          </div>
        </div>
      </section>
    );
}

export default ProductsPage;
```

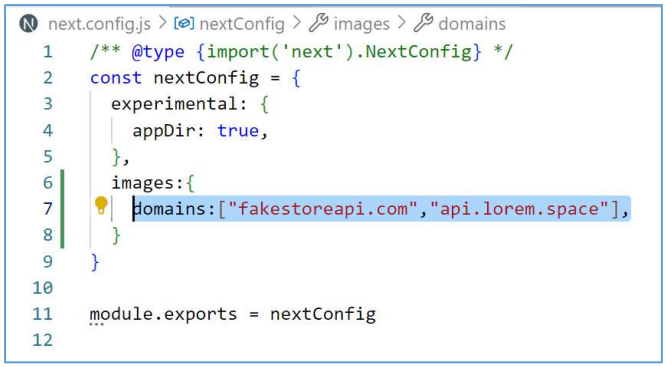
Dans le navigateur on va retrouver cette erreur :




***C'est qu'il faut déclarer les sites- des images dans next.config.js***


Il faut y ajouter

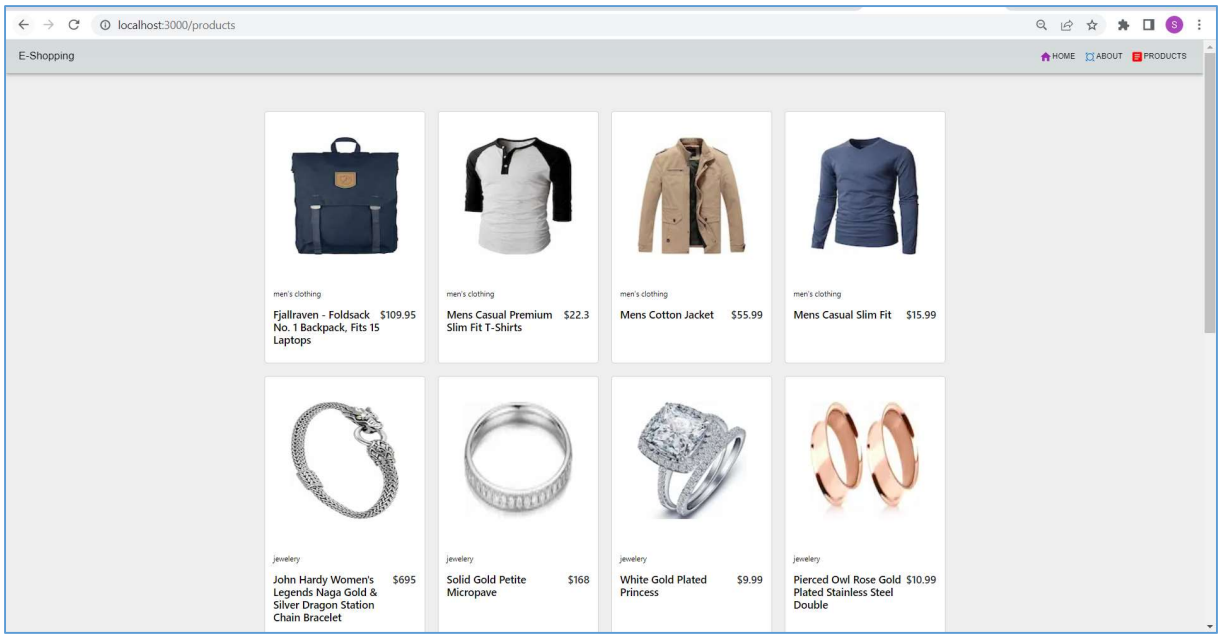
```
images:{
    domains:["fakestoreapi.com","api.lorem.space"],

  }
```




Puis redémarrer le serveur avec :

```
npm run dev
```

Puis actualiser la page dans le navigateur.


localhost:3000/products
E-Shopping
HOME
ABOUT
PRODUCTS
men's clothing
Fjallraven - Foldsack No. 1 Backpack, Fits 15 Laptops $109.95
men's clothing
Mens Casual Premium Slim Fit T-Shirts $22.3
men's clothing
Mens Cotton Jacket $55.99
men's clothing
Mens Casual Slim Fit $15.99
jewelery
John Hardy Women's Legends Naga Gold & Silver Dragon Station Chain Bracelet $695
jewelery
Solid Gold Petite Micropave $168
jewelery
White Gold Plated Princess $9.99
jewelery
Pierced Owl Rose Gold Plated Stainless Steel Double $10.99