# Give Link Exercise

**Γαλάνης Χρήστος**

**GitHub_Repo:**
**https://github.com/chrisgalanis/react-exercise**

# ΠΕΡΙΕΧΟΜΕΝΑ

# *Εισαγωγή*

What you need to install ?

1) Clone Repository:

```
git clone git@github.com:chrisgalanis/react-exercise.git
```

2) Install Next Js:

```
npm install next@latest react@latest react-dom@latest
```

Install Axios:

```
npm install axios
```

How to run the code ?

1) First type in a new terminal

```
npm run dev
```

2) Second open local host at : http://localhost:3000

## Axios

Το Axios χρησιμοποιείται ώστε να γίνουν fetch, τα δεδομένα από το link: 'https://be.givelink.app/data'. Συγκεκριμένα, τα δεδομένα αποτελούνται από αντικείμενα της javascript, σε μορφή JSON. Από όλα τα δεδομένα, επιλέγονται αυτά που περιέχουν πληροφορίες για τα προιοντα που είναι σε ανάγκη.

```
import axios from 'axios';

interface Product {
 imagePath: string;
 name: string;
 price: number;
 active: boolean;
}
```

Το interface Product, χρησιμοποιείται ώστε να ανακτηθούν δεδομένα με περιεχόμενα ( imagePath, name, price, active ), με βάση τα δεδομένα της άσκησης.

```
async function fetchData()
{
     try {
       const response = await
axios.get<Record<string,Product[]>>('https://be.givelink.app/data');

       console.log(Object.values(response.data.products));
        setProducts(Object.values(response.data.products));

     } catch (error) {
       console.error('Error fetching data:', error);
     }
}
```

Η συνάρτηση περιμένει να λάβει απάντηση από την ιστοσελίδα, στο παραπάνω link. Αποθηκεύει τα δεδομένα σε ένα dictionary αντικειμένων, που έχει σαν κλειδί το όνομα της πληροφοριας που ψάχνουμε και σαν δεδομένα Array τύπου Product interface.

Έπειτα ανανεώνουμε το Array της εφαρμογής με την setter μέθοδο: setProducts().

## Next.js / Typescript

Η εφαρμογή χρησιμοποιεί το React Framework, Next.js με στόχο το γρήγορο rendering σε στοιχεία που δεν ανανεώνονται συνεχώς.
Το Next.js παρέχει Server-Side Rendering και Client-Side rendering. Αυτή η παροχή μας επιτρέπει να αφήνουμε στην μεριά του Client, μόνο τα δυναμικά στοιχεία της εφαρμογής και στην πλευρά του server τα στατικά στοιχεία.

```
'use client'

import ProductCard from "./components/ProductCard";
import { useEffect, useState } from 'react';
import axios from 'axios';


// Object type Product need to have this Struct
interface Product {
 imagePath: string;
 name: string;
 price: number;
 active: boolean;
}

function App() {

 // Array of Products => Initialised empty
 const [products, setProducts] = useState<Product[]>([]);


 useEffect(() => {

   // Fetch Data from link : https://be.givelink.app/data' => Save
data in Product Array
   async function fetchData() {
     try
     {
       // Waiting for response from API endpoint
```

```
      const response = await
axios.get<Record<string,Product[]>>('https://be.givelink.app/data');

      // Set Array Product to the updated response
      setProducts(Object.values(response.data.products)); // => From
the response.data get only object with key 'product'

    } catch (error)
    {
      console.error('Error fetching data:', error);
    }
  }

  fetchData();
}, []);


return (
  <main>
    <div className='body'>
      <div className='container'>
        <h1> Λίστα Αναγκών μας </h1>
        <div className='product-list'>
          {

            products.map((product, index) => (
              <ProductCard key={index} product={product} />

            ))
          }
        </div>
      </div>
    </main>
);
}

export default App;
```

Στην περίπτωση αυτή, καθώς χρησιμοποιούμε το Hook useEffect, για να κάνουμε fetch τα δεδομένα η σελίδα που περιέχει την συνάρτηση App γίνεται rendered Client-side. Όπως δηλώνει και η λέξη στην πρώτη γραμμή.

Έπειτα για κάθε στοιχείο που ανήκει στο array product, δημιουργείται ένα αντικείμενο Product Card με ξεχωριστό κλειδί.

```
// Server Side Rendering Component
import React from 'react'


interface Product
{
 imagePath: string,
 name: string,
 price: number,
 active: boolean

}


const ProductCard = ({product} : {product : Product}) => {
 if (product.active)
 {
   return (
     <div className='product'>

         <img src={ "https://be.givelink.app/images/products/" +
product.imagePath } alt={ product.name }></img>

         <div className='product-info'>
             <p>{product.name}</p>
             <div className='price'>
                 <p>{product.price} €</p>
             </div>
         </div>
     </div>
   )
 }
 return(
   <></>
 )
}

export default ProductCard
```

Στο ProductCard, περνάμε σαν prop ένα αντικείμενο type δομής Product ( imagePath, name, price, active).

Για κάθε τέτοιο ξεχωριστό αντικείμενο προβάλουμε την φωτογραφία του, που την παίρνουμε από το link: `https://be.givelink.app/images/products/ + imagePath` και το όνομα και την τιμή του αντίστοιχου προϊόντος.

**Παρατήρηση:**
Δεν έχω κάνει βελτιστοποίηση χρησιμοποιώντας τον server-side, κάθως ήταν η πρωτή φορά που έγραφα σε Next.js , και δεν κατάφερα να βρώ τρόπο πως να κάνω fetch data χρησιμοποιώντας useEffect, αλλά να γίνει στο server side.

## Tailwind

Η εφαρμογή επίσης είναι responsive. Τοποθέτησα τα αντικείμενα σε grid Nx5, και μορφοποίησα τα κελιά με έμπνευση από την ιστοσελίδα της GiveLink.

Η σελίδα είναι χωρισμένη στα εξής μέρη :

- Body => ‘container’ => ‘product-list’ = > GRID Products

Η καρτέλα των προϊόντων έχει την εξής δομής

- ‘Product’ => img + ‘product-info’ => ‘price’

Μερικά μορφικά χαρακτηριστικά που έχουν γραφτεί την μορφή των shortcuts της Tailwind, για την καρτέλα των προϊόντων :

Grid με 5 στήλες και κενό 1.25rem :

```
.product-list
{
 @apply grid grid-cols-5 gap-5;
}
```

Άσπρο φόντο, με rounded ακρα, κείμενο στο κέντρο, και αλλαγή χρώματος όταν γίνεται hover !

```
.product
{
 @apply bg-white rounded-lg p-4 text-center shadow-lg  flex flex-col
items-center hover:bg-slate-300;
}
```

Εικόνα μέγεθος 6rem x 6 rem

```
.product img
{
 @apply w-24 h-24 ;
}
```

Οι 5 στήλες γίνονται 2 στήλες όταν έχουμε μετριο μέγεθος οθόνης και 1 στήλη για κινητά !

```
/* Responsive styles */
@media (max-width: 1200px) {
 .product-list {
   @apply grid-cols-2; /* Two columns on medium screens */
 }
}

@media (max-width: 768px) {
 .product-list {
   @apply grid-cols-1; /* Single column on small screens */
 }

 .container {
   @apply flex-col; /* Stack items vertically on small screens */
 }
```

```
@tailwind base;
@tailwind components;
@tailwind utilities;

:root
{
 --background-left: #7F4BEE;
 --background-right: #FF5758;
 --foreground: #171717;
 --product_name: #393939;
}

@media (prefers-color-scheme: dark)
{
 :root {
   --background-left: #7F4BEE;
   --background-right: #FF5758;
   --background: #0a0a0a;
   --foreground: #ededed;
 }
}


.body
{
 background: linear-gradient(to right,
var(--background-left), var(--background-right));
 @apply p-5;
}


.container
{
 @apply font-sans bg-indigo-100 text-gray-800 m-0 p-5 mx-auto
justify-center items-center min-h-screen rounded-lg;
}


h1
```

```
{
 @apply text-purple-600 text-center p-5 text-4xl;
}



/* Product List */

.product-list
{
 @apply grid grid-cols-5 gap-5;
}


.product
{
 @apply bg-white rounded-lg p-4 text-center shadow-lg  flex
flex-col items-center hover:bg-slate-300;
}

.product img
{
 @apply w-24 h-24 ;
}

.product-info
{
 @apply text-product_name my-2 truncate w-40;
}

.price
{
 @apply text-xl font-bold mt-2;
}


@layer utilities {
 .text-balance {
   text-wrap: balance;
```

```
 }
}

/* Media query for smaller screens */
@media (max-width: 500px) {
 .cart-card {
   @apply p-4;
 }

 .cart-card h2 {
   @apply text-xl;
 }

 .product-info img {
   @apply w-12 h-12;
 }
}

/* Responsive styles */
@media (max-width: 1200px) {
 .product-list {
   @apply grid-cols-2; /* Two columns on medium screens */
 }
}

@media (max-width: 768px) {
 .product-list {
   @apply grid-cols-1; /* Single column on small screens */
 }

 .container {
   @apply flex-col; /* Stack items vertically on small
screens */
 }


}
```