Read

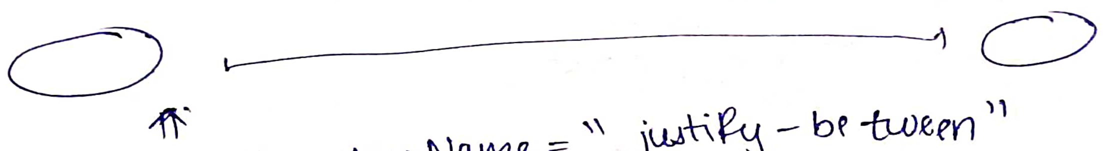
Library → react-router-~~DOM~~ dom

# import { BrowserRouter as Router, Switch, Route, Link }
from ~~Re~~ 'react-router-dom';

<Route path = "/" component = {Home} exact ></Route>

<Link to = "/" > Home </Link>

# Adding tailwind css in index.html through ~~cdn~~.
className = " container mx-auto flex items-center"



add className = " justify-between"
"py-4" → y-axis padding
" w-1/2" → 1/2 of width

We can add css instead of tailwind inside JS file also

```
const Navigation = () => {
    const cartstyle = {
        background: 'Red',
        display: 'flex'.
    }
    return (
        <>
          <nav style = {cartstyle}>
                                    </nav>
    );
}
```

grid grid-cols-5 gap-24 ← gap b/w them.
text-center

useEffect and UseState

import { useState, useEffect } from 'react';

To fetch data from ~~API~~ server (we will use browser fetch API)

Hooks → we can maintain state of application/component

In components → products.js

```
const Products = () => {
    const [products, setProducts] = useState([ ]);
```

it is an array of obj and function

destructuring →

we can change this by calling that

```
useEffect(() => {

}, [products]);
```

This is React Hook

→ array of dependencies

(when products will change then useEffect function will be called immediately)

(when the array is empty then it will run only one time after the component is mounted)

(So, here we can perfectly run the fetching the service)

~~useEffect~~

```
useEffect (() => {
    fetch ('api/products')
     .then (response => response.json())
     .then ( products => {
            setProducts (products);
     });
  }, [ ]);

  return (
        <div className ="grid - grid-cols-5 my-8 ">
         {
              products. map (product =>
                     < Product key ={product ._id }
                                    product ={product}/>)

              }
    )
```

In ~~components~~ component in Product.js

```
  Const Product = (props) => {
      return (
        // we can use props.product.name or price -or
                                                 image

      )

  }.
```

We can concatenate string with js ~~var~~ string variable

Example <Link to = {`/products/ ${product._id} `}>

We will make single page for each Product

- To fetch url parameter there is inbuilt hook
useParams

```
const SingleProduct = () => {
    const [product, setProduct] = useState({}),
    const params = useParams();

    useEffect(() => {
        fetch(`/api/products/${params._id}`
        .th

    }, [params._id]);

    return (
        //we can use ~~params~~.product --
    );
}.
```

We can make back button by using another Hook useHistory

```
const history = useHistory();
return (
    <button onClick={ () => {history.goback()}
```

```
products.map(product => <Product key={product._id}
                                  product={product} />
```

Here we are passing data from parent to child by using props

child → Product.js (Product)
Parent → Products.js ← in this we are adding import Product from './Product.js'

Making Add to Cart Page

Now we want the particular id of product from Product to Product.js (child to parent)

We will centralise the data (CONTEXT API). We can share the data to any component we want to. Here data is small we can use CONTEXT API. But we can do Redux.

We want ~~Product~~ Context data in Navigation, Products Page, Cart.

App.js

```
<CartContext.Provider>

      <Products>

 </CartContext.Provider>
```

Products.js

```
const Products = () => {
      const {name} = useContext(CartContext);

      return (
            <h1> Products {name} </h1>
      );
```

If we click the Add Button in a particular Product. It will give information of that product.

Product.js

```
const Product = (props) => {
    const {product} = props;
    const addToCart = (event, product) => {
        console.log(product);
    }
    return (
        ______
        <button onClick = {(e) => {addToCart(e, product)}}>
        ______
    );
```

We are now making localStorage DB for Cart

App.js

```
const [cart, setCart] = useState({})
// Fetch cart from local storage
useEffect(() => {
    const cart = window.localStorage.getItem('cart');
    setCart(JSON.parse(cart));
}, []);
return (
    <><Router>
        <CartContext.Provider value = {{ cart, setCart }}>
            ______ Homepage
            ______ ProductsPage ______
        </CartContext.Provider>
);
```

when site reloads it runs

passing local storage cart data

//whenever cart is changing then we need to save the cart to local storage.

```
useEffect(() => {
    window.localStorage.setItem('cart', JSON.stringify(cart));
}, [cart]);
```

Product.js

```
const {product} = props;
const addToCart = (event, products) => {
        event.preventDefault();
        let _cart = {...cart}  → add previous cart data
        if (!_cart.items) {
                _cart.items = {}
        }
        if (_cart.items[product._id]) {
                _cart.items[product._id] += 1;
        }
        else {
                _cart.items[product._id] = 1;
        }
        if (!_cart.totalItems) {
                _cart.totalItems = 0;
        }
        _cart.totalItems += 1
        setCart(_cart);
```

For the only first time when cart is whole empty

when similar product is exist or not

Totalitem in cart

saving the cart Data

Structure of Cart DB.

```
const cart = {
    items: {
        '6080--': 2
        '8070--': 3
    },
    totalItem = 5
}
```

Changing the Navigation data to Cart total Items

```
import {CartContext} from '../CartContext';
Const Navigation = () => {
        Const {cart} = useContext(CartContext);
        return (
            <>
                <span> {cart.totalItems} </span>
            </>
        );
}
```