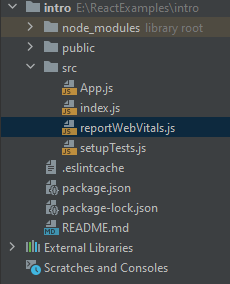
**REACT**

* npx create-react-app intro (intro isimli yeni proje) / npm i bootstrap / npm install reactstrap / npm install
* klasöre git / npm start



* Gereksiz dosyaları sildikten sonra projenin görünümü yukarıda ki şekilde olacaktır
* Component’ler 3 tipte olurlar (Fonksiyon componentleri, class componentler, react hooks componentler
* Navi, ProductList,CategoryList component’leri sırası ile projeye eklenir; rcc kısa yolu ile bir component’in syntax’ını otomatik olarak create edebiliriz.

import React, {Component} from 'react';  
  
class Navi extends Component {  
 */\*render tum sayfayi yenilemektense sadece bu component'in virtual dom uzerinde refresh edilmesini saglar\*/* render() {  
 return (  
 <div>  
 <h1>This is navi component</h1>  
 </div>  
 );  
 }  
}  
*/\*dis dunyadan erisebilmek icin componente erisimi aciyoruz\*/*export default Navi;

* İkinci componentimiz CategoryList componenti

import React, {Component} from 'react';  
  
class CategoryList extends Component {  
 render() {  
 return (  
 <div>  
 <h3>Category List</h3>  
 </div>  
 );  
 }  
}  
export default CategoryList;

* Üçüncü componentimiz ProductList componenti

import React, {Component} from 'react';  
  
class ProductList extends Component {  
 render() {  
 return (  
 <div>  
 <h3>Product List</h3>  
 </div>  
 );  
 }  
}  
export default ProductList;

* Index.js dosyası içerisinde bootstrap kütüphanesinin library’sini kullanılabilir hale getirdim

import 'bootstrap/dist/css/bootstrap.min.css'

* Oluşturduğumuz tüm componentleri app.js içerisinde kullanalım.App.js ana component’imiz, Navi component’ide en üstte yer alacak olan navigation bar olacak.CategoryList solda yer alacak ProductList onun sağında yer alacak

import {Col, Container, Row} from "reactstrap";  
import Navi from './Navi'  
import CategoryList from "./CategoryList";  
import ProductList from "./ProductList";  
  
function *App*() {  
 return (  
 <div className="App">  
 <Container>  
 <Row><Navi/></Row>  
 <Row>  
 <Col xs={3}><CategoryList/></Col>  
 <Col xs={9}><ProductList/></Col>  
 </Row>  
 </Container>  
 </div>  
 );  
}  
  
export default *App*;

* <div> tagları burada html’in aynısı olmasına rağmen aslında JSX render işlemini işaret etmektedir ve React’a aittir.Col değerleri görüldüğü üzere xs3 ve xs9 olarak işaretlendi.Yani CategoryList componenti 3 genişliğinde, ProductList componentide 9 genişliğinde.
* Component’leri aynı zamanda function olarakta tanımlayabiliriz. ProductList componentini örnek olması açısından function olarak’da denedim.
* import ***React***,{Component} from 'react';  
    
  function ProductList(){  
   return(  
   <div>  
   </div>  
   );  
  }  
    
  export default ProductList;

**PROPS**

* Component’ler arası veri taşımak için **props’lar** kullanılırlar
* Örnek olarak ProductList ve CategoryList component’lerinde title’i ortak kullanmak istiyorum. Bunun içinde app.js içerisinden ProductList ve CategoryList component’lerine title’i props ile göndermem gerekiyor.Bunun içinde app.js içerisinde değişkenler tanımlıyoruz

import {Col, Container, Row} from "reactstrap";  
import Navi from './Navi'  
import CategoryList from "./CategoryList";  
import ProductList from "./ProductList";  
  
function *App*() {  
 let categoryInfo = {title:"this is category title"};  
 let productInfo = {title:"this is product title"};  
 return (  
 <div className="App">  
 <Container>  
 <Row><Navi/></Row>  
 <Row>  
 <Col xs={3}><CategoryList info={categoryInfo}/></Col>  
 <Col xs={9}><ProductList info={productInfo}/></Col>  
 </Row>  
 </Container>  
 </div>  
 );  
}  
export default *App*;

* productInfo ve categoryInfo objelerini object olarak component’lere gönderiyoruz
* Ardından h3 etiketinde props kullanıyoruz.props’un içerisinde app.js içerisinden gelen title değerleri var. ProductList ve ComponentList adlı component’lerde aynı kullanımı gerçekleştiriyoruz

<h3>{this.props.info.title}</h3>

**STATE**

* State bir component için özel bir data anlamına gelirken, props component’ler arası data ve event taşımaya yarar.CategoryList componentine özel bir state ve bir değişken tanımlıyorum (currentCategory)

class CategoryList extends Component {  
 state = {  
 categories: [  
 {id: "1", categoryName: "Beverages"},  
 {id: "2", categoryName: "Condiment"},  
 {id: "3", categoryName: "Drinks"}  
 ],  
 currentCategory: ""  
 }  
  
 *changeCurrentCategoryAndSetTextH4Tag* = (cat) => {  
 this.setState({currentCategory: cat.categoryName})  
 }  
  
 render() {  
 return (  
 <div>  
 <h1>{this.props.info.title}</h1>  
 <ListGroup>  
 {this.state.categories.map(cat=>  
 <ListGroupItem  
 onClick={()=>this.*changeCurrentCategoryAndSetTextH4Tag*(cat)}  
 key={cat.id}>{cat.categoryName}  
 </ListGroupItem>  
 )}  
 </ListGroup>  
 <h4>{this.state.currentCategory}</h4>  
 </div>  
 );  
 }  
}  
  
export default CategoryList;

* State içerisinde hard code olarak 3 adet category ve currentCategory adında objeler create ediyorum.changeCurrentCategoryAndSetTextH4Tag adında ki method ile de Solda oluşacak olan categoryList objelerine mouse ile tıkladığımda bir h4 etiketi içerisinde tıkladığım nesnenin ismini ekrana basacak olan methodu create ettim. Şimdi CategoryList componenti icerisinde bir JSX’e ait ListGroup oluşturarak, içerisinde de state içinde dönerek her bir category’i ekrana ListGroupItem olarak basacak kodu yazıyorum.
* ListGroupItem içerisinde eğer key değişkenini tanımlamaz isek kodumuz hata verecektir. React’a özel bir güvenlik önlemidir.State içerisinde ki tüm değerleri ekrana ListGroupItem olarak basar
* ListGroupItem’a yazmış olduğumuz onClick eventi sayesinde hangi category’e tıklarsa h4 etiketi içerisinde o etiket gözükecek
* Şimdi bu işlemleri uyguladıktan sonra component’ler arası data taşıma işlemi react’de tek yönlü olarak çalıştığından dolayı bazı önlemler almamız gerekecek ve ileride redux kullanımına geçeceğiz.CategoryList componentinde ki itema tıkladığımız da ProductList componenti otomatik olarak seçilen category’e göre şekillenecek, yani CategoryList component ProductList componente tek taraflı bir şey göndermek zorunda kalacak. Bu problemin önüne geçmek içinde ana component olan app.js componentimizi function olmaktan çıkarıp class component haline getiriyoruz ve editliyoruz

class App extends Component{  
 */\*Category componentinden kesip buraya yapistirdigimiz state ve currentCategory degiskeni\*/  
 /\*currentCategory'i set ettigim anda props olarak bunu da Category componentine gondermem gerekiyor bunuda asagida gonderiyorum\*/* state = {  
 currentCategory:""  
 }  
 */\*Category Componentinden buraya taşıdık\*/  
 changeCurrentCategory* = (category)=>{  
 this.setState({currentCategory: category.categoryName})  
 }  
 productInfo = {title:"Product List"}  
 categoryInfo = {title:"Category List"}  
 render(){  
 return (  
 */\*Navi componentini direk burada kullanmis olduk\*/* <div>  
 <Container>  
 <Row>  
 <Navi/>  
 </Row>  
 <Row>  
 {*/\*props ile gonderdigim categoryInfo degiskeni gibi changeCurrentCategory methodunu da props ile category componentine gonderdim\*/*}  
 {*/\*currentCategory degiskenini de props olarak Category componentine gonderiyorum\*/*}  
 <Col xs={3}><CategoryList currentCategory={this.state.currentCategory} changeCategory={this.*changeCurrentCategory*} info={this.categoryInfo}/></Col>  
 {*/\*Current category bilgisine ProductList componentininde ihtiyaci olacagi icin kolaylikla ona da gecebilirim\*/*}  
 <Col xs={9}><ProductList currentCategory={this.state.currentCategory} info={this.productInfo}/></Col>  
 </Row>  
 </Container>  
 </div>  
 );  
 }  
}  
  
export default App;

* Category componentide duzenlendi

class CategoryList extends Component {  
 state = {  
 categories: [  
 {categoryId: "1", categoryName: "Beverages"},  
 {categoryId: "2", categoryName: "Condiment"}  
 ],  
 };  
  
 render() {  
 return (  
 <div>  
 <h3>{this.props.info.title}</h3>  
 <ListGroup>  
 {this.state.categories.map(cat=>  
 */\*burada onClick eventinde app.js ana componentinden gelen changeCategory adli degiskeni daha dogrusu function'i props araciligi ile aldik\*/* <ListGroupItem onClick={()=>this.props.changeCategory(cat)} key={cat.categoryId}>{cat.categoryName}</ListGroupItem>)  
 }  
 </ListGroup>  
 <h4>{this.props.currentCategory}</h4>  
 </div>  
 );  
 }  
}  
  
export default CategoryList;

**JSON-SERVER;**

Java yada C# gibi backend bir dille api yazamayacağımız için bu tür bir fake json server’i kullanacağız. İleri de nasıl kullanılacağı ile ilgili örnekleri göstereceğiz;

npm i json-server -g komutu ile kurulumu gerçeklenir.

Json-server normalde 3000 portunu kullandığından dolayı react’in portunu değiştirmemiz gerekiyor. Bunun içinde package.json içerisinde

"start": "set PORT=3001 && react-scripts start",

projemizin başlangıç portunu 3001 haline getirdik. Projenin klasörü içerisinde api adlı klasörü ekledim ve içerisine db.json adlı dosyayı create ettim

json-server --watch db.json ile fake server’imi başlatabilirim. Json-server 3000 numaralı portta çalışmaktadır.

Json Server’i install ettikten sonra datayı artık db.json dosyasından çekeceğimden dolayı CategoryList adlı componentime bir method yazmam gerekiyor;

*getCategories* = () => {  
 */\*fetch apiye ulaşmak için kullanılır\*/  
 fetch*(**"http://localhost:3000/categories"**)  
 .then(res => res.json()) */\*response u json'a dondur\*/* .then(data => **this**.setState({**categories**: data})); */\*yukarından gelen json datayı(data) categories state'i içerisine bas\*/*}

bu methodu yerleştirdiğimde şöyle bir problem ile karşılaşıyorum. React’de ilk önce component’ler yüklenir ardından içlerinde ki render methodları çalıştırılırlar.Ben bu methodu başlangıçta çalıştırabilmem için componentDidMount adlı methodu yazmalıyım;

componentDidMount() {  
 **this**.*getCategories*();  
}

component mount olduktan hemen sonra bu methodu call et.

Tüm işlemleri tamamladıktan sonra CategoryList component’inde cat.categoryId olarak geçen kısmıda db.json dosyasında sadece id kullanıldığı için şu şekilde değiştiriyoruz.

<**ListGroupItem onClick=**{() => **this**.**props**.**changeCategory**(cat)}  
 **key=**{cat.**id**}>{cat.categoryName}</**ListGroupItem**>)

ProductList’i categoryList’deki herhangi bir category’e tıkladığımda değişecek şekilde kullanmak istiyorum.Bunu yapabilmek için ana componentimde app.js içerisinde productlist’i çekeceğim ve prop olarak ProductList componentine göndereceğim.İlk önce app.js içerisinde state içerisinde bir productList array’i create ediyorum;

**state** = {  
 **currentCategory**:**""**,  
 **productList**:[]  
}

Ardından fetch ile datayı api üzerinden çekiyorum;

componentDidMount(){  
 **this**.*getProductList*();  
}

*getProductList* = ()=>{  
 *fetch*(**"http://localhost:3000/products"**)  
 .then(res => res.json())  
 .then(data=>**this**.setState({**productList**: data}));  
}

Gelen datayı prop olarak ProductList componentine gönderiyorum (this.state.productList);

<**Col xs=**{9}><**ProductList products=**{**this**.**state**.**productList**} **currentCategory=**{**this**.**state**.**currentCategory**} **info=**{**this**.**productInfo**}/></**Col**>

ProductList componentinde bir table (reactstrap sitesinden aldım) içerisinde props ile gelen tüm dataları gösteriyor olacağım;

<**Table**>  
 <**thead**>  
 <**tr**>  
 <**th**>#</**th**>  
 <**th**>Product Name</**th**>  
 <**th**>Quantity Per Unit</**th**>  
 <**th**>Unit Price</**th**>  
 <**th**>Units In Stock</**th**>  
 </**tr**>  
 </**thead**>  
 <**tbody**>  
 {**this**.**props**.**products**.map(product=>(  
 <**tr key=**{product.**id**}>  
 <**th scope="row"**>{product.**id**}</**th**>  
 <**td**>{product.productName}</**td**>  
 <**td**>{product.quantityPerUnit}</**td**>  
 <**td**>{product.unitPrice}</**td**>  
 <**td**>{product.unitsInStock}</**td**>  
 </**tr**>  
 ))}  
 </**tbody**>  
</**Table**>