3.Pizza.

1.Оборачиваем ссылку a в App компонентой <Button/>:

      <Button className="button--cart">

      <span>520 ₽</span>

      <div className="button\_\_delimiter"></div>

      <svg

        width="18"

        height="18"

        viewBox="0 0 18 18"

        fill="none"

        xmlns="http://www.w3.org/2000/svg"

      >

        <path

          d="M6.33333 16.3333C7.06971 16.3333 7.66667 15.7364 7.66667 15C7.66667 14.2636 7.06971 13.6667 6.33333 13.6667C5.59695 13.6667 5 14.2636 5 15C5 15.7364 5.59695 16.3333 6.33333 16.3333Z"

          stroke="white"

          strokeWidth="1.8"

          strokeLinecap="round"

          strokeLinejoin="round"

        />

        <path

          d="M14.3333 16.3333C15.0697 16.3333 15.6667 15.7364 15.6667 15C15.6667 14.2636 15.0697 13.6667 14.3333 13.6667C13.597 13.6667 13 14.2636 13 15C13 15.7364 13.597 16.3333 14.3333 16.3333Z"

          stroke="white"

          strokeWidth="1.8"

          strokeLinecap="round"

          strokeLinejoin="round"

        />

        <path

          d="M4.78002 4.99999H16.3334L15.2134 10.5933C15.1524 10.9003 14.9854 11.176 14.7417 11.3722C14.4979 11.5684 14.1929 11.6727 13.88 11.6667H6.83335C6.50781 11.6694 6.1925 11.553 5.94689 11.3393C5.70128 11.1256 5.54233 10.8295 5.50002 10.5067L4.48669 2.82666C4.44466 2.50615 4.28764 2.21182 4.04482 1.99844C3.80201 1.78505 3.48994 1.66715 3.16669 1.66666H1.66669"

          stroke="white"

          strokeWidth="1.8"

          strokeLinecap="round"

          strokeLinejoin="round"

        />

      </svg>

      <span>3</span>

    </Button>

2.Делаем небольшие изменения в компоненте Button для отображения второго класса:

function Button(props){

return <div><button onClick={function(){alert("Hello!")}}className={classNames("button",props.className,{"button--cart":props.outline})}>

{props.children}</button>

</div>

}

export default Button

Добавили props.className.

3.Создаем новую папку components и перекидываем туда наши компоненты.

4.Создаем компоненту Categories:

import React from "react";

const Categories=()=>{

return  <div className="categories">

   <ul>

     <li className="active">Все</li>

     <li>Мясные</li>

     <li>Вегетарианская</li>

     <li>Гриль</li>

     <li>Острые</li>

     <li>Закрытые</li>

   </ul>

 </div>

}

export default Categories

И подставляем ее в App:

<Categories/>

5.Устанавливаем расширение react snippets.

6.Создаем файл импортов:

import Categories from '../components/Categories';

import Header from '../components/Header';

export {

Categories,Header

}

7.Компоненту категории в App.js:

 <Categories className="active"

items={["Мясные","Вегетарианская","Гриль","Острые","Закрытые"]}/>

в Categories.js:

import React from "react";

const Categories=(props)=>{

return  <div className="categories">

----------------------------------------------------------------------------

<ul>

<li className="active">Все</li>

{props.items.map((elem,index)=>{

return <li key={`${elem}\_${index}`}

 > {elem}</li>

})}

</ul>

------------------------------------------------------------------------------

</div>

}

export default Categories

Создали уникальный key.

8.Свойства и методы в реакте работают сверху вниз,переданные в пропсах:

9.Создаем классовую компоненту CategoriesClass.Для изменений в компоненте используем setState:

import React from "react";

class CategoriesClass extends React.Component{

    state={

activeItem:null,

    }

    onSelectItem(index){

        this.setState({activeItem:index})

    }

render(){

return  <div className="categories">

<ul>

{this.props.items.map((elem,index)=>{

return <li className={this.state.activeItem===index?"active":""}

onClick ={()=>this.onSelectItem(index)} key={`${elem}\_${index}`} > {elem}</li>

})}

</ul>

</div>

}

}

export default CategoriesClass

Метод setState синергирует с реактом изменяет копию обьекта и оповещает ,что нужно сделать перерисовку.Менять можно только копии обьекта.Основной принцип реакта иммутабельность.

Если делать это не через setState,то придется использовать костыль:

onSelectItem(index){

        this.state.activeItem=index

this.forceUpdate()🡨--------------Принудительная перерисовка

    }

10.В классовой компоненте есть локальный стейт ,рендер,методы жизненного цикла,setState.Таким образом создавали компоненты раньше.Сейчас все даелают при помощи хуков.

11.Создаем функциональную компоненту CategoriesWithHooks:

import React from "react";

import { useState } from "react";

const CategoriesWithHookS=(props)=>{

const[activeItem,setActiveItem]=useState(null)

function onSelectItem(index){

  setActiveItem(index)

}

return  <div className="categories">

<ul>

{props.items.map((elem,index)=>{

return <li  className={activeItem===index?"active":""}

 onClick ={()=>onSelectItem(index)} key={`${elem}\_${index}`} > {elem}</li>

})}

</ul>

</div>

}

export default CategoriesWithHookS

12.Хуки работают по другому.Функциональные работают не так как классовые .У них есть состояния.

При помощи setActiveItem мы можем изменять состояние и реакт об этом узнает и перерисовует компоненту.

Зануляем наше состояние:

import React from "react";

import { useState } from "react";

const CategoriesWithHookS=(props)=>{

const[activeItem,setActiveItem]=useState(null)

function onSelectItem(index){

  setActiveItem(index)

}

return  <div className="categories">

<ul>

<li  className={activeItem===null?"active":""}

 onClick ={()=>onSelectItem(null)}>Все</li>

{props.items.map((elem,index)=>{

return <li  className={activeItem===index?"active":""}

 onClick ={()=>onSelectItem(index)} key={`${elem}\_${index}`} > {elem}</li>

})}

</ul>

</div>

}

export default CategoriesWithHookS

13.Делаем проверку на псевдоистину:

{props.items && props.items.map((elem,index)=>{

return <li  className={activeItem===index?"active":""}

 onClick ={()=>onSelectItem(index)} key={`${elem}\_${index}`} > {elem}</li>

})}

В Js есть пять псевдоложных значений:

* undefined
* null
* 0
* пустая строка
* NaN