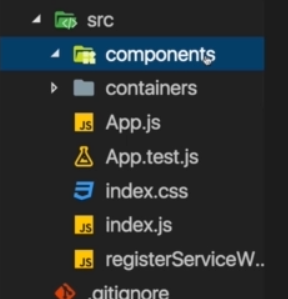
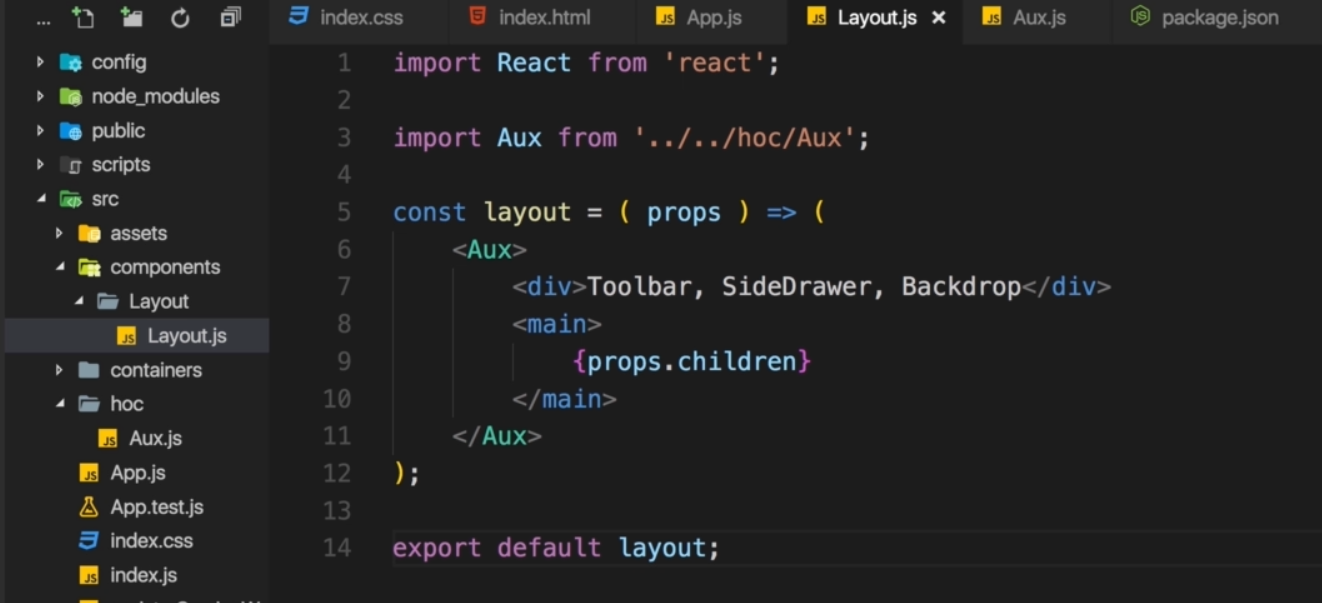
Có thể thêm font, css file … vào public/index.html

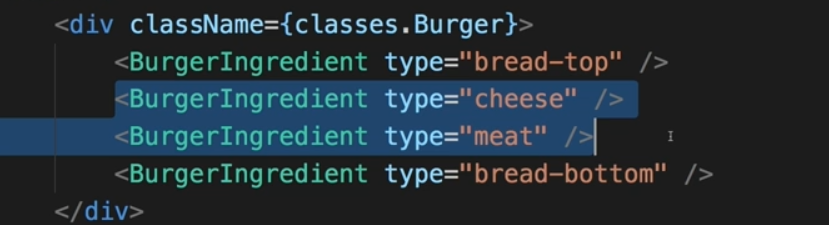
componets: chứa các dump component, nghĩa là các thành phần lẻ, ko có state

containers: các thành phần lớn, có cấu trúc, có state  
Layout là compent có thanh công cụ, thanh định hướng

Dùng nó để chứa nội dung page



Cách render ntn ko hiệu quả



Nên dùng cách này

Các thành phần nhỏ có thể tổ chức bằng swich để render linh động theo props được truyền vào, ở đây sẽ render từng kiểu riêng phụ thuộc vào thông số truyền vào props.type

Tự đó dùng thêm state để quản lý

Và dùng loop để render

* Tọa được cấu trúc dễ xử lý, bảo trì

import React, { Component } from 'react';

import PropTypes from 'prop-types';

import classes from './BurgerIngredient.css';

class BurgerIngredient extends Component {

    render () {

        let ingredient = null;

        switch ( this.props.type ) {

            case ( 'bread-bottom' ):

                ingredient = <div className={classes.BreadBottom}></div>;

                break;

            case ( 'bread-top' ):

                ingredient = (

                    <div className={classes.BreadTop}>

                        <div className={classes.Seeds1}></div>

                        <div className={classes.Seeds2}></div>

                    </div>

                );

                break;

            case ( 'meat' ):

                ingredient = <div className={classes.Meat}></div>;

                break;

            case ( 'cheese' ):

                ingredient = <div className={classes.Cheese}></div>;

                break;

            case ( 'bacon' ):

                ingredient = <div className={classes.Bacon}></div>;

                break;

            case ( 'salad' ):

                ingredient = <div className={classes.Salad}></div>;

                break;

            default:

                ingredient = null;

        }

        return ingredient;

    }

}

BurgerIngredient.propTypes = {

    type: PropTypes.string.isRequired

};

export default BurgerIngredient;

css style

xem file css tham khảo các tạo ra hình vẽ bằng css

burger sử dụng burgerIngerdient

loop qua props ( props này là object và được lấy từ state, có thể là state trong componnet này hoặc của componet cha truyền qua props)

object.key(biến object) trả về mãng chứa keys của object theo thứ tự: loop qua mãng này dùng map để tạo ra mãng các componnent cần render

đặc biệt riêng ở bài này:

props chứa tên của các thành phần và số lượng

nên tạo ra mãng có số phần từ bằng số lượng của mỗi thành phần khi loop qua

loop qua mãng đó, mỗi lần return ra phần tử đó với type là name của state

vậy mãng trả về sẽ có dạng mãng chứa mãng các phần tử, các phần tử lại là mãng chứa phần tử: [[a,b],a,[a,b,c]]

transfromIngredient lưu mãng đó

Để dể hình dung nên tách ra từng phần rồi loop

Dùng reduce và concat nối hàm đưa về hàm chuẩn

import React from 'react';

import classes from './Burger.css';

import BurgerIngredient from './BurgerIngredient/BurgerIngredient';

const burger = ( props ) => {

    let transformedIngredients = Object.keys( props.ingredients )

        .map( igKey => {

            return [...Array( props.ingredients[igKey] )].map( ( \_, i ) => {

                return <BurgerIngredient key={igKey + i} type={igKey} />;

            } );

        } )

        .reduce((arr, el) => {

            return arr.concat(el)

        }, []);

    if (transformedIngredients.length === 0) {

        transformedIngredients = <p>Please start adding ingredients!</p>;

    }

    return (

        <div className={classes.Burger}>

            <BurgerIngredient type="bread-top" />

            {transformedIngredients}

            <BurgerIngredient type="bread-bottom" />

        </div>

    );

};

export default burger;

class của mỗi thành phần sẽ đi đôi với nó, responsive cũng vậy

.Burger {

    width: 100%;

    margin: auto;

    height: 250px;

    overflow: scroll;

    text-align: center;

    font-weight: bold;

    font-size: 1.2rem;

}

@media (min-width: 500px) and (min-height: 400px) {

    .Burger {

        width: 350px;

        height: 300px;

    }

}

@media (min-width: 500px) and (min-height: 401px) {

    .Burger {

        width: 450px;

        height: 400px;

    }

}

@media (min-width: 1000px) and (min-height: 700px) {

    .Burger {

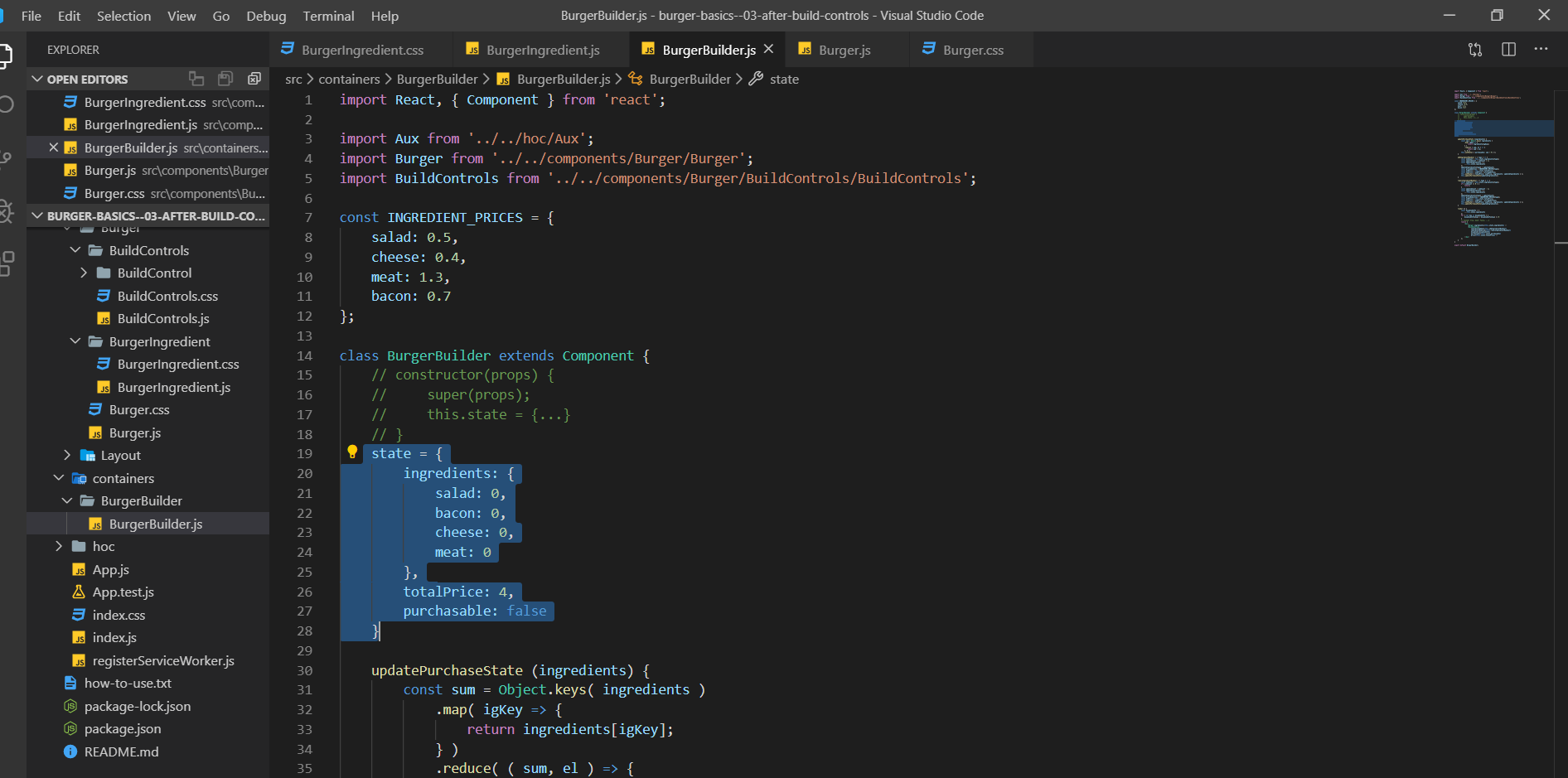
        width: 700px;

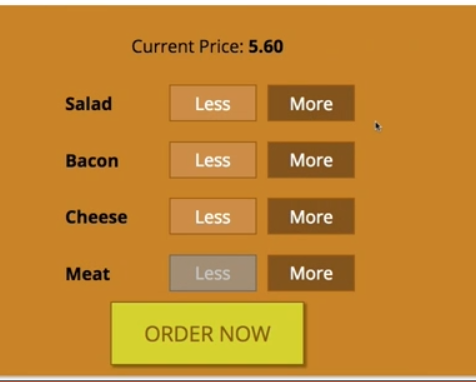
        height: 600px;

    }

}

Trên component cha lưu type và số lượng của component





Build control: là từng hàng trong hình trên, vì các hàng tương tự nhau nên tạo riêng từng hàng để rồi loop ra nhiều hàng

Onclick, disable … được truyền qua props đưa lên class cha ( hoặc cha của cha) để xử lý

import React from 'react';

import classes from './BuildControl.css';

const buildControl = (props) => (

    <div className={classes.BuildControl}>

        <div className={classes.Label}>{props.label}</div>

        <button

            className={classes.Less}

            onClick={props.removed}

            disabled={props.disabled}>Less</button>

        <button

            className={classes.More}

            onClick={props.added}>More</button>

    </div>

);

export default buildControl;

thuộc tính disable nhận từ props true hoặc false đế disable

.BuildControl button:disabled {

    background-color: #AC9980;

    border: 1px solid #7E7365;

    color: #ccc;

    cursor: default;

}

.BuildControl button:hover:disabled {

    background-color: #AC9980;

    color: #ccc;

    cursor: not-allowed;

}

Tạo mãng chứa các thông số cần truyển để tọa ra các dòng ctrol, các thông số để truyền ko cần lưu lại nên có thể tạo ở đây

Khi nhấn nút sẽ thay đổi các thông số trong state( component cha)

Loop qua mãng đó để render ra từng dòng

Tạo button để order

Price được truyền từ state của class cha xuống để hiển thị

import React from 'react';

import classes from './BuildControls.css';

import BuildControl from './BuildControl/BuildControl';

const controls = [

    { label: 'Salad', type: 'salad' },

    { label: 'Bacon', type: 'bacon' },

    { label: 'Cheese', type: 'cheese' },

    { label: 'Meat', type: 'meat' },

];

const buildControls = (props) => (

    <div className={classes.BuildControls}>

        <p>Current Price: <strong>{props.price.toFixed(2)}</strong></p>

        {controls.map(ctrl => (

            <BuildControl

                key={ctrl.label}

                label={ctrl.label}

                added={() => props.ingredientAdded(ctrl.type)}

                removed={() => props.ingredientRemoved(ctrl.type)}

                disabled={props.disabled[ctrl.type]} />

        ))}

        <button

            className={classes.OrderButton}

            disabled={!props.purchasable}>ORDER NOW</button>

    </div>

);

export default buildControls;

hàm add và remove, update state nào được chọn +1 vào state đó hoặc -1, đồng thời thay đổi price vì khi add price sẽ thay đổi

hàm này thực hiện trên BurgerBuider là class cha của buildControl và chứa state

addIngredientHandler = ( type ) => {

        const oldCount = this.state.ingredients[type];

        const updatedCount = oldCount + 1;

        const updatedIngredients = {

            ...this.state.ingredients

        };

        updatedIngredients[type] = updatedCount;

        const priceAddition = INGREDIENT\_PRICES[type];

        const oldPrice = this.state.totalPrice;

        const newPrice = oldPrice + priceAddition;

        this.setState( { totalPrice: newPrice, ingredients: updatedIngredients } );

        this.updatePurchaseState(updatedIngredients);

    }

    removeIngredientHandler = ( type ) => {

        const oldCount = this.state.ingredients[type];

        if ( oldCount <= 0 ) {

            return;

        }

        const updatedCount = oldCount - 1;

        const updatedIngredients = {

            ...this.state.ingredients

        };

        updatedIngredients[type] = updatedCount;

        const priceDeduction = INGREDIENT\_PRICES[type];

        const oldPrice = this.state.totalPrice;

        const newPrice = oldPrice - priceDeduction;

        this.setState( { totalPrice: newPrice, ingredients: updatedIngredients } );

        this.updatePurchaseState(updatedIngredients);

    }

Price của từng thần phân được lưu ở ngoài

Hàm add hoặc remove lấy dữ liệu này để cộng trừ để setState cho totalprice

Total price giá ban đầu là 4 ( vì mặc định có 2 lát bánh rồi)

import Aux from '../../hoc/Aux';

import Burger from '../../components/Burger/Burger';

import BuildControls from '../../components/Burger/BuildControls/BuildControls';

const INGREDIENT\_PRICES = {

    salad: 0.5,

    cheese: 0.4,

    meat: 1.3,

    bacon: 0.7

};

class BurgerBuilder extends Component {

    // constructor(props) {

    //     super(props);

    //     this.state = {...}

    // }

    state = {

        ingredients: {

            salad: 0,

            bacon: 0,

            cheese: 0,

            meat: 0

        },

        totalPrice: 4,

        purchasable: false

    }

Gọi Builcontrol

Disable infor được truyên nguyên mãng, bởi vì cái componnt còn buildControl xử lý tất cả từng dòng

 render () {

        const disabledInfo = {

            ...this.state.ingredients

        };

        for ( let key in disabledInfo ) {

            disabledInfo[key] = disabledInfo[key] <= 0

        }

        // {salad: true, meat: false, ...}

        return (

            <Aux>

                <Burger ingredients={this.state.ingredients} />

                <BuildControls

                    ingredientAdded={this.addIngredientHandler}

                    ingredientRemoved={this.removeIngredientHandler}

                    disabled={disabledInfo}

                    purchasable={this.state.purchasable}

                    price={this.state.totalPrice} />

            </Aux>

        );

    }

Hàm này được gọi mỗi khi thêm thành phần hoặc bớt -> điều khiển disabel của nút order

   updatePurchaseState (ingredients) {

        const sum = Object.keys( ingredients )

            .map( igKey => {

                return ingredients[igKey];

            } )

            .reduce( ( sum, el ) => {

                return sum + el;

            }, 0 );

        this.setState( { purchasable: sum > 0 } );

    }

Button:

import React from 'react';

import classes from './Button.css';

const button = (props) => (

    <button

        className={[classes.Button, classes[props.btnType]].join(' ')}

        onClick={props.clicked}>{props.children}</button>

);

export default button;

.Button {

    background-color: transparent;

    border: none;

    color: white;

    outline: none;

    cursor: pointer;

    font: inherit;

    padding: 10px;

    margin: 10px;

    font-weight: bold;

}

.Button:first-of-type {

    margin-left: 0;

    padding-left: 0;

}

.Button:disabled {

    color: #ccc;

    cursor: not-allowed;

}

.Success {

    color: #5C9210;

}

.Danger {

    color: #944317;

}

Sử dụng button

     <Button btnType="Danger" clicked={this.props.purchaseCancelled}>CANCEL</Button>

                <Button btnType="Success" clicked={this.props.purchaseContinued}>CONTINUE</Button>

backdrop

import React from 'react';

import classes from './Backdrop.css';

const backdrop = (props) => (

    props.show ? <div className={classes.Backdrop} onClick={props.clicked}></div> : null

);

export default backdrop;

.Backdrop {

    width: 100%;

    height: 100%;

    position: fixed;

    z-index: 100;

    left: 0;

    top: 0;

    background-color: rgba(0, 0, 0, 0.5);

}

Modal

import React, { Component } from 'react';

import classes from './Modal.css';

import Aux from '../../../hoc/Aux/Aux';

import Backdrop from '../Backdrop/Backdrop';

class Modal extends Component {

    shouldComponentUpdate ( nextProps, nextState ) {

        return nextProps.show !== this.props.show;

    }

    componentWillUpdate () {

        console.log('[Modal] WillUpdate');

    }

    render () {

        return (

            <Aux>

                <Backdrop show={this.props.show} clicked={this.props.modalClosed} />

                <div

                    className={classes.Modal}

                    style={{

                        transform: this.props.show ? 'translateY(0)' : 'translateY(-100vh)',

                        opacity: this.props.show ? '1' : '0'

                    }}>

                    {this.props.children}

                </div>

            </Aux>

        )

    }

}

export default Modal;

.Modal {

    position: fixed;

    z-index: 500;

    background-color: white;

    width: 70%;

    border: 1px solid #ccc;

    box-shadow: 1px 1px 1px black;

    padding: 16px;

    left: 15%;

    top: 30%;

    box-sizing: border-box;

    transition: all 0.3s ease-out;

}

@media (min-width: 600px) {

    .Modal {

        width: 500px;

        left: calc(50% - 250px);

    }

}

There are two other built-in CSS timing functions:

* ease-in: slow at the beginning, fast/abrupt at the end
* ease-out: fast/abrupt at the beginning, slow at the end

While they make a certain intuitive sense looked at that way, the general "rule" (in quotes) is to use them *opposite* of how they are named:

* ease-in: when things are moving *out*.
* ease-out: when things are moving *in*.

dùng modal

sử dụng wrap 1 thành phần nào đó

show= một biến state true thì hiện modal, false thì k hiện

modalClosed trỏ đến một hàm set biến purchasing=false để close modal, và hàm này sẽ trỏ đến click của backdrop như vậy khi click vào backdrop thì sẽ close modal

 <button

            className={classes.OrderButton}

            disabled={!props.purchasable}

            onClick={props.ordered}>ORDER NOW</button>

nut nhấn trên Buildcontrol sẽ trỏ đển ordered

ordered trỏ đến hàm set puchasing true để hiện backdrop

 return (

            <Aux>

                <Modal show={this.state.purchasing} modalClosed={this.purchaseCancelHandler}>

                    <OrderSummary

                        ingredients={this.state.ingredients}

                        price={this.state.totalPrice}

                        purchaseCancelled={this.purchaseCancelHandler}

                        purchaseContinued={this.purchaseContinueHandler} />

                </Modal>

                <Burger ingredients={this.state.ingredients} />

                <BuildControls

                    ingredientAdded={this.addIngredientHandler}

                    ingredientRemoved={this.removeIngredientHandler}

                    disabled={disabledInfo}

                    purchasable={this.state.purchasable}

                    ordered={this.purchaseHandler}

                    price={this.state.totalPrice} />

            </Aux>

    purchaseHandler = () => {

        this.setState({purchasing: true});

    }

    purchaseCancelHandler = () => {

        this.setState({purchasing: false});

    }

Order sumary

Order sumary đơn giản chỉ là một component

import React, { Component } from 'react';

import Aux from '../../../hoc/Aux/Aux';

import Button from '../../UI/Button/Button';

class OrderSummary extends Component {

    // This could be a functional component, doesn't have to be a class

    componentWillUpdate() {

        console.log('[OrderSummary] WillUpdate');

    }

    render () {

        const ingredientSummary = Object.keys( this.props.ingredients )

            .map( igKey => {

                return (

                    <li key={igKey}>

                        <span style={{ textTransform: 'capitalize' }}>{igKey}</span>: {this.props.ingredients[igKey]}

                    </li> );

            } );

        return (

            <Aux>

                <h3>Your Order</h3>

                <p>A delicious burger with the following ingredients:</p>

                <ul>

                    {ingredientSummary}

                </ul>

                <p><strong>Total Price: {this.props.price.toFixed( 2 )}</strong></p>

                <p>Continue to Checkout?</p>

                <Button btnType="Danger" clicked={this.props.purchaseCancelled}>CANCEL</Button>

                <Button btnType="Success" clicked={this.props.purchaseContinued}>CONTINUE</Button>

            </Aux>

        );

    }

}

export default OrderSummary;

toolbar: thanh công cụ nganh phía trên

chứa logo và navigation item, điểm đặc biệt là toobar luôn nằm đó, và có thể ẩn hiện

import React from 'react';

import classes from './Toolbar.css';

import Logo from '../../Logo/Logo';

import NavigationItems from '../NavigationItems/NavigationItems';

import DrawerToggle from '../SideDrawer/DrawerToggle/DrawerToggle';

const toolbar = ( props ) => (

    <header className={classes.Toolbar}>

        <DrawerToggle clicked={props.drawerToggleClicked} />

        <div className={classes.Logo}>

            <Logo />

        </div>

        <nav className={classes.DesktopOnly}>

            <NavigationItems />

        </nav>

    </header>

);

export default toolbar;

.Toolbar {

    height: 56px;

    width: 100%;

    position: fixed;

    top: 0;

    left: 0;

    background-color: #703B09;

    display: flex;

    justify-content: space-between;

    align-items: center;

    padding: 0 20px;

    box-sizing: border-box;

    z-index: 90;

}

.Toolbar nav {

    height: 100%;

}

.Logo {

    height: 80%;

}

@media (max-width: 499px) { với width nhỏ hơn 499 ko hiển thị

    .DesktopOnly {

        display: none; //resposive, nghĩa là thanh nav sẽ ko hiện khi ở mobile thay vào đó dùng sidedraw

    }

}

import React from 'react';

import classes from './NavigationItems.css';

import NavigationItem from './NavigationItem/NavigationItem';

const navigationItems = () => (

    <ul className={classes.NavigationItems}>

        <NavigationItem link="/" active>Burger Builder</NavigationItem>

        <NavigationItem link="/">Checkout</NavigationItem>

    </ul>

);

export default navigationItems;

css

.NavigationItems {

    margin: 0;

    padding: 0;

    list-style: none;

    display: flex;

    flex-flow: column;

    align-items: center;

    height: 100%;

}

@media (min-width: 500px) {

    .NavigationItems {

        flex-flow: row;

    }

}

import React from 'react';

import classes from './NavigationItem.css';

const navigationItem = ( props ) => (

    <li className={classes.NavigationItem}>

        <a

            href={props.link}

            className={props.active ? classes.active : null}>{props.children}</a>

    </li>

);

export default navigationItem;

.NavigationItem {

    margin: 10px 0;

    box-sizing: border-box;

    display: block;

    width: 100%;

}

.NavigationItem a {

    color: #8F5C2C;

    text-decoration: none;

    width: 100%;

    box-sizing: border-box;

    display: block;

}

.NavigationItem a:hover,

.NavigationItem a:active,

.NavigationItem a.active {

    color: #40A4C8;

}

@media (min-width: 500px) {

    .NavigationItem {

        margin: 0;

        display: flex;

        height: 100%;

        width: auto;

        align-items: center;

    }

    .NavigationItem a {

        color: white;

        height: 100%;

        padding: 16px 10px;

        border-bottom: 4px solid transparent;

    }

    .NavigationItem a:hover,

    .NavigationItem a:active,

    .NavigationItem a.active {

        background-color: #8F5C2C;

        border-bottom: 4px solid #40A4C8;

        color: white;

    }

}

Dùng toolbar và sidedraw trong layout

import React, { Component } from 'react';

import Aux from '../Aux/Aux';

import classes from './Layout.css';

import Toolbar from '../../components/Navigation/Toolbar/Toolbar';

import SideDrawer from '../../components/Navigation/SideDrawer/SideDrawer';

class Layout extends Component {

    state = {

        showSideDrawer: false

    }

    sideDrawerClosedHandler = () => {

        this.setState( { showSideDrawer: false } );

    }

    sideDrawerToggleHandler = () => {

        this.setState( ( prevState ) => {

            return { showSideDrawer: !prevState.showSideDrawer };

        } );

    }

    render () {

        return (

            <Aux>

                <Toolbar drawerToggleClicked={this.sideDrawerToggleHandler} />

**<SideDrawer**

                    open={this.state.showSideDrawer}

                    closed={this.sideDrawerClosedHandler} />

                <main className={classes.Content}>

                    {this.props.children}

                </main>

            </Aux>

        )

    }

}

export default Layout;

logo

import React from 'react';

import burgerLogo from '../../assets/images/burger-logo.png';

import classes from './Logo.css';

const logo = (props) => (

    <div className={classes.Logo} style={{height: props.height}}>

        <img src={burgerLogo} alt="MyBurger" />

    </div>

);

export default logo;

.Logo {

    background-color: white;

    padding: 8px;

    height: 100%;

    box-sizing: border-box;

    border-radius: 5px;

}

.Logo img {

    height: 100%;

}

Sidedraw sẽ hiện ra khi nhấn nút trên nav

import React from 'react';

import Logo from '../../Logo/Logo';

import NavigationItems from '../NavigationItems/NavigationItems';

import classes from './SideDrawer.css';

import Backdrop from '../../UI/Backdrop/Backdrop';

import Aux from '../../../hoc/Aux/Aux';

const sideDrawer = ( props ) => {

    let attachedClasses = [classes.SideDrawer, classes.Close];

    if (props.open) {

        attachedClasses = [classes.SideDrawer, classes.Open];

    }

    return (

        <Aux>

            <Backdrop show={props.open} clicked={props.closed}/>

            <div className={attachedClasses.join(' ')}>

                <div className={classes.Logo}>

                    <Logo />

                </div>

                <nav>

                    <NavigationItems />

                </nav>

            </div>

        </Aux>

    );

};

export default sideDrawer;

điểm hay là sử dụng open và close css class để hiện hoặc không hiện sidedraw

.SideDrawer {

    position: fixed;

    width: 280px;

    max-width: 70%;

    height: 100%;

    left: 0;

    top: 0;

    z-index: 200;

    background-color: white;

    padding: 32px 16px;

    box-sizing: border-box;

    transition: transform 0.3s ease-out;

}

@media (min-width: 500px) { với widht lớn hơn 500px, ko hiển thị

    .SideDrawer {

        display: none;

    }

}

.Open {

    transform: translateX(0);

}

.Close {

    transform: translateX(-100%);

}

.Logo {

    height: 11%;

    margin-bottom: 32px;

}

drawerToggle: thanh ngang click vào hiện ra sideDraw, nhìn vào tootbar để hiểu cách dùng

import React from 'react';

import classes from './DrawerToggle.css';

const drawerToggle = (props) => (

    <div className={classes.DrawerToggle} onClick={props.clicked}>

        <div></div>

        <div></div>

        <div></div>

    </div>

);

export default drawerToggle;

khi width lớn hơn 500px thì ko hiển thị

.DrawerToggle {

    width: 40px;

    height: 100%;

    display: flex;

    flex-flow: column;

    justify-content: space-around;

    align-items: center;

    padding: 10px 0;

    box-sizing: border-box;

    cursor: pointer;

}

.DrawerToggle div {

    width: 90%;

    height: 3px;

    background-color: white;

}

@media (min-width: 500px) {

    .DrawerToggle {

        display: none;

    }

}

Khi thay tác trong burger hoặc builder control mặc dù ko thay đổi orderSumary nhưng nó vẫn render lại

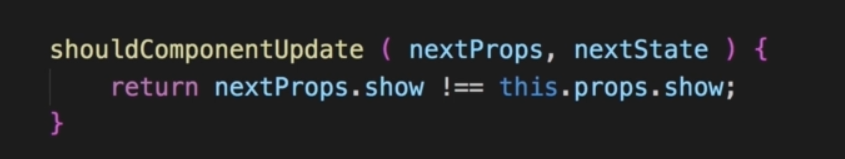
  componentWillUpdate() {

        console.log('[OrderSummary] WillUpdate');

    }

Dùng hàm này trong order sumary để xem update

* dùng shouldupdate để queyets định update hay ko



Trong modal vì modal chứa ordersumary

   shouldComponentUpdate ( nextProps, nextState ) {

        return nextProps.show !== this.props.show;

    }

    componentWillUpdate () {

        console.log('[Modal] WillUpdate');

    }

   <Aux>

                <Modal show={this.state.purchasing} modalClosed={this.purchaseCancelHandler}>

                    <OrderSummary

                        ingredients={this.state.ingredients}

                        price={this.state.totalPrice}

                        purchaseCancelled={this.purchaseCancelHandler}

                        purchaseContinued={this.purchaseContinueHandler} />

                </Modal>

                <Burger ingredients={this.state.ingredients} />

                <BuildControls

                    ingredientAdded={this.addIngredientHandler}

                    ingredientRemoved={this.removeIngredientHandler}

                    disabled={disabledInfo}

                    purchasable={this.state.purchasable}

                    ordered={this.purchaseHandler}

                    price={this.state.totalPrice} />

            </Aux>