Для того чтобы запустить реакт у нас в проекте, переходим <https://github.com/facebook/create-react-app>

В консоли node.js выбираем папку с проектом и пишим команду

Чтобы выбрать папку пишем сначала d: - перешли на диск D

А далее cd PROJECT\ React.js\ 01-first-project

Теперь мы вбиваем npx create-react-app my-app

my-app ‘то название проекта, Которое может быть любым

После установки командой запускаем реакт, предварительно перейдя в папку my-app

И устанавливается пакет node\_modules, который в дальнейшем я буду устнавливать по аналогии с gulp, тоесть просто копируя в новый проект файл package.json

class в реакте ClassName

**Что такое компонета**

Компонента – это функция, возвращающая разметку (jsx(скриптовый html))

**const** *App* = () =>  
{  
 **return**(  
 <**div className="App"**>  
 <**p className="ds"**>Привет Наташка</**p**>  
 </**div**>  
 )  
};

App – это компонента, то есть функция, которая возвращает html размету. Название компоненты должно быть с большой буквы!!! Далее попробую прописать дальнейшую судьбу этого компонента.

Вызывается он в файле index.js, который находится на одном уровне с файлом App.js, в котором находится компонент App

………

**import** *App* **from './App'**; импортируй компонент App из файла App

………

ReactDOM.render(<**App** />, ***document***.getElementById(**'root'**));

Переводится это буквально так – ReactDOM.отрисуй тэг <App />, а это не что инное, как компонент(функция) App,которую мы вставляем как тэг, и вставляем его в крнтейнер с айдишником **root** , котрый находится в файле index.html . А в этом файле всего лишь пустой контейнер, в который в последствии с помощью js будет формироваться разметка страницы.

Реакт всецело состоит из компонентов. Для примера мы один компонент можем составить из разных компонентов, кторые мы будем вызывать:

**const** *App* = () =>  
{  
 **return**(  
 <**div className="container App"**>  
 <**Fun\_header**/>  
 <**Fun\_footer**/>  
 </**div**>  
 )  
};

**const** *Fun\_header*=()=>  
{  
 **return**(  
 <**header**>  
 <**p**>Это классный хедер</**p**>  
 </**header**>  
 )  
};  
**const** *Fun\_footer* = () =>  
{  
 **return**(  
 <**footer**>  
 <**p**>Это классный футер</**p**>  
 </**footer**>  
 )  
};

**export default** *App*;

Но это не является правильным.Каждый компонент должен быть отдельным файлом. Поэтому отдельно создаем файл Header.js =>

Отдельно создаем Footer.js

**const** *Fun\_footer* = () =>  
{  
 **return**(  
 <**footer**>  
 <**p**>Это классный футер</**p**>  
 </**footer**>  
 )  
};  
  
**export default** *Fun\_footer*;

**const** *Fun\_header*=()=>  
{  
 **return**(  
 <**header**>  
 <**p**>Это классный хедер</**p**>  
 </**header**>  
 )  
};  
  
**export default** *Fun\_header*;

И в файле App.js прописываем

**import** *Fun\_header* **from './Header'**;  
**import** *Fun\_footer* **from './Footer'**;  
  
**const** *App* = () =>  
{  
 **return**(  
 <**div className="container App"**>  
 <**Fun\_header**/>  
 <**Fun\_footer**/>  
 </**div**>  
 )  
};

**Стили. CSS модули**

Для каждого компонента можно делать отдельный css файл. Есть возможность в разных компоннтах элементам давать одни и те же классы с разными стилями, не волнуясь, что они перекроют друг друга. Для этого файл со стилями называем таким образом:

Переименовываем Body.css в Body.module.css

Подключение файла со стилями import classes(любое название) from “./ Body.module.css ”

classes – это объект, ключ – значение, где ключ это моё название класса, а значение сгенерированное реактом, пример:

let classes = {

‘’item”:”Body\_\_item\_\_65759”

}

Но теперь название всех классов без исключения задаем так:

<**div className={classes.item}** >…..</div>

Но теперь мы не можем использовать в названии класса дефис, так он записывается как джаваскрипт. Поэтому ибо в названии класса не использовать девис, либо записывать так

<**div className={class[‘item-logo’]}** >…..</div>

Если нужно в элемент записать два и более класса:

<**p className=**{**`**${classes.item} ${classes.active}**`**}>А это классный Боди</**p**>

Пример:

Компонент Body

**import** classes **from './Body.module.css'  
const** *Body*=()=>  
{  
 **return**(  
 <**div className=**{classes.body}>  
 <**ul className="navbar"**>  
 <**li className=**{**`**${classes.item} ${classes.active}**`**}>Пункт 1</**li**>  
 <**li className=**{classes.item}>Пункт 2</**li**>  
 </**ul**>  
  
 </**div**>  
 )  
};  
  
**export default** *Body*;

Будет сгенерирован класс

<li class="Body\_item\_\_2kh\_N">Пункт 2</li>

А в компоненте Footer

<li class="Footer\_item\_\_3wVE4">Пукт 1</li>

В прошлый раз я не понял как это делать поэтому подключал с сервера

подключаю с сервера вот так:

<**script src="https://unpkg.com/react@16/umd/react.development.js"**></**script**>  
<**script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"**></**script**>  
  
*<!-- Don't use this in production: -->*<**script src="https://unpkg.com/babel-standalone@6.15.0/babel.min.js"**></**script**>

props

Любая компонента это функция, которая может иметь какой-либо параметр. Этот параметр и назыается props, который мы можем в дальней передавать при вызове компонента. Происходит это как с обычной функцией. Пример, у нас есть компонент Post

**const** *Post*=(props)=>  
{  
  
 **return**(  
 <**div className=**{classes.post\_item}>  
 <**div className=**{classes.post\_item\_\_text}>  
 <**p**>{props.**message**}</**p**>  
 </**div**>  
 <**div className=**{classes.post\_item\_\_likes}>  
 <**p**>{props.like} лайков</**p**>  
 </**div**>  
 </**div**>  
 )  
};

Когда мы вызываем компонент(а это ни что инное как функция) мы через атрибуты передаём значения параметров

**const** *MyPosts*=()=>  
{  
 **return**(  
 <**div className=**{classes.wrapper\_posts}>  
 <**Post like=**{**"15"**} **message="Привет"** />  
 </**div**>  
 )  
};

Итог:

Привет

15 лайков

**Переходы между страницами**

Для дого чтобы работали ссылки и мы могла перемещатся с одной страницы на другую в рамках проекта устанавливается модуль роутинга, котрый не установлен по умолчанию.

Npm install react-router-dom -save

-save чтобы запись об установлнном модуле добавилась в package.json

Далее в ссылках на станицу мы прописываем путь, он выглядит так

**const** *NavBar*=()=>  
{  
 **return**(  
 <**div className=**{classes.wrapper\_nav\_bar}>  
 <**ul className=**{classes.nav\_bar\_list}>  
 <**li**><**a href="/posts"**>MyPosts</**a**></**li**>  
 <**li**><**a href="/dialogs"**>Dialog</**a**></**li**>  
 </**ul**>  
 </**div**>  
 )  
};

**href="/dialogs"** – своего рода это якоря на компоненты, которые в дальнейшем при клике на ссылку где бы ссылка не находилась, браузером будут искаться

Далее оборрачиваем самый верхний компонент, в моём случае это App.js в новый компонент

<**BrowserRouter**>. Другие компоненты не нужно будет оборачивать этот компонент. Перед этим подключив его.

**import** {BrowserRouter} **from "react-router-dom"**;

Теперь переходим в компонент с компонентами, которые будут переключаться при нажатии на ссылку. Эти компоненты записываем по новому.

Как было, когда я только начал изучать:

**const** *Profile*=(props)=>  
{  
 **return** (  
 <**div className=**{classes.body}>  
 <**MyPosts**/> */\*компонент с постами\*/* <**Dialogs**/> */\*компонент с диалогами\*/* </**div**>  
  
 )  
};  
**export default** *Profile*;

И как стало:

**const** *Profile* = (props) => {  
 **return** (  
 <**div className=**{classes.body}>  
 <**Route path='/posts' component=**{*MyPosts*}/>  
 <**Route path='/dialogs' component=**{*Dialogs*}/>  
 </**div**>  
 )  
};  
**export default** *Profile*;

Теперь, если в какой либо части страницы у нас есть ссылка на загрузку компонента например диалог, достаочно в тэге a прописать

<**a href="/dialogs"**>Dialog</**a**></**li**>

Но так как на нужный контент мы переходим по ссылке у нас проичсходит перезагрузка страницы, а это не должно быть. По этому ссылки

<**li**><**a href='/posts'**>MyPosts</**a**></**li**>  
<**li**><**a href='/dialogs'**>MyPosts</**a**></**li**>

Мы записываем так:

<**li**><**NavLink to="/posts"**>MyPosts</**NavLink**></**li**>  
<**li**><**NavLink to="/dialogs"**>Dialog</**NavLink**></**li**>

Перед этим мы подключаем компонент *NavLink*

**import** {*NavLink*} **from 'react-router-dom'**

При выборе той или инной ссылке обернутой в компонент NavLink ей автоматом ставится класс active, которой можно задать классы и где не нужно будет программно добавлять этот класс

Но активной ссылке можно задать и свой класс, который будет ставится автоматом, для этого:

Прописываем её в activeClassName Это нужно делать у каждой ссылке

<**NavLink activeClassName=**{**"active\_link"**} **to="/posts"**>MyPosts</**NavLink**>

<**NavLink activeClassName=**{**"active\_link"**} **to="/dialogs"**>Dialogs</**NavLink**>