Git commands:-

1. git status
2. git add React.docx
3. git commit -m "first\_react\_app creation with component creation"
4. git push

**React**

* React is a front-end library.
* It is developed by Facebook.
* Unlike Angular JS, which is MVC(Model View Controller), it focus on View part only but not on model and controller. For model and controller, we need Node JS , Spring Boot,etc.
* It creates re-useable components.
* React is all about component. You need to think everything as a component.
* The virtual DOM makes react fast.(DOM represents objects/elements/nodes as a tree structure).

**Pre-requisites:-**

* **Solid understandings of HTML, CSS, Java Script is required.**

**Features:-**

1. JSX:- JavaScript syntax extension.

**Limitations:-**

It covers only View layer. So, we need to choose another technology for complete development.

**NOTE: It runs on Node, So Node JS is needed to be installed.**

**To install React App globally:**

npm install –g create-react-app –save

Where npm = node package manager

Create-react-app will create your app with all the necessary files needed.

To start React app:

We use npm start

When we create React App, By default it is created with required packages under node\_modules

And following directories to start off with.

These directories namly:

1. node\_modules
2. public: It contains index.html
3. src: It contains components

When we do **npm start**-> it will search ‘start’ in package.json .

start comes under react-scripts package.

"dependencies": {

"react": "^16.8.6",

"react-dom": "^16.8.6",

"react-scripts": "3.0.1"

},

"scripts": {

"start": "react-scripts start",

"build": "react-scripts build",

"test": "react-scripts test",

"eject": "react-scripts eject"

},

As we can see start comes under react-scripts.

**How to make class/Component**

Components look like HTML but it is actually **JSX.**

import React, { Component } from 'react';

class App extends Component {

render() {

return (

<div>

Hello World, Hi suryasnata

</div>

)

}

}

export default App;

**Container**:

It is a Component that can contain one or multiple Components.

When you start the app, internally it will go to index.js

**index.js**

import React from 'react';

import ReactDOM from 'react-dom';

import App from './container/App';

import \* as serviceWorker from './serviceWorker';

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

serviceWorker.unregister();

index.js will get ‘root’ from index.html and will then put <App /> Component which is our container in it.

**index.html**

<div id="root">

**App.js**

import React, { Component } from 'react';

import WelcomeMessage from '../components/WelcomeMessage';

class App extends Component {

render() {

return (

<div>

Hello World, Hi suryasnata

<WelcomeMessage />

</div>

)

}

}

export default App;

Here, App Component(In our case it is container) is using WelcomeMessage Component.

**WelcomeMessage.js**

import React, {Component} from 'react';

class WelcomeMessage extends Component {

render(){

return(

<div>

Hello

</div>

)

}

}

export default WelcomeMessage

**Starting the new Application from the beginning:-**

Every component has its state. State is the dynamic data that we render it in DOM.

This is a keyword that is used to get data from the class. This represent the class.

import React, { Component } from 'react';

class WelcomeMessage extends Component {

state = {

name: "surya",

age: 30

}

render() {

return (

<div>

Hello

<p>

My name is {this.state.name} and I am {this.state.age}

</p>

</div>

)

}

}

export default WelcomeMessage

**DAY 2: Exercise**

import React, { Component } from 'react';

class WelcomeMessage extends Component {

state = {

name: "surya",

age: 30

}

handleClick(e) {

console.log(e.target);

}

handleMouseover(event) {

console.log(event.target, event.pageX, event.pageY);

}

copy = (event) => {

console.log('dnt copy me');

}

render() {

return (

<div>

Hello <p> My name is {this.state.name} and I am {this.state.age}</p>

<button onClick={this.handleClick}>CLick Me</button>

<button onMouseOver={this.handleMouseover}>Hover me</button>

<p onCopy={this.copy}>Copy me</p>

</div>

)

}

}

export default WelcomeMessage

onClick, onMouseOver, onCopy are the predefined events.

onClick :- this event will arise when we will click on the button or tag or HTML elements.

onMouseOver:- This event will arise when we take the mouse on the HTML elements.

onCopy:- This event will arise when we try to copy any elements.

If you want to use `this` keyword under a method then you have to bind it.

There are two ways to bind this keyword to a function.

One of the easiest way is using new javascript syntax.

handleClick = (e) => {

console.log(e.target);

}

#Excercise 2

import React, { Component } from 'react';

class WelcomeMessage extends Component {

state = {

name: "surya",

age: 30

}

handleClick = (e) => {

this.setState({

name: 'Tanuj'

})

console.log(this.state.name);

}

handleMouseover(event) {

console.log(event.target, event.pageX, event.pageY);

}

copy(event) {

console.log('dnt copy me');

}

render() {

return (

<div>

Hello <p> My name is {this.state.name} and I am {this.state.age}</p>

<button onClick={this.handleClick}>CLick Me</button>

<button onMouseOver={this.handleMouseover}>Hover me</button>

<p onCopy={this.copy} > Copy me </p>

</div>

)

}

}

export default WelcomeMessage

If we want to change the state we use:-

this.setState({

name: 'Tanuj'

})

We don’t directly mutate state, because it can lead to unforeseen errors in our application.

this.state.name='Tanuj'

**Form creation:-**

The below is the example of form creation.

import React, { Component } from 'react';

class WelcomeMessage extends Component {

state = {

name: '',

age: 30

}

handleSubmit = (e) => {

e.preventDefault();

this.setState({

name:''

})

console.log("form Submitted")

}

handleChange = (e) => {

this.setState({

name: e.target.value

})

}

render() {

return (

<div>

<h1>My name is {this.state.name}</h1>

<form onSubmit={this.handleSubmit}>

<input type='text' onChange={this.handleChange} />

<button>Submit</button>

</form>

</div>

)

}

}

export default WelcomeMessage

**event.preventDefault()** used to prevent the default value.

handleSubmit = (e) => {

e.preventDefault();

this.setState({

name:''

})

onChange:- It is mainly used in :-

<input type='text' onChange={this.handleChange} />

<textarea type='text' onChange={this.handleChange} />

**:DAY 3:**

**#Lecture 1**

**props:**

When we want to retrive data from a parent component we can use props in other words, when we pass the data from parent component to any of the child components then props is being used.

Syntax:

**{this.props.<data>}**

**Below is the example:**

Here, **<App />** is the root Component and **<Ninjas />** is the child Component that is been used inside the **<App />** Component.

***App.js***

import React, { Component } from 'react';

import WelcomeMessage from '../components/WelcomeMessage';

import Ninjas from '../components/Ninjas';

class App extends Component {

render() {

return (

<div>

**<Ninjas name='Tanuj' age='30' />**

</div>

)

}

}

export default App;

***Ninjas.js***

import React, { Component } from 'react';

class Ninjas extends Component {

render() {

return (

<div>

<div>name: {this.props.name} </div>

<div>age: {this.props.age}</div>

</div>

)

}

}

export default Ninjas

Another way to get values from props:-

import React, { Component } from 'react';

class Ninjas extends Component {

render() {

const { name, age } = this.props;

return (

<div>

<div>name: {name} </div>

<div>age: {age}</div>

</div>

)

}

}

export default Ninjas

Iteration of Lists(1st way):

It should be in App.js

state = {

list: [

{ name: 'Tanuj', id: 1 },

{ name: 'Guddunata', id: 2 },

{ name: 'BunuliNayak', id: 3 }

]

}

import React, { Component } from 'react';

class Ninjas extends Component {

render() {

return (

<div>

{

this.props.listOfNinjas.map(ninja => {

return (

<div key={ninja.id} >NAME: {ninja.name}

AGE: {ninja.age}

</div>

)

})

}

</div>

)

}

}

export default Ninjas

Iteration of Lists(2nd way):

import React, { Component } from 'react';

class Ninjas extends Component {

render() {

const ninjas = this.props.listOfNinjas;

return (

<div>

{

ninjas.map(ninja => {

return (

<div key={ninja.id}>

name: {ninja.name}

&nbsp;

age: {ninja.age}

</div>

)

})

}

</div>

)

}

}

export default Ninjas

**DAY4:**

**Revision**

**:DAY5:**

Container Component:

1. Contain state.
2. Contain lifecycle hooks.
3. Not concerned with UI.
4. Uses class to create.

UI Component:

1. Doesnot contain state.
2. Receive data from props.
3. Only concerned with UI.
4. Use functions to create.

**Creation of Function:-**

import React from 'react';

const Prop = (props) => {

const { list } = props;

const listOdProps = list.map(listOfitem => {

return (

<div className='listOfitem' key={listOfitem.id}>

NAME: {listOfitem.name}

</div>

)

})

return (

<div className="Nija-list">

{listOdProps}

</div>

)

}

export default Prop;

**Usage of If condition:**

import React from 'react';

const Prop = (props) => {

const { list } = props;

const listOdProps = list.map(listOfitem => {

if (listOfitem.age > 25){

return (

<div className='listOfitem' key={listOfitem.id}>

NAME: {listOfitem.name}

AGE: {listOfitem.age}

</div>

)

} else {

return null;

}

})

return (

<div className="Nija-list">

{listOdProps}

</div>

)

}

export default Prop;

**Usage of ternary operator:**

import React from 'react';

const Prop = (props) => {

const { list } = props;

const listOdProps = list.map(listOfitem => {

// if (listOfitem.age > 25) {

return listOfitem.age > 25 ? (

<div className='listOfitem' key={listOfitem.id}>

NAME: {listOfitem.name}

AGE: {listOfitem.age}

</div>

) : null;

});

return (

<div className="Nija-list">

{listOdProps}

</div>

)

}

export default Prop;

OR

import React from 'react';

const Prop = (props) => {

const { list } = props;

return (

<div className="Nija-list">

{

list.map(listOfitem => {

return listOfitem.age > 25 ? (

<div className='listOfitem' key={listOfitem.id}>

NAME: {listOfitem.name}

AGE: {listOfitem.age}

</div>

) : null;

})

}

</div>

)

}

export default Prop;

**:DAY6:**

Adding data through form:

import React, { Component } from 'react'

class AddNinjas extends Component {

state = {

name: null,

age: null

}

handleChange = (e) => {

this.setState({

[e.target.id]: e.target.value

})

}

handleSubmit = (e) => {

e.preventDefault();

this.props.addNinja(this.state)

}

render() {

return (

<div>

<form onSubmit={this.handleSubmit}>

<label htmlFor="Name">name:</label>

<input type="text" id="Name" onChange={this.handleChange} />

<label htmlFor="Name">age:</label>

<input type="text" id="age" onChange={this.handleChange} />

<button>Submit</button>

</form>

</div>

)

}

}

export default AddNinjas;

Here we made a program in which we have made 2 methods.

1st method:

handleChange = (e) => {

this.setState({

[e.target.id]: e.target.value

})

}

2nd method:

handleSubmit = (e) => {

e.preventDefault();

this.props.addNinja(this.state)

}

We have made a form.In which we are passing the handle submit method.

In label input type we are adding the handleChange method.

<form onSubmit={this.handleSubmit}>

<label htmlFor="Name">name:</label>

<input type="text" id="Name" onChange={this.handleChange} />

<label htmlFor="Name">age:</label>

<input type="text" id="age" onChange={this.handleChange} />

<button>Submit</button>

</form>

This is a Container Component(App.js). In our case it is using AddNinjas Component which will allow user to add Ninja i.e, its name and age. When the user who wants to be Ninja fills the form and submit it, this in turn call handleSubmit function of AddNinjas Component as below:-

handleChange = (e) => {

this.setState({

[e.target.id]: e.target.value

})

}

handleSubmit = (e) => {

e.preventDefault();

this.props.addNinja(this.state)

}

This handleSubmit function will in turn call the addNinja function of Parent Component which is App component in our case. This addNinja function expects a parameter so we are passing our state which is an object that contains name and age { name:null, age:null }.

As we are calling **this.props.addNinja(this.state)** from AddNinjas Component, it will go back to addNinjas function of App Component.

import React, { Component } from 'react';

import Welcome from '../src/Welcome';

import Form from '../src/Form';

import Prop from '../src/Prop'

import AddNinjas from '../src/AddNinjas'

class App extends Component {

state = {

ninjas: [

{ name: 'Tanuj', age: 35, id: 1 },

{ name: 'Guddunata', age: 29, id: 2 },

{ name: 'BunuliNayak', age: 20, id: 3 }

]

}

**addNanja = (ninja) => {**

**ninja.id = Math.random();**

**let ninjas = [...this.state.ninjas, ninja];**

**this.setState({**

**ninjas: ninjas**

**})**

**}**

deleteNinja = (id) => {

let ninjas = this.state.ninjas.filter(ninja => {

return ninja.id !== id

});

this.setState({

ninjas: ninjas

})

}

render() {

return (

<div>

<i>Hello World</i>

<Welcome />

<Form />

<Prop deleteNinja={this.deleteNinja} ninjas={this.state.ninjas} />

**<AddNinjas addNinja={this.addNanja} />**

</div>

)

}

}

export default App;

**Deleting the data:-**

This is a Container Component(App.js). In our case it is using DeleteNinjas Component which will allow user to delete Ninja i.e, its name and age. When the user who wants to delete Ninja click the delete button which we have made in Prop.js, this in turn call deleteNinja function of App.js Component.

**<button onClick={() => { deleteNinja(ninjas.id) }}>Delete</button>**

**App.js**

import React, { Component } from 'react';

import Welcome from '../src/Welcome';

import Form from '../src/Form';

import Prop from '../src/Prop'

import AddNinjas from '../src/AddNinjas'

class App extends Component {

state = {

ninjas: [

{ name: 'Tanuj', age: 35, id: 1 },

{ name: 'Guddunata', age: 29, id: 2 },

{ name: 'BunuliNayak', age: 20, id: 3 }

]

}

addNanja = (ninja) => {

ninja.id = Math.random();

let ninjas = [...this.state.ninjas, ninja];

this.setState({

ninjas: ninjas

})

}

**deleteNinja = (id) => {**

**let ninjas = this.state.ninjas.filter(ninja => {**

**return ninja.id !== id**

**});**

this.setState({

ninjas: ninjas

})

}

render() {

return (

<div>

<i>Hello World</i>

<Welcome />

<Form />

**<Prop deleteNinja={this.deleteNinja} ninjas={this.state.ninjas} />**

<AddNinjas addNinja={this.addNanja} />

</div>

)

}

}

export default App;

**Prop.js:-**

import React from 'react';

const Prop = (props) => {

const { ninjas**, deleteNinja } = props;**

return (

<div className="Nija-list">

{

ninjas.map(ninjas => {

return ninjas.age > 25 ? (

<div className='ninjas' key={ninjas.id}>

NAME: {ninjas.name}

AGE: {ninjas.age}

**<button onClick={() => { deleteNinja(ninjas.id) }}>Delete</button>**

</div>

) : null;

})

}

</div>

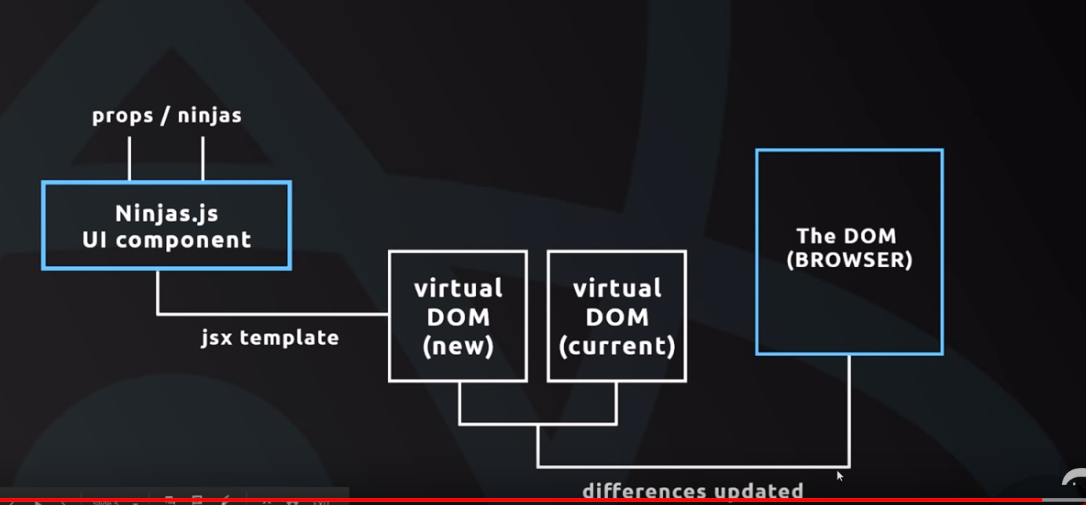
)

}

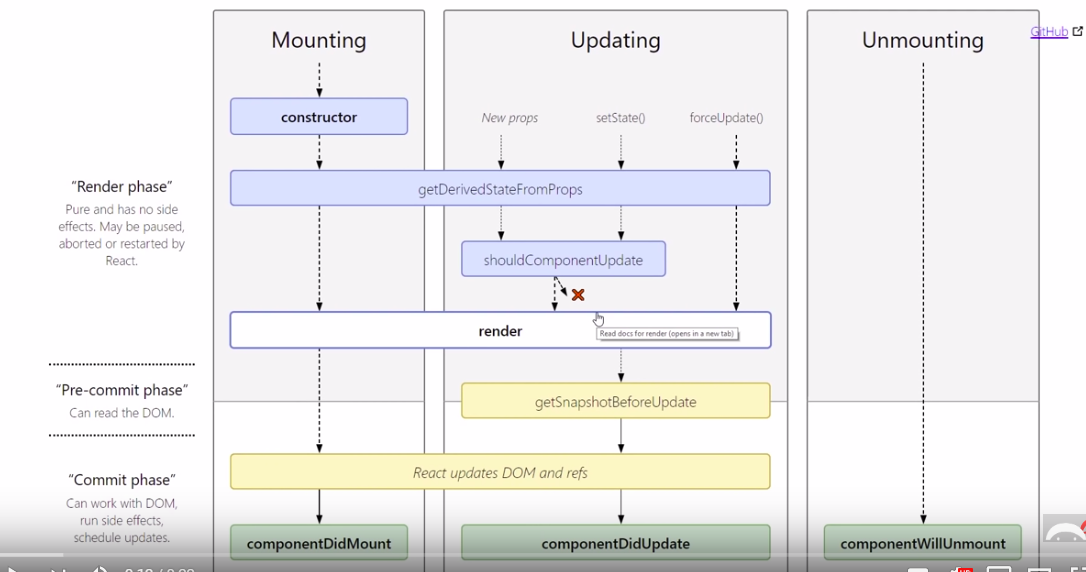
export default Prop;

**:DAY7:**

The React Process:-



Process of React:-



import React, { Component } from 'react';

import Todos from '../src/Todos'

class App extends Component {

state = {

todos: [

{ id: 1, content: 'Play Mario' },

{ id: 2, content: 'study React' }

]

}

deleteTodo = (id) => {

const todos = this.state.todos.filter(todo => {

return todo.id !== id

});

this.setState({

todos

})

}

render() {

return (

<div className="todo-app container">

<h1 className="center blue-text">Todo's </h1>

<Todos todos={this.state.todos} deleteTodo={this.deleteTodo} />

</div>

);

}

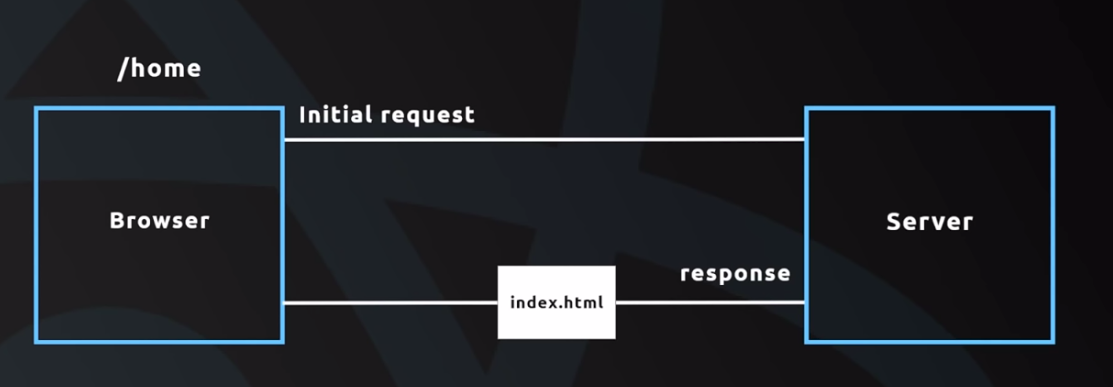
}

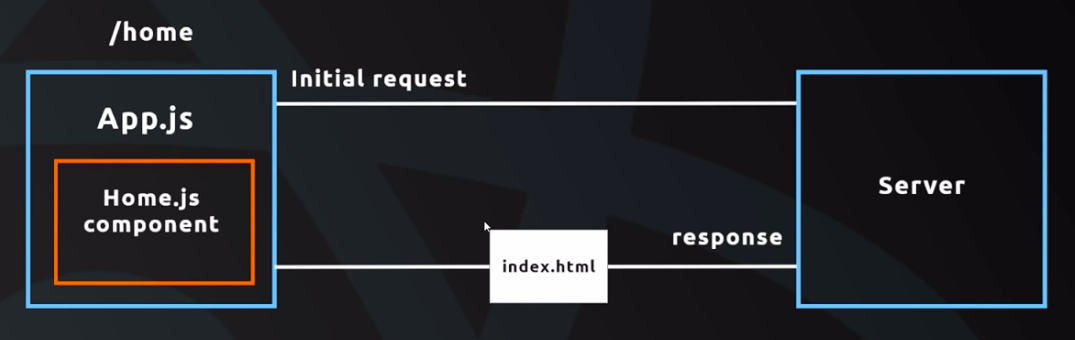
export default App;

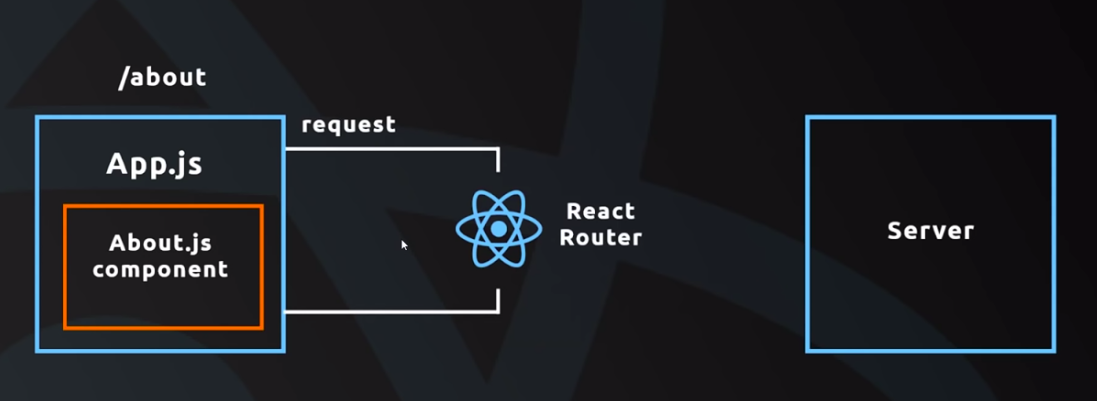
**React Router:**

React Router helps to route from one page to another page.

To get the functionality we need to install react-router-dom.







So As per our requirement,

1. There should be home page,About page,contact page.
2. We can navigate to any page from any page.

First of all we need to build home.js

import React from 'react';

const home = () => {

return (

<div className="conatiner">

<h4 className="center">Home</h4>

<p>Welcome to my home page</p>

</div>

)

}

export default home

Then About.js

import React from 'react';

const About = () => {

return (

<div className="conatiner">

<h4 className="center">About</h4>

<p>Welcome to my About page</p>

</div>

)

}

export default About

Then Contact.js

import React from 'react';

const Contact = () => {

return (

<div className="conatiner">

<h4 className="center">Contact</h4>

<p>Welcome to my Contact page</p>

</div>

)

}

export default Contact

To make all page navigatable, we need to make Navbar.js

import React from 'react'

const Navbar = () => {

return (

<nav className="nav-wrapper red darken-3">

<div className="conatiner">

<a href="/" className="brand-logo">Poke'Times</a>

<ul className="right">

<li><a href="/home">Home</a></li>

<li><a href="/about">About</a></li>

<li><a href="/contact">Contact</a></li>

</ul>

</div>

</nav>

)

}

export default Navbar

We can call all the above js in the conatainer js i.e. App.js

import React, { Component } from 'react';

import Navbar from '../src/components/Navbar'

import { BrowserRouter , Route } from 'react-router-dom'

import home from '../src/components/home'

import About from '../src/components/About'

import Contact from '../src/components/Contact'

class App extends Component {

render() {

return (

<BrowserRouter>

<div className="App">

<Navbar />

<Route path='/home' component={home} />

<Route path='/about' component={About} />

<Route path='/contact' component={Contact} />

</div>

</BrowserRouter>

);

}

}

export default App;

**Links & navLinks:**

Links and NavLinks worked the same as <a href>

Navbar.js

import React from 'react'

import { Link, NavLink } from 'react-router-dom'

const Navbar = () => {

return (

<nav className="nav-wrapper red darken-3">

<div className="conatiner">

<a href="/" className="brand-logo">Poke'Times</a>

<ul className="right">

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

<li><NavLink to ="/about">About</NavLink></li>

<li><NavLink to ="/contact">Contact</NavLink></li>

</ul>

</div>

</nav >

)

}

export default Navbar