React Notes

let – cannot be use in global block

const –cannot be edited

var – global variables

INSTALL NPM

Install react use “npm install –g create-react-app”

Creating Project = npx create-react-app *project-name*

To run program = npm start

Introduction

Function

function fruitColor(fruit){

switch(fruit){

case:”Apple”:{

var color = “red”

return color}

case:”Banana”:{

var color = “yellow”

return color}

default:

return “Other Color”

}

}

Console.log(fruitColor(“Apple”))

Console.log(fruitColor(“Banana”))

Arrow Function

function sayMyName1(name){

return ‘My name is ${name}’

}

console.log(sayMyName(“Peter”))

function sayMyName2=(name)=>{

return ‘My name is ${name}’

}

console.log(sayMyName(“John”))

function sayMyName3=(name)=> ‘My name is ${name}’

console.log(sayMyName(“Mike”))

Classes

Class Animal {

eat =() => console.log(“I am Eating Now”)

}

Class Dog extends Animal {

weight = 20;

age =8;

bark = () =>console.log(“Bark Bark”)

}

puppy = new Dog()

puppy.bark()

puppy.eat()

Spread Operator

/…

Arry1 = [2,1,2312,4]

Arry2 = […Arry1]

Console.log(Arry2) Array can be added to anotherarray series

const showValues = (…args)=>{

for (i=0, i< args.length,i++){

console.log(args[i])

}

}

showValues(“Peter”,”Sample”)

**Destructuring**

const person={

firstName:”Allan”,

lastName:”Lamadrid”,

age:50,

eyeColor:”Blue”

}

const {firstName, age } = person

console.log(firstname)

console.log(age)

if we use normal method we use person.age

**Array Function**

var numbers = [1,2,3]

const plus3 = numbers.map(num=>num+3)

console.log(plus3)

const plus4function = num => num +4

const = plus4 numbers.map(plus4function)

console.log(plus4)

FOR LOOP IN AND OF

conts fruitArray = [“Apple”,”Orange”,”Banana”]

for (let value of fruitArray ){

console.log(value) it prints it one by one

}

For a Object use FOR IN

const fruitsObject = {

apple:3,

mango:4,

banana:5

}

for (let key of fruitsObject){

console.log(fruitsObject[key]) it prints it one by one

}

Index.js

Import React from ‘react’;

Import ReactDom form ‘react-dom’;

const App = () =>{

return(

<div>HI <div>

)

}

ReactDom.render(

<App/>,

Document.querySelection(#root)

)

Function Base

Class Base

React Component

Can be functional or Class

Jxs – Html like

Event Handler – Event form user

JXS

Styling inline JSX <div style= {{ backgroundColor : red}}> </div> or

Crete variable that contain style

const style = { backgroundColor : red, color:white } then bind this

Variable const buttonClick = “ Click Me” use this as { buttonClick } example

<button style= {{ backgroundColor : red}}> *{ buttonClick }* </button>

For Function

function formatName(user){

return user.firstName + ‘ ’ + user.lastName;

}

const user = {

firstName:’Kevin’,

lastName:’Lamadrid’

}

console.log(formatName(user))

**Note: Embedding using function with object**

function getClick (){

return ‘hi’

}

<button style= {{ backgroundColor : red}}> *{* getClick ()*}* </button>

**Note: Embedding using function**

App.js Example for using Dict for Style

Import React from ‘react’;

Import ReactDOM form ‘react-dom’;

const App() => {

const style = {backgroundColor:’red’, color:’white’}

return (

<div>

<label for =”name” className=”label”>Enter E-mail <label>

<input id=”name” type=”text”/>

<button style={style}></button>

</div>

)

}

ReactDOM.render(

<App/>

document.querySelector(‘#root’)

)

Props - is system for passing data from a parent component to child component

Pros

Easy and fast to separate UI elements

Easy to inject props down to children based on the parent component's state

Cons

Less visibility into the composition architecture

Less reusability

Good if

B and C are just presentational components

B should be responsible for C's lifecycle

Components

class Car extends React.Component {

constructor() {

super();

this.state = {color: "red"};

}

render() {

return <h2>I am a Car!</h2>;

}}

Components with state

class Car extends React.Component {

constructor() {

super();

this.state = {color: "red"};

}

render() {

return <h2>I am a {this.state.color} Car!</h2>;

}}

Component with props

class Car extends React.Component {

render() {

return <h2>I am a {this.props.color} Car!</h2>;

}}

ReactDOM.render(<Car color="red"/>, document.getElementById('root'));

2 Components with Props

class Car extends React.Component {

render() {

return <h2>I am a {this.props.brand}!</h2>;

}

}

class Garage extends React.Component {

render() {

return (

<div>

<h1>Who lives in my garage?</h1>

<Car brand="Ford" />

</div>

);

}

}

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

Components Props

class Car extends React.Component {

render() {

return <h2>I am a {this.props.brand}!</h2>;

}}

class Garage extends React.Component {

render() {

const carname = "Ford";

return (

<div>

<h1>Who lives in my garage?</h1>

<Car brand={carname} />

</div>

);

}}

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

class Car extends React.Component {

render(){

return <h1>Hi i am a {this.props.brand.model}</h2>

}

}

class Garage extends React.Component {

const cars = {

name: "Ford",

model: "Mustang"

}

render(){

return

<div>

<h1>This is a new Component</h1>

<Car brand={cars}> // is use cars component to with another component

</div>

;

}

}

Change Properties of state (setState)

class Car extends React.Component{

constructor(props){

super(props);

}

this.state = {

brand: "Ford",

model: "Mustang",

color: "Green",

year: "1964",

}; // This is for state

changeColor = () = {

this.setState(color : "blue")

}

render(){

return(

<div>

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

<p>

It is a {this.state.color} with a model of {this.state.model} from {this.state.brand}

</p>

<button type="button" onClick = {this.changeColor}>

</div>Change Color</button>

);

}

}

}

}