Cheat Sheet: Developing Front-End Apps with React

Estimated reading time: 19 minutes

Package/Method	Description	Code Example

Arrow function	Arrow functions allow you to write shorter function syntax.	`hello = () =>`
		`{`
		`return "Hello World!";`
		`}`

class	Class is a template or blueprint for creating an object.	`function car(name,year)`
		`{`
		`this.name = name`
		`this.year = year`
		`return this;`
		`}`
		`let car = car("Ford", 2014)`
		`console.log(car)`
		`console.log(car.name)`
		`console.log(car.year)`

Hooks	Functions called hooks enable "hooking into" features of the React state and lifecycle from function components.	`importReact,{useState}from 'react';`
		`function CntApp() {`
		`const[count,setCount]=useState(0);`
		`return(Youclicked{count}many times`
		`<buttononClick={()=>setCount(count+1)}>Clickme</button>`
		`);}`
		`export default CntApp;`

Inheritance	A class created with a class inheritance inherits all the methods from another class.	`class Square extends Rectangle`
		`{`
		`constructor(height,width)`
		`{`
		`if(height === width)`
		`{`
		`super(height,width)`
		`}`
		`else`
		`{`
		`super(width,width)`
		`}`
		`}`
		`}`
		`let mySquare = new Square(5,5)`

let and const	let allows you to restrict the scope of variables within the block where they are declared. const allows you to declare constants whose values cannot be changed.	`{`
		`let a = 10`
		`console.log(a)`
		`a = 15`
		`console.log(a)`
		`}`
		`console.log(a)`
		`const num = 5`
		`console.log(num)`
		`num = 8`
		`console.log(num)`

Mounting	When a component instance is created and added to the DOM, these methods are invoked in the following order: constructor(), getDerivedStateFromProps(), render(), componentDidMount().	`class Header extends React.Component {`
		`constructor(props) {`
		`super(props);`
		`console.log("Inside the constructor")`
		`}`
		`componentDidMount =()=> {`
		`console.log("Inside component did mount")`
		`}`
		`render() {`
		`console.log("Inside render method")`
		`return (`
		`The component is rendered`
		`)`
		`}`
		`}`
		`export default App`

onClick	When an event fires, event handlers decide what should happen next. This could involve pressing a button or altering a text entry.	`function changeColor() {`
		`const shoot = () => {`
		`alert("Color Changed!");`
		`}`
		`return (`
		`<button onClick={change}>Change the Color! </button>`
		`);`
		`}`
		`const root =`
		`ReactDOM.createRoot(document.getElementById('root'));`
		`root.render(<changeColor />);`

Promises	The Promise object represents the eventual completion (or failure) of an asynchronous operation and its resulting value.	`let promiseArgument = (resolve, reject) =>`
		`setTimeout (() => {`
		`let currTime = new Date().getTime();`
		`if(currTime % 2 === 0){`
		`resolve("Success")`
		`}else{`
		`reject("Failed!!!")}`

Package/Method	Description	Code Example
		},2000) let myPromise = new Promise(promiseArgument);

Props	Props is short for properties, and it is used to pass data between React components.	class TestComponentextendsReact.component{
		render() {
		return
		Hi {this.props.name}}}
		//passing the props as examples to the test component
		TestComponentname='John'
		TestComponentname='Jill'

React class Component	React class component contains- Props: set from outside the class, State: internal to the class	import React from "react";
		class App extends React.Component {
		constructor(props) {
		super(props);
		this.state={change: true };
		}
		render() {
		return(
		<button Click={()=>{this.setState({change:
		!this.state.change});}}>Click Here!</button>
		{this.state.change?(Hello!!):(Welcome to the React Course)}
		);}}
		export default App;

React components	Components are reusable segments of code that come under the class and functional component types.	import React from 'react';
		import {Text} from 'react-native';
		const Helloworld= ()=>
		{
		return
		(Hello, World!);
		}
		export default Helloworld;

React Forms	React makes use of forms to enable user interaction with the website.	import React, {Component} from "react";
		export default functionApp() {
		const [email, setEmail] = React.useState("");
		const [password, setPassword] = React.useState("");
		const handleSubmit = (event) => {
		console.log(`Email: ${email}Password: ${password}`);
		event.preventDefault();
		}
		return (< formonSubmit = {
		handleSubmit
		} >
		< h1 > Registration < /h1>
		<label> Email:< input name="email" type="email" value={email}
		onChange={e => setEmail(e.target.value)} required/ > < /label>
		<label>Password:<input name="password" type="password" value={password}
		onChange={e => setPassword(e.target.value)}required/></label>
		<button>Submit</button>
		</form>);}

React state	The state object is where you keep the component's property values.	class TestComponentextendsReact.component{
		constructor() {
		this.state= {
		id: 1,
		name: “John”
		age: 28
		};}
		render() {
		return ({this.state.name}{this.state.age}
		)}}

Redux	Redux is a state management library that is often used with React to handle the state of your application. An application state is like a global object that holds information that you use for various purposes later in the app.	$ npm install redux react-redux --save

Unmounting	When a component is removed or unmounted from the DOM, the componentWillUnmount() method enables us to run React code.	import React from 'react';
		class ComponentOne extends React.Component {
		componentWillUnmount() {
		console.log('The component is going to be unmounted');
		}
		render() {
		return Inner component;
		}
		}
		class App extends React.Component {
		state = { innerComponent:<AppInner/>};
		componentDidMount() {
		setTimerout(()=>{
		this.setState({ innerComponent:unmounted});
		},5000)
		}
		render() {
		console.log("Inside render")
		return (
		{this.state.innerComponent});
		}
		}
		export default App;

Package/Method	Description	Code Example

Updating	When a component is updated, five methods are called in the same order: getDerivedStateFromProps(), shouldComponentUpdate(), render(), getSnapshotBeforeUpdate(), componentDidUpdate()	class App extends React.Component{
		state = {counter: "0"};
		incrementCounter = () =>
		this.setState({counter:parseInt(this.state.counter)+1})
		shouldComponentUpdate(){
		console.log("Inside shouldComponent update")
		return true;
		}
		getSnapshotBeforeUpdate(prevProps,prevState){
		console.log("Inside getSnapshotBeforeUpdate")
		console.log("Prev counter is" +prevState.counter);
		console.log("New counter is" +this.state.counter);
		return prevState
		}
		componentDidUpdate(){
		console.log("Inside componentDidUpdate")
		}
		render(){
		console.log("Inside render")
		return(<button onClick={this.incrementCounter}>Click
		Me!</button>{this.state.counter}
		)
		}}
		export default App;

Skills Network