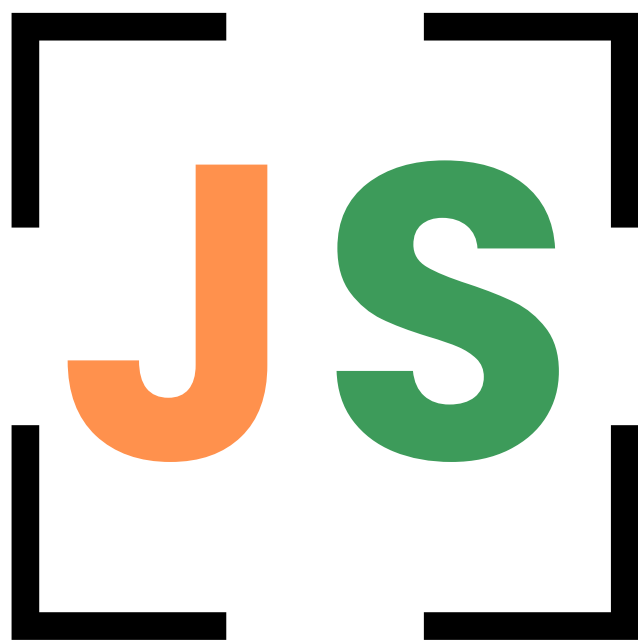
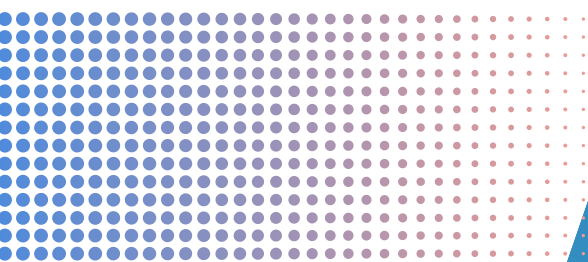


React



Prepared By :

opscodify



www.blacksnwhite.com

What is React JS ?

React JS is an open source java script library used for building interfaces. It focuses on creating dynamic interactive web applications by allowing developers to create reusable UI components.

Why we React JS?

React is popular and widely used for several reasons:

- **Component based architecture:**

React allows you to break down your UI into reuse components. Making it easier to manage.

- **Efficient updates:**

It used virtual DOM, which optimized rendering by updating only parts of pages that change improving performance.



REACT JS



REACT NATIVE

Storage: React JS is a good choice for projects that requires high performance.

Storage: It is a good choice for projects that need to be able scale easy.

Search engine friendly: It is more search engine friendly than react native.

Search engine friendly: It can't be made search engine friendly.

Navigation: React JS uses traditional browser based approach.

Navigation: React Native relies on native platform.

Platform: React is a framework for building application using java script.

Platform: It allows building native and cross platform mobile apps.

Rendering: Browser code is rendered through virtual DOM.

Rendering: It use native API to render all components.

Syntax: Makes use of HTML and its syntax flow.

Syntax: React Native uses react native syntax.

Installation Process: React library is installed via npm package manager.

Installation Process: ReactNative is a command line interface tool requires both node js and reactnative CCI to be installed.

Efficiency: React JS is more efficient in terms of code reuse ability.

Efficiency: React Native is more efficient in terms of performance and memory usage.

Technology Base: React JS is JavaScript library used for building user interfaces.

Technology Base: React Native is a cross platform mobile development.



Components: React JS components are typically written in HTML.

Components: React Native components are written in JSX

KEY DIFFERENCE BETWEEN REACT AND VUE

The main difference between vue and react is how they approach application design.

While react focuses on creating reusable UI components vue takes approach by providing developers with frontend tools.

 VUE	 REACT
It was single file components (SFC) to build different components.	It uses JSX as a component format.
It is used to develop web-based application.	It is used to develop web as well as mobile application.
State management library is called vux	State management library is called Redux.
The performance is slow as react.	The performance is slow when compared to vue.
It is not suitable for long term support.	It is suitable for long term support.

REACT JSX

What is JSX ?

JSX stand for Java script XML. JSX allows us to write HTML in react. JSX makes it easier to write and add HTML in react.

Coding JSX

JSX allows us to write HTML element in java script and place them in DDM without creating createElement or append child () JSX convert HTML tags into react elements. You are not required to use JSX, but JSX makes it easier to write react application.

Exmple 1 :-

```
Const my element = <h1> I love JSX!</h1> ;  
Const root = reactDDM.createroot (document.getElementById  
(‘root’));  
Root.render (myelement);
```

Out-Put :-

I love JSX!

Exmple 2:**Withour JSX:**

```
ConstmyElement = react.createElement ('hi',{ } , 'I do not');
Const root = ReactDOM.creatroot (document.getElementById
('root'));
Root.reander (myElement);
```

Out-Put :-

I do not

In example 1 JSX allows you to write HTML directly within Java script. JSX is an extension of Java script language based on ES6.

EXPRESSION IN JSX

When JSX you can write expression inside curly braces { } The expression can be react variable or property. JSX will execute expression and return the result.

Exmple :

execute the expression 5 + 5

```
ConstmyElement = <h1> React is <5 + 5> times better with JS
</h1>;
```

Out-Put :-

React is 10 times better with JSX

Inserting a large block of HTML

To write HTML no multiple lines, put HTML inside parenthesis.

Exmple :

```
constmyelem = (
<ul>
<li> Apple </li>
<li>Banana</li>
<li>Cherries</li>
</ul>
);
```

Out-Put :-

- Apple
- Banana
- Cherries

Exmple :

```
constmyElement = (
<div> = ( <p> I am
Paragraph </p>
<p> Paragraph too </p>
</div>
);
```

JSX will throw error if HTML is not correct or if HTML misses a parent elemnt here <div> is parent <p> is child.

Out-Put :-

I am Paragraph
Paragraph too

Elements must be closed in close empty elements with / }

SETTING UP A REACT ENVIRONMENT

If you have npx and node.js installed, you can create a react application by using craet-react-app.

The create-react-app will set up everything you need to react application to run.

Run the react application

Cd my-react-app

Run this command to run react application npm start.

A new browser will pop up with your newly created react app!

If not open browser and type localhost:3000



REACT COMPONENTS

When creating a react component, the component name must start with uppercase letter.

Class component must include extends react component statement.

The component also requires render () this method return HTML.

Exmple :

```
class car extends react.component {  
  render ( ) {  
    return <h2> Hi, I am a Car! </h2> }  
  }  
}
```

Function Component:

Same as react component, only difference is written using less code.

```
Function ( ) {  
  return <h2> Hi, I am a Car! </h2>;  
}
```

Props :

Components can be passed as props, which stands for properties.

Component in Files:

React is all about re-using code and it is recommended to split year componts into separate files. To do that create a new file with .js files

React class component state:

React class component have built in state object

Creating the state object:

```
Class car external
react.component {
  Constructor (props) {
    Super (props );
    This.state = {brand:"Ford"};
  } render ( ){
    return (
      <div>
        <h1>My Car</h1>
      </div> );
    } }
```

Using the state object:

React to the state object anywhere in the component using `this.state.propertyname` syntax.

Example:

```
Class car external
react.component {
  Constructor (props) {
    Super (props);
    this.state = {
      brand:"Ford"
      model : "Mustang"
      color: "red"
      year : 1964 };
  }
  render ( )
  {
    return ( <div>
      <h1> My <this.state.brnd>
    </h1>
    <p> It is a <this.state.color>
    <this.state.model>
    From<this.state.year> </p>
    </div> );
  } }
```

Example:

MY FORD
It is a red Mustang From 1960.

Changing the state Object :

To change a value in state object, use the `this.setState()`

Example :

```
Class car extends react.component {  
  Constructor (props)  
  Super (props);  
  This.state = {  
    brand = "Ford",  
    model = "Mustang",  
    color = "red",  
    year = 1960  
  };  
  } Change color = ( ) =>{  
    This.setSatate ({color: "blue" });  
  }  
  render ( ) {  
    return }  
  <div>  
    <h1> My <this.state></h1>  
    <p> It is a <this.state.color>  
    <this.state.model>  
    From <this.state.year> </p>  
    <button> type="button" onclick =  
    <this.change.color> change color  
    </button>  
  </div> };  
} }
```

Out-Put :

My Ford
It is a red Mustang From 1964
[change color] when you click
My Ford
It is a blue Mustang From 1960
[change color]

LIFECYCLE OF COMPONENTS

Each component in React has a lifecycle which you can monitor and manipulate during three main phases:

Mounting

Updating

Unmounting

Mounting

Mounting means putting element in DOM. React has a built-in method that gets called, when mounting a component

Constructor()

getDerivedStateFromProps()

render()

componentDidMount()

Constructor()

The constructor method is called by React, every time you make a Component.

getDerivedStateFromProps()

is a React static lifecycle method that updates the state based on changes in props before rendering. It helps keep the state synchronized with incoming props.

render()

is a React lifecycle method that returns the JSX to display on the screen. It describes what the UI should look like based on the current state and props.

componentDidMount()

is a React lifecycle method that runs after the component is rendered to the DOM. It's commonly used for API calls, subscriptions, or initializing data.

Updating

The Updating phase occurs when a component re-renders due to changes in props or state. React provides specific lifecycle methods to manage and respond to these updates effectively.

`getDerivedStateFromProps(props, state)`

`render ()`

`shouldComponentUpdate(nextProps, nextState)`

`getSnapshotBeforeUpdate(prevProps, prevState)`

`componentDidUpdate(prevProps, prevState, snapshot)`

`getDerivedStateFromProps(props, state)`

A static method invoked before rendering, allowing the component to update its state based on changes in props. It returns an object to update the state or null to do nothing.

`render ()`

A required method that returns the JSX representing the component's UI. It reflects the current state and props and is called during every render cycle.

`shouldComponentUpdate(nextProps, nextState)`

Determines whether the component should re-render by comparing current props and state with the next ones. Returning false prevents unnecessary renders, optimizing performance .

`getSnapshotBeforeUpdate(prevProps, prevState)`

Called right before the DOM updates, this method captures information (like scroll position) from the DOM. The returned value is passed to `componentDidUpdate`.

`componentDidUpdate(prevProps, prevState, snapshot)`

Invoked immediately after the component updates and the DOM is re-rendered. It's ideal for performing side effects like API calls or updating the DOM in response to changes.

Unmounting

The Unmounting phase happens when a component is about to be removed from the DOM. It allows you to perform cleanup like cancelling network requests or removing event listeners.

`componentWillUnmount()`

`componentWillUnmount()` is a React lifecycle method called just before a component is destroyed. It's used to clean up resources such as timers, subscriptions, or event listeners.

