SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A++' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF INFORMATION TECHNOLOGY

AI IN WEB TECHNOLOGY

III YEAR - VI SEM

UNIT 2 – JAVA SCRIPT & REACTJS

REACTJS

ReactJS

UNIT – II - JAVA SCRIPT & REACTJS

ISyntax & execution of JS, Internal, Embedded and External JS, Variables, arrays, functions, conditions, loops, Pop up boxes, objects and DOM, Inbuilt functions, Validations and Regular expressions, Event handling, Callbacks, Function as arguments Introduction to JQuery, Exploring JQuery-introduction to ReactJS, Introducing JSX, Rendering Elements, Component and Props, State and Life Cycle, Handling Events, Conditional Rendering, List and Keys, Forms, Lifting State Up, Composition vs Inheritance, Thinking in React

ReactJS

React is a JavaScript library for building user interfaces.

React is used to build single-page applications.

React allows us to create reusable UI components.

ReactJS

```
import React from "react";
import ReactDOM from "react-dom/client";

function Hello(props) {
  return <h1>Hello World!</h1>;
}

const container = document.getElementById('root');
const root = ReactDOM.createRoot(container);
root.render(<Hello />);
```

Hello World!

ReactJS

What You Should Already Know

Before starting with React.JS, you should have intermediate experience in:

- HTML
- CSS
- JavaScript

You should also have some experience with the new JavaScript features introduced in ECMAScript 6 (ES6), you will learn about them in the React ES6 chapter.

ReactJS
React Introduction

What is React?

React, sometimes referred to as a frontend JavaScript framework, is a JavaScript library created by Facebook.

React is a tool for building UI components.

ReactJS

How does React Work?

React creates a VIRTUAL DOM in memory.

Instead of manipulating the browser's DOM directly, React creates a virtual DOM in memory, where it does all the necessary manipulating, before making the changes in the browser DOM.

React only changes what needs to be changed!

React finds out what changes have been made, and changes **only** what needs to be changed.

ReactJS

React.JS History

Current version of React.JS is V18.0.0 (April 2022).

Initial Release to the Public (V0.3.0) was in July 2013.

React.JS was first used in 2011 for Facebook's Newsfeed feature.

Facebook Software Engineer, Jordan Walke, created it.

Current version of `create-react-app` is v5.0.1 (April 2022).

`create-react-app` includes built tools such as webpack, Babel, and ESLint.

ReactJS

React Getting Started

To use React in production, you need npm which is included with Node.js.

To get an overview of what React is, you can write React code directly in HTML.

But in order to use React in production, you need npm and Node.js installed.

ReactJS

```
<!DOCTYPE html>
<html>
  <head>
    <script src="https://unpkg.com/react@18/umd/react.development.js" crossorigin></script>
    <script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js" crossorigin></script>
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
  </head>
  <body>

    <div id="mydiv"></div>

    <script type="text/babel">
      function Hello() {
        return <h1>Hello World!</h1>;
      }

      const container = document.getElementById('mydiv');
      const root = ReactDOM.createRoot(container);
      root.render(<Hello />)
    </script>

  </body>
</html>
```

ReactJS

Setting up a React Environment

If you have npx and Node.js installed, you can create a React application by using `create-react-app`.

If you've previously installed `create-react-app` globally, it is recommended that you uninstall the package to ensure npx always uses the latest version of `create-react-app`.

To uninstall, run this command: `npm uninstall -g create-react-app`.

Run this command to create a React application named `my-react-app`:

```
npx create-react-app my-react-app
```

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

ReactJS

Run the React Application

Now you are ready to run your first *real* React application!

Run this command to move to the `my-react-app` directory:

```
cd my-react-app
```

Run this command to run the React application `my-react-app`:

```
npm start
```

A new browser window will pop up with your newly created React App! If not, open your browser and type `localhost:3000` in the address bar.

ReactJS

What is Dom in React?

What is ReactDOM ?

ReactDOM is a package in React that provides DOM-specific methods that can be used at the top level of a web app to enable an efficient way of managing DOM elements of the web page. ReactDOM provides the developers with an API containing the various methods to manipulate DOM.

How to use ReactDOM ?

To use the ReactDOM in any React web app we must first install the react-dom package in our project. To install the react-dom package use the following command.

```
// Installing
npm i react-dom
```

ReactJS

What is Dom in React?

After installing the package use the following command to import the package in your application file

```
// Importing
import ReactDOM from 'react-dom'
```

After installing **react-dom** it will appear in the **dependenices** in **package.json** file like:

```
"dependencies": {
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-scripts": "5.0.1",
}
```

ReactJS

What is Dom in React?

Why ReactDOM is used ?

Earlier, React Developers directly manipulated the DOM elements which resulted in frequent DOM manipulation, and each time an update was made the browser had to recalculate and repaint the whole view according to the particular CSS of the page, which made the total process to consume a lot of time.

To solve this issue, React brought into the scene the virtual DOM. The **Virtual DOM** can be referred to as a copy of the actual DOM representation that is used to hold the updates made by the user and finally reflect it over to the original Browser DOM at once consuming much lesser time.

ReactJS

What is Dom in React?

Important functions provided by ReactDOM

- render(): This is one of the most important methods of ReactDOM. This function is used to render a single React Component or several Components wrapped together in a Component or a div element.
- findDOMNode(): This function is generally used to get the DOM node where a particular React component was rendered. This method is very less used like the following can be done by adding a ref attribute to each component itself.
- unmountComponentAtNode(): This function is used to unmount or remove the React Component that was rendered to a particular container.
- **hydrate():** This method is equivalent to the render() method but is implemented while using server-side rendering.
- createPortal(): It allow us to render a component into a DOM node that resides outside the current DOM hierarchy of the parent component.

ReactJS

What is Dom in React?

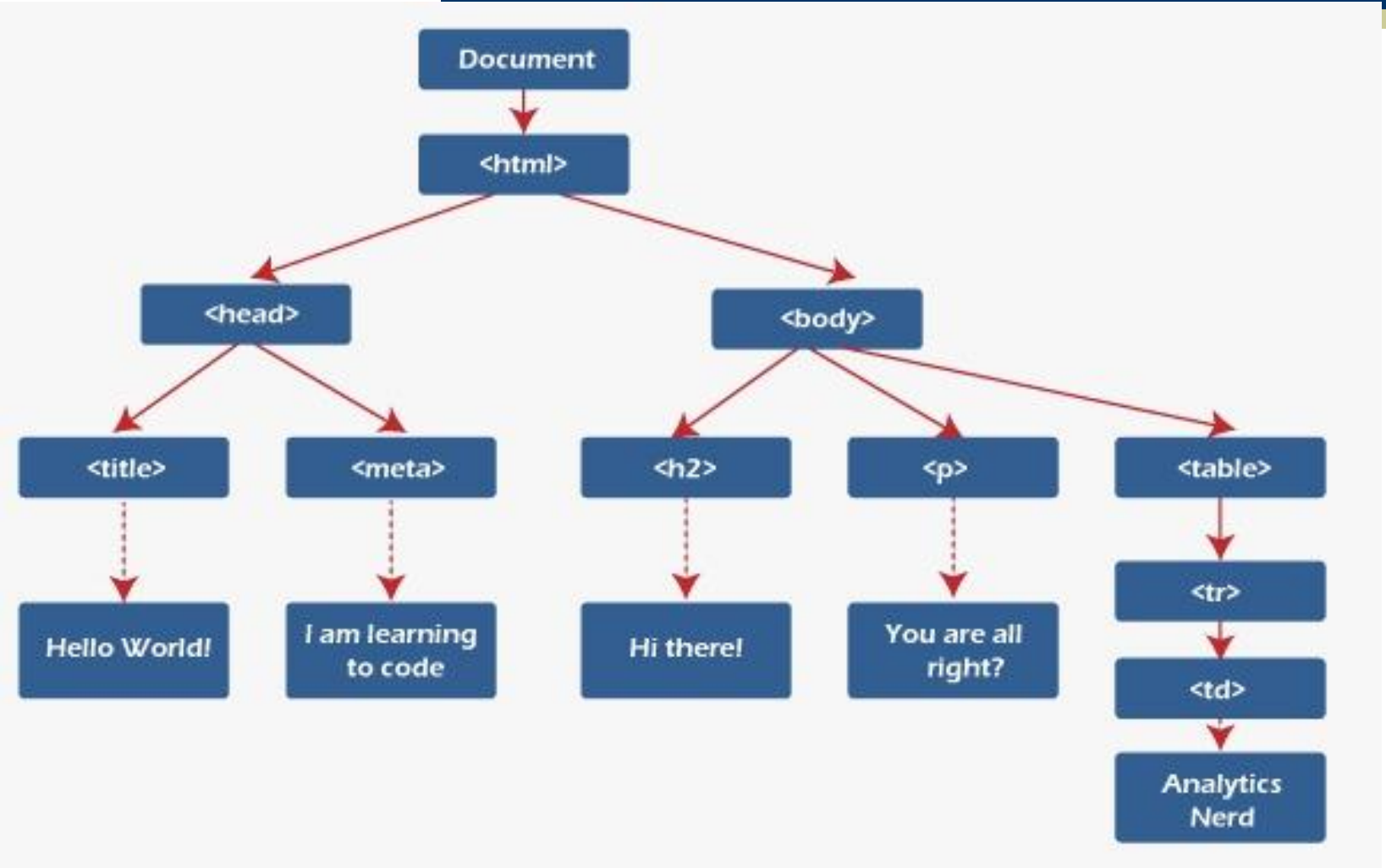
Real/Browser DOM:

DOM stands for '**Document Object Model'.** It is a structured representation of HTML in the webpage or application. It represents the entire UI**(User Interface)** of the *web application as the tree data structure.*

It is a *structural representation of HTML elements* of a web application in simple words.

ReactJS

What is Dom in React?



ReactJS

What is Dom in React?

Whenever there is any change in the *state of the application UI,* DOM is updated and represents the change. The DOM is rendered and manipulated with every change for updating the application User Interface, which affects the performance and slows it down.

Therefore, with many UI components and the complex structure of **DOM,** It will update in more expensive as it needs to be re-rendered with each change.

The DOM is constituted as a tree data structure. It consists of the node for each **UI element** present in the web document.

ReactJS

What is Dom in React?

Updating the DOM:

If we know some about JavaScript you may see people using the **'getElementById()'** or **'getElementByClass()'** method to modify the contents of DOM.

Whenever there is any change that occurs in the state of your application, the DOM is updated to reflect the change in the UI.

ReactJS

React Component Life-Cycle

In ReactJS, every component creation process involves various lifecycle methods. These lifecycle methods are termed as component's lifecycle. These lifecycle methods are not very complicated and called at various points during a component's life. The lifecycle of the component is divided into **four phases**. They are:

1. Initial Phase
2. Mounting Phase
3. Updating Phase
4. Unmounting Phase

Each phase contains some lifecycle methods that are specific to the particular phase. Let us discuss each of these phases one by one.

ReactJS

React Component Life-Cycle

1. Initial Phase

It is the **birth** phase of the lifecycle of a ReactJS component. Here, the component starts its journey on a way to the DOM. In this phase, a component contains the default Props and initial State. These default properties are done in the constructor of a component. The initial phase only occurs once and consists of the following methods.

- **getDefaultProps()**
  It is used to specify the default value of this.props. It is invoked before the creation of the component or any props from the parent is passed into it.
- **getInitialState()**
  It is used to specify the default value of this.state. It is invoked before the creation of the component.

ReactJS

React Component Life-Cycle

2. Mounting Phase

In this phase, the instance of a component is created and inserted into the DOM. It consists of the following methods.

- **componentWillMount()**
  This is invoked immediately before a component gets rendered into the DOM. In the case, when you call **setState()** inside this method, the component will not **re-render**.
- **componentDidMount()**
  This is invoked immediately after a component gets rendered and placed on the DOM. Now, you can do any DOM querying operations.
- **render()**
  This method is defined in each and every component. It is responsible for returning a single root **HTML node** element. If you don't want to render anything, you can return a **null** or **false** value.

ReactJS

React Component Life-Cycle

3. Updating Phase

It is the next phase of the lifecycle of a react component. Here, we get new **Props** and change **State**. This phase also allows to handle user interaction and provide communication with the components hierarchy. The main aim of this phase is to ensure that the component is displaying the latest version of itself. Unlike the Birth or Death phase, this phase repeats again and again. This phase consists of the following methods.

ReactJS

React Component Life-Cycle

- **componentWillRecieveProps()**

  It is invoked when a component receives new props. If you want to update the state in response to prop changes, you should compare this.props and nextProps to perform state transition by using **this.setState()** method.

- **shouldComponentUpdate()**

  It is invoked when a component decides any changes/updation to the DOM. It allows you to control the component's behavior of updating itself. If this method returns true, the component will update. Otherwise, the component will skip the updating.

ReactJS

React Component Life-Cycle

- **componentWillUpdate()**

  It is invoked just before the component updating occurs. Here, you can't change the component state by invoking **this.setState()** method. It will not be called, if **shouldComponentUpdate()** returns false.

- **render()**

  It is invoked to examine **this.props** and **this.state** and return one of the following types: React elements, Arrays and fragments, Booleans or null, String and Number. If shouldComponentUpdate() returns false, the code inside render() will be invoked again to ensure that the component displays itself properly.

- **componentDidUpdate()**

  It is invoked immediately after the component updating occurs. In this method, you can put any code inside this which you want to execute once the updating occurs. This method is not invoked for the initial render.

ReactJS

React Component Life-Cycle

4. Unmounting Phase

It is the final phase of the react component lifecycle. It is called when a component instance is **destroyed** and **unmounted** from the DOM. This phase contains only one method and is given below.

- **componentWillUnmount()**
  This method is invoked immediately before a component is destroyed and unmounted permanently. It performs any necessary **cleanup** related task such as invalidating timers, event listener, canceling network requests, or cleaning up DOM elements. If a component instance is unmounted, you cannot mount it again.