1. npm vs yarn

“npm” stands for node package manager. It is a javascript package manager and the default for the runtime environment Nodejs. Npm is installed when nodejs is installed on a machine.

“npm” comes with a command line interface CLI which is used to interact with the online database of npm. Npm database is called the npm registry, and it hosts public and private packages.

“yarn” stands for **yet another resource negotiator**. Just like npm, yarn is also a package manager for public or private packages hosted on online databases. Yarn was initially developed by developers at facebook but now is open source.

“yarn” was developed to overcome the performance and security issues that npm had at that tim. Since then, even npm developers have raised the game and both the package managers have almost reached feature parity. Even then there are some core differences between them in terms of installation, locking and usage.

**Installation**

When we install nodejs, npm also install with nodejs. But we need to manually install “yarn”.

>npm install yarn –global

Locking mechanism

“npm” makes a “package-lock.json” file. This file contains entries of all dependencies used along with their versions. This file ensures that the directory and file structure in the node\_modules folder are the same across all the project versions.

“yarn” has a “yarn.lock” file which is auto-generated when the project is initialized. This file also keeps the track of the dependencies and “locks” their versions to a project. The “yarn.lock” file helps in the easy merge of packages and dependencies in the required folders.

1. Dom

Real or browser dom

“dom” stands for document object model. It represent the entire UI of the web application as a tree data structure. It is a structural representation of the HTML elements of the web application.

document

<html>

<body>

<html>

<table>

<p>

<title>

<tr>

<div>

<meta>

You are all right

i am learning to code

React js

Hi there

<td>

Analytics nerd

React virtual dom

To make the performance of the real dom better and faster, the concept of virtual arrives. The virtual dom is nothing but the virtual representation of the dom. Every tim with every change, the virtual dom gets updated instead of the real dom.

Similar to real dom, virtual dom is also represented as a tree structure. Each element is a node in this tree as well. When a new item is added to the new UI, anode is added to the tree as well. If the state of any of these elements changes, a new virtual dom tree is created.

The virtual dom computes best possible way or we can say the minimal operations on the real dom to make changes to the real dom. Thus, it makes efficient and better performance by reducing the cost and operations of re rendering the whole real dom.

* In react, each ui is an individual component and each component has it’s own state.
* React follows the observable pattern and observes the changes of the states.
* Whenever a change is made in the state of any component, react updates the virtual dom tree but does not change the real dom.
* After updating, react then compares the current version of the virtual dom with the previous version.

React compares the virtual copy of real dom to updated copy of virtual dom, compares or picks out the changes, and finally renders it to real dom. This process is called diffing and the algorithm used is called diffing algorithm.

Reconciliation

When a component state or prop changes, react decides whether it should render the changes on real dom or not. So if the state or props of two nodes or components are not the same, then it renders the changes to real dom. This process is called reconciliation.

Reconciliation is dependent on virtual dom and diffing algorithm.

State and props

**State** are used to manage the data or dynamic information inside a component. It determine the behavior and rendering of components. When the state of a component changes, react re-renders the component to reflect the new state.

**State** is a fundamental concept in react and is essential for building interactive and dynamic user interfaces.

props are used to transfer the data from parent component to child component. Props are accessed in the child component via the props object. Props are allow us to customize and configure child component based on the data passed from the parent.

props are immutable, and their value should not be changed by the child component. props make it possible to create reusable and modular components in react.

**React render and return**

React life cycle method

Each component in react has a lifecycle which we can monitor and manipulate during its three main phases.

Mounting

constructor ()

getDeriveStateFromProps ()

render ()

componentDidMount()

Updating

getDerivedStateFromProps()

shouldComponentUpdate()

render()

getSnapshotBeforeUpdate()

componentDidUpdate()

**Mounting**

The mounting phase in react component lifecycle is the initial stage when a component is being created and inserted into the DOM. this phase consists of several key methods that are called in a specific order, allowing us to setup the component, render its initial output and perform tasks such as data fetching.

**Constructor**

* This is first method that’s called when a component is created.
* It’s typically used for two purposes:
  + Initializing the component’s state by setting **‘this.state’** and binding event handler. State represents data that can change and trigger re-renders.
  + Setting up any other initial configuration for the component.
* constructor() method is called with the props, as argument, and we should always start by calling the super(props) before anything else, this will initiate the parent’s constructor method and allows the component to inherit methods from its parent (React.Component).

**getDerivedStateFromProps**

The getDerivedStateFromProps () method is called right before rendering the elements in the dom. natural place to set the object on the initial props.it takes state as argument and returns a object with changes to the state.

**render ()**

* render () method is required, and is the method that actually outputs the HTML to the DOM.
* This method is called after the constructor and is responsible for returning the JSX (javascript XML).
* It should be a pure function, it shouldn’t have any side effects or make any direct changes to the dom or component state. Its sole purpose is to describe the component’s view.
* React takes the JSX retuned by the render method and uses it to construct the initial dom representation of the component.

**componentDidMount**

* componentDidMount method is called after the component is rendered.
* It’s a good place to perform tasks that require access to the dom, such as data fetching from a api or setting up third-party libraries that need access to the component’s rendered elements.
* It is commonly used for initialization and setting up event listeners.
* It’s important to note that this method is only called once, right after the initial rendering, and not during subsequent re-renders.

**Updating**

the update phase in react component lifecycle is the stage where a component is re-rendered because of changes in its props or state. During this phase, react compares the previous output of the component with the new output generated by the component’s render method. If there are differences, react updates the actual dom to reflect those changes.

**shouldComponentUpdate**

* this is a optional method that can be defined in class components.
* It is called before the component re-renders and allows us to control whether the component should update or not.
* By default, this method returns ‘true’, indicating that the component should re-render. However, we can implement custom logic to optimize performance by returning ‘false’ when the component doesn’t

**render**

As in the mounting phase the ‘render’ method is called during the update phase. It generates the new output for the component based on the updated props or state.it is always be a pure function and not have a side effects.

**componentDidUpdate**

After the component’s re-render, the ‘componentDidUpdate’ method is called. It’s a suitable place for handling side effects that are dependent on the component’s updated state or props.

common use cases include making network requests to fetch updated data, performing animations, or interacting with the dom based on the new content.

getDerivedStateFromProps ()

render() method is responsible for generating the components virtual dom representation based on its current props and state.it is call every tim the component needs to be re-rendered, either because its props or state have changed, or because of parent component re-rendered.

**Unmounting phase**

Unmounting phase refers to the lifecycle stage when a component is being removed from the dom and is no longer rendered or accessible. During this phase, react performs a series of cleanup operations to ensure that the component and its associated resources are properly disposed of.

React element and React component

React element is the basic building block in a react application. It is a object representation of a virtual dom node. It contains both type and property. It may contain other elements in its props.

React element doesn’t have any methods, making it light and faster to render than components.

React Component it

|  |  |
| --- | --- |
| React component | React element |
| Component are like javascript functions that accepts input and return react elements. | Element is a plain object describing a component instance or dom node and its desired properties. |
| Components state is mutable. | Element are immutable. Once we create a element, we can’t change its children and attributes. |
| Components have their own lifecycle methods like component mounts, updates and unmounts. We can use effects to achieve component lifecycle methods. | Element doesn’t have any methods. |
| Component are slower than elements. | Element creation is cheap and it doesn’t have a state, therefore it is faster than components. |
| Component can be declared in several ways, such as a class with a render () method or as a function. | Element can be created using React.createElement (), JSX or a element factory helper. |