**React Docs**

**DOM vs Virtual DOM:**

DOM or Document object model is a tree like representation of the html document that browser keeps the DOM represents the document as nodes and objects, that way programming languages can interact with the page.

V-DOM or Virtual DOM is a light weight representation of the original DOM which react keeps in memory and in sync with the DOM.

Now making changes directly to the DOM is very expensive and slower than making changes to the virtual DOM as virtual DOM doesn’t has the power to directly change what’s on the screen hence nothing gets drawn on screen.

When anything new is added to the application, a virtual DOM is created and it is represented as a tree. Each element in the application is a node in this tree. So, whenever there is a change in the state of any element, a new Virtual DOM tree is created. This new Virtual DOM tree is then compared with the previous Virtual DOM tree and make a note of the changes. After this, it finds the best possible ways to make these changes to the real DOM. Now only the updated elements will get rendered on the page again.

**React Fiber:**

Fiber is the new reconciliation engine in React16. Its main goal is to enable incremental rendering of the virtual DOM.

The entire process of transforming changes to the real DOM is called Reconciliation.

**Events in React:**

React supported events are:

* Clipboard events (onCopy, onCut, onPaste)
* Composition events
* Keyboard events (onKeyDown onKeyPress, onKeyUp)
* Focus events( onFocus, onBlur)
* Form events (onChange, onInput, onInvalid, onReset, onSubmit)
* Generic events (onError, onLoad)
* Mouse events ( onClick, onDoubleClick, onDrag, onMouseEnter, …)
* Pointer events (onPointerDown, onPointerMove, ….)
* Selection events (onSelect)
* Touch events (onTouchCancel, onTouchEnd, onTouchMove, onTouchStart)
* UI events (onScroll)
* Wheel events (onWheel)
* Media events (onLoad, onError)
* Animation events (onAnimationStart, onAnimationEnd, onAnimationIteration)
* Other events (onToggle)

**Stateful components:**

In React, a stateful component is a component that holds some state. Stateless component, by contrast, have no state. Note that both types of components can use props.

**Binding in react event handlers:**

**prop Types:**

As React is a javaScript library and javaScript is a dynamically typed language we use extensions like typeScript to get that sweet type-safety and catch bugs early on in the development phase, but even if we don’t use it React has a built in tool known as propTypes with which we can easily define the type of props that a component will be receiving.

**Working of react/ imp features:**

**Features of ReactJS:**

i) JSX: JavaScript Syntax Extension is a combination of HTML and JS. You can embed javaScript objects inside the HTML elements. JSX is not supported by the browsers, as a result Babel compiler transcompile the code into JS code. JSX makes codes easy and understandable. It is easy to learn if you know HTML and JS.

ii)Virtual DOM: DOM stands for Document Object Model. It is the most imp part of the web as it divides into modules and executes the code. Usually, JS Frameworks updated the whole DOM at once, which makes the web application slow. But react uses virtual DOM which is an exact copy of the real DOM. Whenever there is a modification in the web application, the whole virtual DOM is updated first and finds the difference between real DOM and Virtual DOM. Once it finds the difference, then DOM updates only the part that has changed recently and everything remains the same and this process is known as reconciliation.

iii) One-way Data Binding: One-way data binding, the name itself says that its a one-directional flow. The data in react flows only in one direction i.e. the data is transferred from top to bottom i.e. from parent components to child components. The properties(props) in the child component cannot return the data to its parent component but it can have communication with the parent components to modify the states according to the provided inputs. This is the working process off one-way data binding. This keeps everything modular and fast.

iv)Performance: As React uses VDOM and updates only the modified parts. So, this makes the DOM to run faster. DOM executes in memory so we can create separate components which makes the DOM run faster.

v)Extension: React has many extensions that we can use to create full-fledged UI applications. It supports mobil app development and provide server-side rendering. React is extended with Flux. Redux, React Native, etc. which helps us to create good-looking UI.

vi) Conditional Statements: JSX allows us to write conditional statements. The data in the browser is displayed according to the conditions provided inside the JSX.

vii) Components: ReactJS divides the web page into multiple components as it is components-based. Each component is a part of the UI design which has its own logic and design. So the component logic which is written in JS makes it easy and run faster and can be resable.

viii)Simplicity: React is a component-based which makes the code reusable and React.js uses JSX which is a combination of HTML and JS and makes the code declarative. This makes code easy to understand and easy to debug and has less code.

**JSX:**

JSX, is a syntax extension/ syntactic sugar to JS. We recommend using it with React to describe what the UI should look like. JSX produces React “elements”.

React doesn’t require JSX, but most people find it helpful as a visual aid when working with UI inside the JavaScript code.

**Can browsers read JSX:**

No, Browsers can’t read JSX because there is not inherent implementation for the browser engines to read and understand them. JSX is not intended to be implemented by the engines or browsers.

**Features, advantages and Limitations of React:**

**Features/ Advantages:**

Easy to learn and use.

Fast.

3rd party community library and packages.

React supports handy dev tools extension for firefox and chromium based browsers.

Reusable components.

**Limitations:**

As React is more of a library rather than a framework it comes with a lesser number of development tools out of the box.

No limitations or set of protocols to develop a webapp sometimes can be seen as an advantage but sometimes seems out to be a limitation as many people will have many different solution for the same problem and handing off in this setting becomes problematic.

**How rendering works in React:**

**States vs Props:**

The Sate represents parts of an application that can change. Each component can have its state. The state is mutable and it is local to the component only

Props are know as properties it can be used to pass data from one component to another. Props cannot be modified, read-only, and immutable.

**Use of an arrow function:**

**React Architecture:**

**Higher order components(HOC’s) need/use:**

**3 phases of a components life cycle:**

**How are events created:**

**How is routing in react diff from conventional**

**routing:**

**Diff b/w flux and redux:**

**Synthetic events:**

event.preventDefault();

**refs in React:**

**Purpose of render():**

**LifeCycle methods in REACT:**

**Controlled v/s Uncontrolled components:**

**Pure Components:**

**Keys?**

**Redux? Principles? Components of Redux?**

**Advantages?**

**React-Router?**

**Use of <Switch /> in react-router-dom ?**

**React-router-dom Hooks?**

**Axios?**

**Instance Property vs State property:**

**Why is Router required in React**

**How can you tell react to build in prod mode:**

**Diff b/w clone element & create Element:**

**Strict mode component:**

**<React.StrictMode />** is sort of a helper component that will help you write better React components, you can wrap a set of components with <React.StrictMode /> and it’ll basically:

i) Verify that the components inside are following some of the recommended practices and warn you if not in the console.

ii) Verify the deprecated methods are not being used, and if they’re used strict mode will warn you in the console.

iii) Help you prevent some side effects by identifying potential risks.

As the documentation says, strict mode is development oriented so you don’t need to worry about it impacting on your production build.

I’ve found it especially useful to implement strict mode when I’m working on new code bases and I want to see what kind of code/components I’m facing. Also if you’re on bug hinting mode, sometimes it’s a good idea to wrap with <StrictMode /*> the components* blocks of code you think might be the source of the problem.

**Hooks**