**Why React?**

Popularity -

How many starts?

How many active contributors are present across the world?

**Core Features:**

1. Easy to learn. (Learning curve is not very steep).

2. Good community support.

3. Built by keeping design principles in mind (SoC etc...)

4. Battle tested. (Facebook, Instagram, WhatsApp).

5. Better performance (uses Virtual DOM concept).

6. Zipped version will come around (40KB).

7. Easy to Debug/Test.

**What is DOM/Virtual DOM?**

1. DOM stands for Document Object Model.

2. It is an **API** to represent and interact with an HTML document.

3. The DOM is a **bridge between the HTML world of elements (paragraphs, lists, divs, and so on) and the JavaScript** world of variables, functions, objects, and methods.

4. Every web browser has the DOM built-in.

5. Virtual DOM - in memory representation of DOM. (Concept of Java Script).

============

**React - Timer Example.**

============

**What is**

Package manager – npm/yarn.

Bundler – Webpack/Browserify.

Compiler – Babel.

============

Proxy settings for libraries to install.

Install ‘create-react-app’.

============

**Why JSX?**

1. It is called JSX, and it is a **syntax extension** to JavaScript.

2. JSX may **remind you of a template language**, but it comes with the full power of JavaScript.

3. Instead of *artificially separating technologies by putting markup and logic in separate files*, React **separates concerns** with loosely coupled units called “**components**” that contain both.

4. JSX is an Expression Too - <https://reactjs.org/docs/introducing-jsx.html#jsx-is-an-expression-too>

5. Specifying Attributes with JSX - <https://reactjs.org/docs/introducing-jsx.html#specifying-attributes-with-jsx>

6. JSX **Prevents Injection Attacks** - <https://reactjs.org/docs/introducing-jsx.html#jsx-prevents-injection-attacks>

============

**Rendering Elements**

1. Rendering an Element into the DOM.

*(Open the same timer example and explain the clock rendering into div 'root')*.

2. React Only Updates What’s Necessary - <https://reactjs.org/docs/rendering-elements.html#react-only-updates-whats-necessary>

3. React DOM **compares the element and its children to the previous one**, and *only applies the DOM updates necessary to bring the DOM to the desired state*.

4. Even though we create an element describing the whole UI tree on every tick, only the text node whose **contents has changed gets updated by React DOM**.

============

**Functional and Class Components**

1. Functional and Class Components - <https://reactjs.org/docs/components-and-props.html#functional-and-class-components>

2. The **'Welcome'** function is a valid React component because it accepts a single “**props**” (which stands for properties) object argument with data and returns a React element.

3. We call such components “functional” because they are literally JavaScript functions.

4. Classes have some **additional features** that we will discuss in the next sections.

5. Rendering a Component - <https://reactjs.org/docs/components-and-props.html#rendering-a-component>

6. Composing Components - <https://reactjs.org/docs/components-and-props.html#composing-components>

============

**What is Props?**

1. Props are **Read-Only** - <https://reactjs.org/docs/components-and-props.html#props-are-read-only>

2. *Pure functions* - It must never modify its own props.

3. All **React components must act like pure functions** *with respect to their props*.

============

**What is State?**

1. State allows React **components to change their output over time** in response to user actions, network responses, and anything else, without violating the above rule.