# Tell me about yourself?

I am working as react full stack developer. In total I have (x) years of experience on various technologies like React js, Redux, HTML5, CSS3, ES6 and rest API’s using node js.

My primary roles and responsibilities are requirement analysis, design, font end development, api integration and delivery of applications.

I have worked with various clients, product team, graphic designers, and testing team and on multiple domains like ecommerce and b2b applications.

# What are the inputs for your projects?

We will get the read-only access to **GitHub** for the following documents

SRS 🡺 Software Requirement Specification

This will have requirements, Tools, and Software to be used.

SDD 🡺Software Design Document

Flow charts, and Constants and Variables to be used

Figma Design 🡺 for UI Design

UI Design for Development

Our team has 5-20 resources

Scrum Master 🡺 Usually project manager

Product Owner 🡺 Team Lead => Assigns the Story (Work)

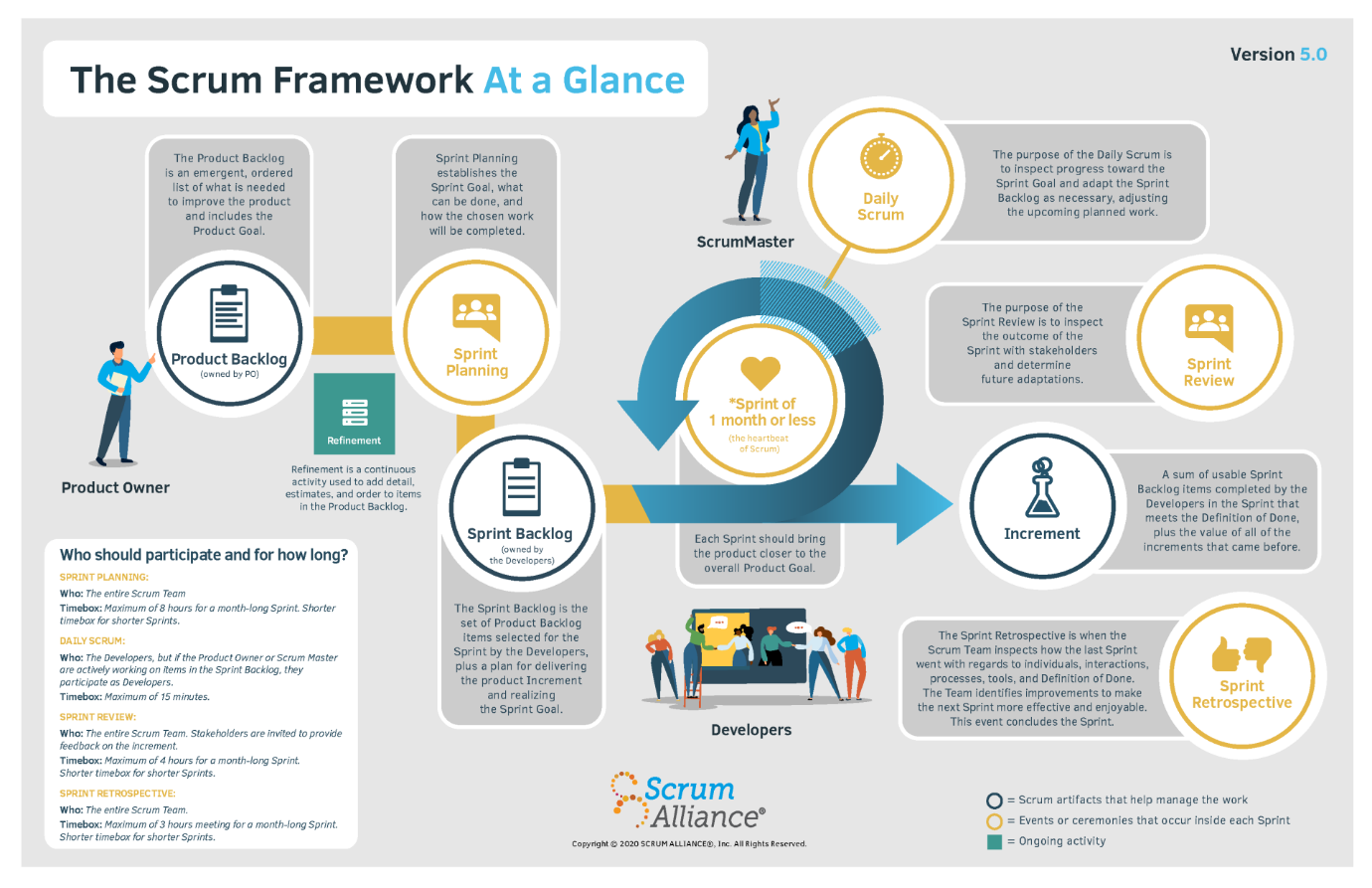
Stakeholders 🡺 Increments the sprints

Team Members 🡺 works on Stories

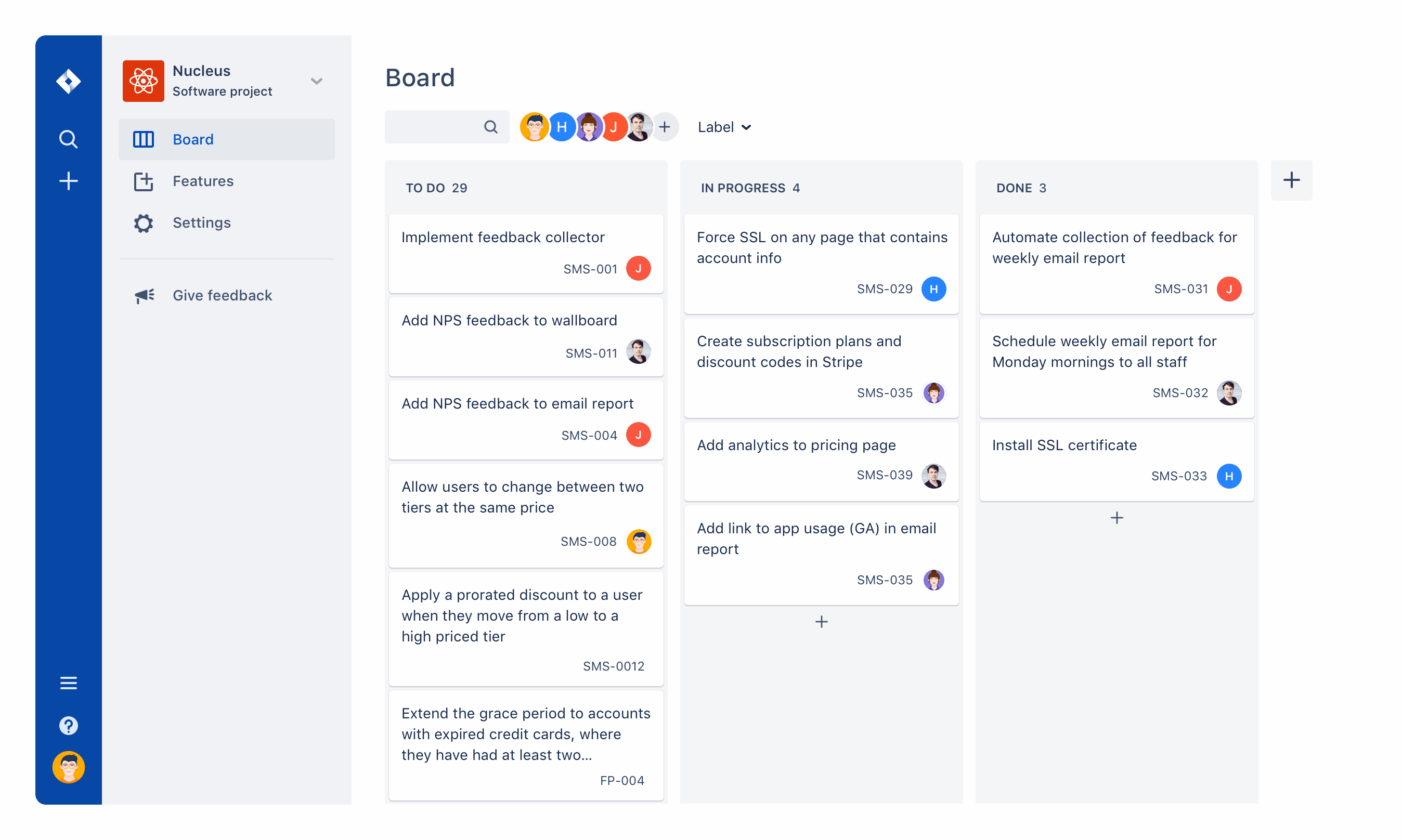
## Can you describe the agile process / Can you describe your project process?

We are following Scrum Process

1. Product Backlog
2. Sprint Meeting
3. Sprint Backlog
4. Sprint Review and Retrospect



Jira Tool for project management



## Why is version control important/ What is the version control (VCS) system you use in your project

<https://www.nobledesktop.com/blog/what-is-git-and-why-should-you-use-it>

## What are the middleware you have used

<https://selvaganesh93.medium.com/how-node-js-middleware-works-d8e02a936113#:~:text=Middleware%20functions%20are%20functions%20that%20have%20access%20to%20the%20request,by%20a%20variable%20named%20next%20>.

## How many resources will a browser download from a given domain at a time?

https://blog.stackpath.com/glossary-domain-sharding/#:~:text=Modern%20web%20browsers%20support%20on,requests%20for%20a%20single%20user.

## Ways to decrease page load (perceived or actual load time).

https://blog.hubspot.com/marketing/how-to-reduce-your-websites-page-speed

## What is an IDE?

Integrated development environment

Ex: Visual Studio Code

## What is MVC?

The **Model-View-Controller (MVC)** is an architectural pattern that separates an application into three main logical components: the **model**, the view, and the controller. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects

## What is ‘CORS’ and `CSRF`? What is it used for?

https://medium.com/@electra\_chong/what-is-cors-what-is-it-used-for-308cafa4df1a

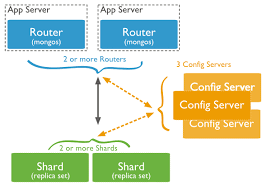
## 

## What is ‘Indexing’ in MySQL

<https://www.liquidweb.com/kb/mysql-optimization-how-to-leverage-mysql-database-indexing/#:~:text=Indexing%20is%20a%20powerful%20structure,be%20used%20to%20enforce%20uniqueness>.

## Sharding 🡺 Mongo DB

https://www.bmc.com/blogs/mongodb-sharding-explained/#:~:text=Sharding%20is%20the%20process%20of,sets%20across%20multiple%20MongoDB%20instances.



# **React JS**

## What is React?

React is an open-source front-end JavaScript library that is used for building user interfaces, especially for single-page applications. It is used for handling view layer for web and mobile apps.

## What are the major features of React?

* It uses VirtualDOM instead of RealDOM considering that RealDOM manipulations are expensive.
* Supports server-side rendering.
* Follows Unidirectional data flow or data binding.
* Uses reusable/composable UI components to develop the view.

## What is JSX?

JSX is a XML-like syntax extension to ECMAScript (the acronym stands for JavaScript XML). Basically it just provides syntactic sugar for the React.createElement() function, giving us expressiveness of JavaScript along with HTML like template syntax.

## What is the difference between Element and Component?

An Element is a plain object describing what you want to appear on the screen in terms of the DOM nodes or other components. Elements can contain other Elements in their props. Creating a React element is cheap. Once an element is created, it is never mutated.

The object representation of React Element would be as follows:

const element = React.createElement(

'div',

{id: 'login-btn'},

'Login'

)

The above React.createElement() function returns an object:

{

type: 'div',

props: {

children: 'Login',

id: 'login-btn'

}

}

And finally it renders to the DOM using ReactDOM.render():

<div id='login-btn'>Login</div>

Whereas a component can be declared in several different ways. It can be a class with a render() method or it can be defined as a function. In either case, it takes props as an input, and returns a JSX tree as the output:

const Button = ({ onLogin }) =>

<div id={'login-btn'} onClick={onLogin}>Login</div>

Then JSX gets transpiled to a React.createElement() function tree:

const Button = ({ onLogin }) => React.createElement(

'div',

{ id: 'login-btn', onClick: onLogin },

'Login'

)

## How can we find the version of React at runtime in the browser?

You can use React.version to get the version.

const REACT\_VERSION = React.version

ReactDOM.render(

<div>{`React version: ${REACT\_VERSION}`}</div>,

document.getElementById('app'))

## Why you can't update props in React?

The React philosophy is that props should be immutable and top-down. This means that a parent can send any prop values to a child, but the child can't modify received props.

## What are Pure Components?

React.PureComponent is exactly the same as React.Component except that it handles the shouldComponentUpdate() method for you. When props or state changes, PureComponent will do a shallow comparison on both props and state. Component on the other hand won't compare current props and state to next out of the box. Thus, the component will re-render by default whenever shouldComponentUpdate is called.

## What is state in React?

State of a component is an object that holds some information that may change over the lifetime of the component. We should always try to make our state as simple as possible and minimize the number of stateful components.

Let's create a user component with message state,

class User extends React.Component {

constructor(props) {

super(props)

this.state = {

message: 'Welcome to React world'

}

}

render() {

return (

<div>

<h1>{this.state.message}</h1>

</div>)

}

}

State is similar to props, but it is private and fully controlled by the component, i.e., it is not accessible to any other component till the owner component decides to pass it.

## What are props in React?

Props are inputs to components. They are single values or objects containing a set of values that are passed to components on creation using a naming convention similar to HTML-tag attributes. They are data passed down from a parent component to a child component.

The primary purpose of props in React is to provide following component functionality:

i. Pass custom data to your component.

ii. Trigger state changes.

iii. Use via this.props.reactProp inside component's render() method.

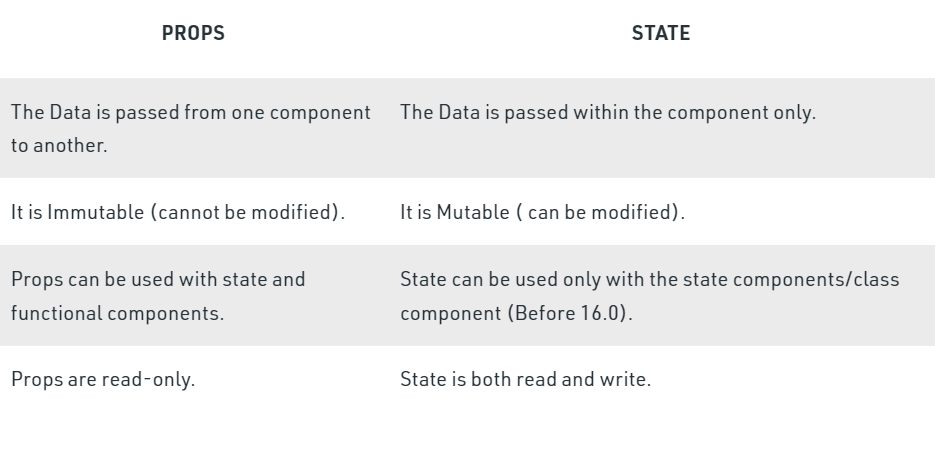
For example, let us create an element with reactProp property:

<Element reactProp={'1'} />

This reactProp (or whatever you came up with) name then becomes a property attached to React's native props object which originally already exists on all components created using React library.

props.reactProp

## What is the difference between state and props?



* Props are used to pass data from one component to another.
* The state is a local data storage that is local to the component only and cannot be passed to other components.
* The this.setState property is used to update the state values in the component

<https://www.youtube.com/watch?v=dMH1_YtUTSQ>

## What is the difference between HTML and React event handling?

* In HTML, the event name usually represents in lowercase as a convention:

<button onclick='activateLasers()'>

Whereas in React it follows camelCase convention:

<button onClick={activateLasers}>

* In HTML, you can return false to prevent default behavior:

<a href='#' onclick='console.log("The link was clicked."); return false;' />

Whereas in React you must call preventDefault() explicitly:

function handleClick(event) {

event.preventDefault()

console.log('The link was clicked.')

}

* In HTML, you need to invoke the function by appending () Whereas in react you should not append () with the function name. (refer "activateLasers" function in the first point for example)

## What is "key" prop and what is the benefit of using it in arrays of elements?

A key is a string attribute you should include when creating arrays of elements. Key prop helps React identify which items have changed, are added, or are removed.

Most often we use ID from our data as key:

const todoItems = todos.map((todo) =>

<li key={todo.id}>

{todo.text}

</li>

)

When you don't have stable IDs for rendered items, you may use the item index as a key as a last resort:

const todoItems = todos.map((todo, index) =>

<li key={index}>

{todo.text}

</li>

)

## What is the use of refs?

The ref is used to return a reference to the element. They should be avoided in most cases, however, they can be useful when you need a direct access to the DOM element or an instance of a component.

## What is Virtual DOM?

The Virtual DOM (VDOM) is an in-memory representation of Real DOM. The representation of a UI is kept in memory and synced with the "real" DOM. It's a step that happens between the render function being called and the displaying of elements on the screen. This entire process is called reconciliation.

The Virtual DOM works in three simple steps.

* Whenever any underlying data changes, the entire UI is re-rendered in Virtual DOM representation.
* Then the difference between the previous DOM representation and the new one is calculated.
* Once the calculations are done, the real DOM will be updated with only the things that have changed.

## What are controlled components and uncontrolled components?

**A Controlled Component** is one that takes its current value through props and notifies changes through callbacks like onChange. A parent component "controls" it by handling the callback and managing its own state and passing the new values as props to the controlled component. You could also call this a "dumb component".

**A Uncontrolled Component** is one that stores its own state internally, and you query the DOM using a ref to find its current value when you need it. This is a bit more like traditional HTML.

Most native React form components support both controlled and uncontrolled usage:

// Controlled:

<input type="text" value={value} onChange={handleChange} />

// Uncontrolled:

<input type="text" defaultValue="foo" ref={inputRef} />

// Use `inputRef.current.value` to read the current value of <input>

## What is Lifting State Up in React?

When several components need to share the same changing data then it is recommended to lift the shared state up to their closest common ancestor. That means if two child components share the same data from its parent, then move the state to parent instead of maintaining local state in both of the child components.

## What are the different phases of component lifecycle?

1. Mounting: The component is ready to mount in the browser DOM. This phase covers initialization from constructor(), getDerivedStateFromProps(), render(), and componentDidMount() lifecycle methods.
2. Updating: In this phase, the component gets updated in two ways, sending the new props and updating the state either from setState() or forceUpdate(). This phase covers getDerivedStateFromProps(), shouldComponentUpdate(), render(), getSnapshotBeforeUpdate() and componentDidUpdate() lifecycle methods.
3. Unmounting: In this last phase, the component is not needed and gets unmounted from the browser DOM. This phase includes componentWillUnmount() lifecycle method.

It is worth mentioning that React internally has a concept of phases when applying changes to the DOM. They are separated as follows

* Render The component will render without any side effects. This applies to Pure components and in this phase, React can pause, abort, or restart the render.
* Pre-commit Before the component actually applies the changes to the DOM, there is a moment that allows React to read from the DOM through the getSnapshotBeforeUpdate().
* Commit React works with the DOM and executes the final lifecycles respectively componentDidMount() for mounting, componentDidUpdate() for updating, and componentWillUnmount() for unmounting.

<https://www.youtube.com/watch?v=S_JCXnKIyxI>

## What are Higher-Order Components?

A higher-order component (HOC) is a function that takes a component and returns a new component. Basically, it's a pattern that is derived from React's compositional nature.

We call them pure components because they can accept any dynamically provided child component but they won't modify or copy any behavior from their input components.

const EnhancedComponent = higherOrderComponent(WrappedComponent)

HOC can be used for many use cases:

i. Code reuse, logic and bootstrap abstraction.

ii. Render hijacking.

iii. State abstraction and manipulation.

iv. Props manipulation.

<https://www.youtube.com/watch?v=A9_9gQIkfx4>

## How to apply validation on props in React?

When the application is running in development mode, React will automatically check all props that we set on components to make sure they have correct type. If the type is incorrect, React will generate warning messages in the console. It's disabled in production mode due to performance impact. The mandatory props are defined with isRequired.

i. PropTypes.number

ii. PropTypes.string

iii. PropTypes.array

iv. PropTypes.object

v. PropTypes.func

vi. PropTypes.node

vii. PropTypes.element

viii. PropTypes.bool

ix. PropTypes.symbol

x. PropTypes.any

We can define propTypes for User component as below:

import React from 'react'

import PropTypes from 'prop-types'

class User extends React.Component {

static propTypes = {

name: PropTypes.string.isRequired,

age: PropTypes.number.isRequired

}

render() {

return (

<>

<h1>{`Welcome, ${this.props.name}`}</h1>

<h2>{`Age, ${this.props.age}`}</h2>

</>

)

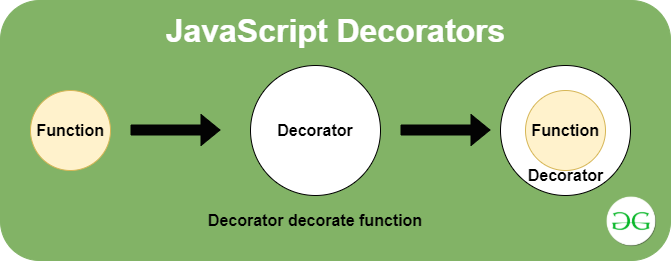
}

}

## What will happen if you use setState() in constructor?

When you use setState(), then apart from assigning to the object state React also re-renders the component and all its children. You would get error like this: Can only update a mounted or mounting component. So we need to use this.state to initialize variables inside constructor.

Decorators in React



**Decorators** are the way of wrapping one piece of code with another or apply a wrapper around a function in JavaScript. **Decorators** are the design pattern that allows behaviour to be added to an individual object, either statically or dynamically without affecting the behaviour of other objects from the same class. They are used to enhance the functionality of the function without modifying the underlying function. They are just modifying the behaviour of the function or method passed to it by returning a new function

For all practical reasons, decorators and HOC (Higher-Order-Component aka Wrapper) do the same thing.

One major difference is that, once you add a decorator, the **property/class can only be used in its decorated form (can render anything)**. HOC pattern leaves higher order as well as the lower order components available for use.

You can decorate your class components, which is the same as passing the component into a function. Decorators are flexible and readable way of modifying component functionality.

@setTitle('Profile')

class Profile extends React.Component {

//....

}

/\*

title is a string that will be set as a document title

WrappedComponent is what our decorator will receive when

put directly above a component class as seen in the example above

\*/

const setTitle = (title) => (WrappedComponent) => {

return class extends React.Component {

componentDidMount() {

document.title = title

}

render() {

return <WrappedComponent {...this.props} />

}

}

}

## How you implement Server Side Rendering or SSR?

React is already equipped to handle rendering on Node servers. A special version of the DOM renderer is available, which follows the same pattern as on the client side.

import ReactDOMServer from 'react-dom/server'

import App from './App'

ReactDOMServer.renderToString(<App />)

This method will output the regular HTML as a string, which can be then placed inside a page body as part of the server response. On the client side, React detects the pre-rendered content and seamlessly picks up where it left off.

## What is CRA and its benefits?

The create-react-app (CRA) CLI tool allows you to quickly create & run React applications with no configuration step.

## How do you access props in attribute quotes?

React (or JSX) doesn't support variable interpolation inside an attribute value. The below representation won't work:

<img className='image' src='images/{this.props.image}' />

But you can put any JS expression inside curly braces as the entire attribute value. So the below expression works:

<img className='image' src={'images/' + this.props.image} />

Using template strings will also work:

<img className='image' src={`images/${this.props.image}`} />

## What is the recommended approach of removing an array element in React state?

The better approach is to use Array.prototype.filter() method.

For example, let's create a removeItem() method for updating the state.

removeItem(index) {

this.setState({

data: this.state.data.filter((item, i) => i !== index)

})

}

## How to use https instead of http in create-react-app?

You just need to use HTTPS=true configuration. You can edit your package.json scripts section:

"scripts": {

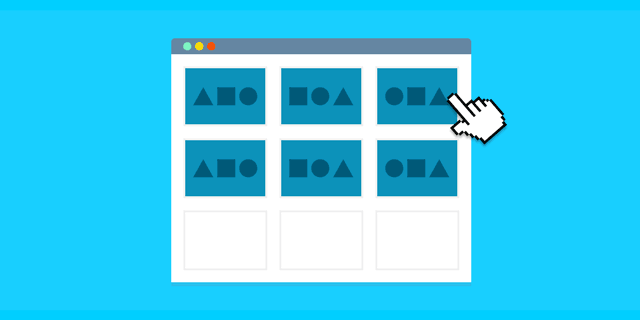
"start": "set HTTPS=true && react-scripts start"

}

or just run set HTTPS=true && npm start

## Lazy loading

Lazy loading is **a strategy to identify resources as non-blocking (non-critical) and load these only when needed**. It's a way to shorten the length of the critical rendering path, which translates into reduced page load times. Lazy loading solves the problem of reducing initial page load time — displaying only resources like images or other content that a user needs to



## React Routers

<https://www.w3schools.com/react/react_router.asp>

<https://v5.reactrouter.com/web/guides/quick-start>

## UNIT Testing on Node

<https://www.youtube.com/watch?v=FgnxcUQ5vho>

## Server-Side Rendering VS. Client-Side Rendering

<https://www.clariontech.com/blog/server-side-rendering-vs.-client-side-rendering>