# Controlled Vs Uncontrolled Components

## Uncontrolled Components

Let Say we have input

```
<input placeholder="name" ref={name} />

let name = React.createRef();
```

onsubmit we get value as (name.current.value)
In this case we cannot control the value of input.

To control value we need to make changes

```
<input placeholder="name" onChange={ } />
```

instead of ref we use onchange Here we can control the value.

## Reconcilation

When we do changes or change states in react. the virtual dom provided by the react is compared with browser real DOM & changes are impactable then changes of virtual dom is pushed into

the real DOM this complete of changing DOM is known as Reconcilation.

# InnerHTML

Instead of rendering JSX into HTML sometimes directly render HTML By injecting HTML files

For x

```
let data = '<p stype="font-size: 25px;
                color: red"> This is Home </p>';
```

To render the above line we use

"dangerouslySetInnerHTML"

```
<div dangerouslySetInnerHTML={{__html: data}}>
</div>
```

# Fragment

<> </>

<React.fragment> </React.fragment>

both are same.

Normal   creates Node in DOM sometimes we don't need that purpose we use fragments.

## Rest API Request

In react we can use Axios to fetch or send request.

1. Install Axios
2. Import Axios

```
import axios from "axios";
```

## To Post Data

★ axios.Post("url", {object to sent});

(using Promise)

★ axios.post("URL", {obj}).then((res) => {}).
then((err) => {});

## GET Data

★ Mostly we use Useffect with GET requ

★ axios.get("URL").then((res) => {}).then(
(err) => {});

# Put or PATCH

We used Put or PATCH to update data.

# Debouncing

Basic making API request after sometime for we can use setTimeOut function.

# Context API

Sharing Common State in every Component.

* Make a file

```
Export const UseData = CreateContext([]);
                                      ↑
                                  can use any thing
```

* Wrap all the child You want to send data

```
<UserData.Provider name={name}>
    <child1/>
    <child2/>
</UserData.Provider>
```

* Import context to desired child using UseContext API.

```
Const { name } = UseContext(Usedata)
```

Holla Impressed

## Class and ClassName in React

In react/Javascript class is reserve keyword for CLASS. So we use ClassName instead.

# Higher Order Components in React

When Same peice of code is written for each components in react we use Higher Order Component to remove this redundant writting of code:
For example if there is two pages Post1, Post2 both are fetching same API we need to create Higher Order Component for this.

## HOC

```
export default function HOC (component) {

  return function HOC () {

    =
    =
    =

  }

};
```

# Lazy Loading

In react most of the library are large enough to load Sometimes take much more time than need. For this we use Lazy load.

```
const Home = React.Lazy(() => import("./Loaem"));
```

## Suspend

```
import { Suspend } from "react";

<Supend fallback={<loading Componnt />}>
    <Child />
</suspend>
```

Basically Loading is shown here.

# Recursion in react

Let Suppose we have some consisting of parent and child. The child can consist of childs. In this when we need to render at element we use recursion.

# Custom Hooks in React

filename: useJSONPlaceholder.js

We can create custom Hooks in react JS Just Import useState and Creat it

```
import {useState, useEffects} from "React";

Function useData() {

  const [users, setUsers] = useState([]);
  const [pots, setPosts] = useState([]);

const getUsers = () => {
    fetch("url").then((response)) => {}.
      then(json) => SetUsers(jon);

  };
```

```
Const getPost = () => {
  fetch("URL").then((response) => { }.
        then((jon)) => SetPost(jon));
  };

Useeffect(() => { getusers(), getPost(), }, []);

return { users, getUsers };

}

export default UserPath;
```

## Promises in Javascript/React

```
Const Checle(parm) => {
    return new response(response, resolve) => {
        If(condition true) {
            response("Good");
        }
        else {
            reject("no Good");
        }
    }
```

# Dynamic Imports in React

Sometime we just need one function from another but for that we usually load complete file. Instead of loading complete file we can simply import required function we needed such type of loading is called Dynamic loading.

```
const add = () => {
  import("./Math").then((Maths) => {
      let sum = math.add(5,6);
      console.log(sum);
    });

  }
};
```

## Advantages of react

* SPA - Singe page Application
* Cross platform and open source
* Lightweight & fast (Virtual DOM)
* Large community and Ecosystem
* Testing is easy

## Disadvantages of react

* React no good choice for small application.

## DOM Stucture

Tree like representation of HTML dynamically acces and manipulate the content and structure a web page.

## Virtual DOM

Same as DOM but its virtual copy of DOM. React uses virtual DOM to efficiently update the UI without re-render the entire page, which helps improve performance and application more responsive.

* There can more than one Virtual DOM (NO need to worry)

## React Components

* In react, a components is a reusable building blocks for creating User Interface

## Advantages of JSX

* Improve code redability and writability.
* Error checking in advance.
* Support Javascript extension
* Improved performance.
* Code resusability.

**Babel** → Convert JSX into regular Javascript.

**Transpiler** A transpiler is a tool that converts source code from one programming language to another.

fragment → (<> </>) ⇒ In react fragment is a way to group a list of children without adding extra nodes to the DOM.

+ Map in Javascript Method

numbers.map((number) ⇒ (number*2))

## * Props in JSX

Props (properties) are a way to pass data from a parent component to a child component.

## * Spread operator

{...arr}

↑ Basically Spread array or objects

# Conditional renders

1. if/else statement
2. Ternary operator
3. && operator
4. switch statement

* Hoisting

```
function foo(){
    console.log("  ")
}
```

This is function declaration and it can be accessed anywhere in the file. Basically function declaration supports file Hoisting.

```
const bar = () => {
    console.log(  );
};
```

above is function expression. and Hoising is not available for function declaration.

## Generator

Generators function can be paused at any point of time.

```
function * foo() {
    yeild 10;
    yeild 20;
}
```

This function will written generators object.

* To access the object

```
const result = foo();
console.log ( foo.next(); // 10
console.log(foo.next()); // 20
```

## Object → array

```
Const person = {
    name : "Ram",
    age : 30
};
```

```
Object.entries(person);
```

o/p ⇒ `[["name", "ram"], ["age": "30"]]`

## dangerouslySetInnerHTML

Convert string HTML to properly HTML so react can render it.

ex
```
Const data = "<h1 style='color: blue:'>
        some really useful content </h1>";

export default function App() {
return <div dangerouslySetInnerHTML =
    {{ __html: data }} />;
}
```

it exposes usy to cross site attacks:

* Implement

Vmit on 12