React

to run → npm start

- is Javascript library for building user interfaces

⊙ index.html is main file which contain root div
Everything we make is inserted inside this element.

in codesandbox to add dependencies
just click at left side & add dependencies

1) JSX and Babel

to use dependencies first install them
& to import
→ either import React from "react";
import ReactDOM from "react-dom"; → index.js

बुधवार 12

or

```
var React = require("react");
var ReactDOM = require("react-dom");
```

→ ReactDOM.render(What to show, where to show);

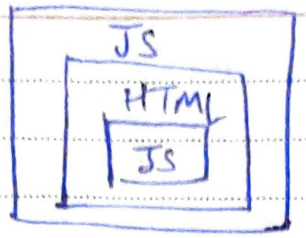
eg
ReactDOM.render(<h1>Hello World</h1>, document.getElementById('root'));
↑ JSX file ↑ inside

at ~~what~~ back what happen is that dependency react has a compiler called Babel which compile HTML inside js.

⊙ render can take singl HTML Element
To use more than one enclose them in one

* In JSX all

with JSX →




```
const name = "Palash";
ReactDOM.render (<h1> Hello {name} </h1>,
                 document.getElementById("react"));
```

* To variable use → { }
can use expression also
{num+5}

or { Math.floor (Math.random() * 10) } </p>

14 शुक्रवार

here

Template literals

<h1> Hello {`${fname} ${lname}`} ! </h1>

or <h1> Hello {fname + " " + lname} ! </h1>

* to add style in all as normal do make css in styles.css like normal x {

header { }

Add Styling

style in like style.css & to use it import it as top.

Also all properties here becomes camel case

* & class becomes className.

contenteditable → contentEditable

full code

```
import "./Styles.css";
var React = require ("react");
var ReactDOM = require ("react-dom");

ReactDOM.render (
  <div>
  <p className="header"> Food </p>
  <ul>
    <li> Dal Batti </li>
    <li> pav Bhaji </li>
  </ul>
  </div>,
  document.getElementById("root")
);
```

⊙ cant add style in <Header className="heading"/>
but do
add in ./Header file

inline style

can't do → can't accept string so convert to object
`<h1 style = "color : red" >`

eg: { color : "red" } → object
& to insert object we need {{}}
so `<h1 style = {{color : "red"}} >`
- `{` → to embed an object
- `{color : "red"}` → JS object

eg2:
```
const custom={                    // JS object
  color : "red" ,                 ← as JS object
  fontSize : "20px" ,             font-size → fontSize
  border : "1px solid black"
};
```

18 मंगलवार

`<h1 style = {custom} > Hello </h1>`

adding if else

eg
```
var date= new Date () ;   var x = date.getHours()
var header ;
  if (x<12)
      header = <h1> Morning </h1>
  else
      header = <h1> Afternoon </h1>
ReactDOM.render (
  <div> {header} </div> ,
  document.getElementById ("root") ;
);
```

- for reference → airbnb [javascript
- Function Name should begin with capital Letter.

React Components

- inside src make file → Heading.jsx
- first letter of component must be capital

Heading.jsx →

```
import React from 'react';

function Heading () {
    return <h1> Fav food </h1>;
}
export default Heading;
```

& in index.js
import Heading from "./Heading"
& use as
<Heading />

↑ Self closing

Best Structure

make file App.js & put all this content there

App.jsx →

```
import React from 'react';
import Heading from './Heading';
import List from './List';
function App() {
    return ( <div>
        <Heading />
        <List /> </div> );
}
export default App;
```

in index.js → `import App from "./App";`
`ReactDOM.render( <App/>, document.getElementById("root") );`

Import, export

- to export var,

```
const pi = 3.1415962;
export default pi;
```

→ to use
```
import pi from "./pi";
```
to use {pi} as variable.

- to export more than 1 things →

eg
```
const pi = 3.141592;
function doublePi () {
    return pi * 2;
}
function triplePi () {
    return pi * 3;
}
```

same name
```
import pi, {doublePi, triplePi}
    from "./pi";
```

to use
```
<li> {pi} </li>
<li> {doublePi()} </li>
<li> {triplePi()} </li>
```
as is fn

```
export default pi;
export {doublePi, triplePi};
```
→ only 1 default

22 शनिवार

can do
```
import * as pi from './pi';

<li> {pi.defaullt} </li>
<li> {pi.doublePi()} </li>
<li> {pi.triplePi()} </li>
```

can make
```
<li> {add (1,2)} </li>
```
→
```
function add (n1, n2)
{
    return n1+n2;
}
```

(directly send arguments)

⊙ For class component → this.props.data
• for function component → props.data

2019 जुलाई

React Props

⊙ Props are like arguments passed in a function. They helps to make components reusable.

eg card component

```
function Card (props) {
  return (                    → can return only 1 thing so all inside div
    <div>
      <h2> {props.name} </h2>
      <img
        src = {prop.img}
        alt = "avatar-img"
      />
      <p> {props.tel} </p>
      <p> {props.email} </p>
    </div>
  );
}
```

to call

```
<Card name="Beyonce" img="img.png" tel="2875r43"
      email="beyonce@gmail.com" />
```

this data is passed within files also

सोमवार 24

Note components using class

```
→ class Heading extends React.Component {
    render() {
      return (
        <h1> Hi </h1>
      );
    }
  }
```

Note → src → App.jsx → to use import Name from '../Name'

Name.js Sourside folder

MAP function JS

it iterates over all objed defined inside it

eg

```
const contacts = [
    {
      id: 1,
      name: 'Beyonce'
    }
    { id: 2,
      name: 'John Morris
    }
    { id: 3;
      name: 'Shakira'
    }
```

26 बुधवार

earlier we do
contact[0].id
contact[0].name

then contact[1].id
contact[1].name

but now use map

```
function createCard (contact) {          → callback fn
return (
  < Card   id = { contact.id}
           name = { contact.name }
    />
);
}
{ inside App() {
    { contacts.map( createCard) }
```

as fn so inside { }

Note

Note all components should have some unique Key

```
< Card
  → Key = { contact.id }
  />
```

MAP, filter, Reduce

⊙ Map → Create a new array by doing something with each item in an array.

```
var numbers = [3, 56, 2, 48, 5];
function double (x) {
    return x*2; }

numbers.map(double);   → print it → [6, 112, 4, 176, 10]
```

Can even do by (for each) →

```
var newNum = []
numbers.forEach(double);

function double(x) {
    newNum.push(x*2);
}
```

शुक्रवार 28

- can even do by anonymous function (without name)

```
numbers.map(function(x)
    return x*2; });
```

⊙ Filter → Create a new array by keeping items that return true

```
const newNum = numbers.filter(function(x) {
    return x > 10.
});
```

⊙ reduce = Accumlate a value by doing something for each item in an array.

eg sum all num

holds ans ↓

```
var x = numbers.reduce(function(accumulator, currentNumber)
{
  return accumulator + currentNumber;
})
```

⊙ find → find first item that matches from an array

```
numbers.find(function(num){
    return num > 10;
})
```

→ return 1st item > 10

30 रविवार

with arrow function

Can be used in place of anonymous functi

```
const newNum = numbers.map((n) => n*n);
```

If else, ternary operator

⊙ can do
```
function renderCondition (){
  if (isLoggedIn === true)
      return <h1> Hello </h1>
  else
      return <Login />
}
& function App() {
    return <div> { renderConditional() } </div>
}
```
↑ so will be call

but can add if else inside this { }
as if else, while, switch are statements not expressions
(expression return at once)

मंगलवार

& to solve this problem → ternary operator

→
```
function App() {
  return <div>
    { isLoggedIn ? <h1> Hello </h1> : <Login /> }
  </div>
}
```

Can use && and || here

⊙ if no value to pass use null here.

To bind this to class function toggle

this.toggle = this.toggle.bind(this);

जुलाई 2019

3 बुधवार

State in React

UI = f(state)

eg todo list when variable isdone = true strike it

```
do
var isDone = false;

const strikeThrough = { textDecoration: "line-through" };

return <p style={isDone ? strikeThrough : null}>Buy Milk</p>;
```

eg Buy milk when isdone = true

↑ So UI dependent on variable isDone

4 गुरुवार

⊙ This programming is declarative programming

When we do → this is Imperative

```
function strike() {
  document.getElementById("root").style.textDecoration = "line-through";
}

function App() {
  return (
    <div>
      <p>Buy milk</p>
      <button onClick={strike}>change strike</button>
    </div>
  );
}

export default App;
```

React hooks

use to bring change in state

→ ~~let~~ var count = 0

```
function increase () { count++; }
```

0
1

```
ReactDOM.render (
 <div> <h1> { count } </h1>
<button onClick = {increase} > + <button
</div>,
 document.getElementById("root");
);
```

⇒ but now also in screen it not changes

State should be passed as component only

- in App.JS

```
import React, { useState } from "react";
function App () {
 const [count , setCount ] = useState(0);

 function increase () {
        ~~return~~ setCount ( count + 1) ; }

return { <div> <h1> { count } </h1>
  <button onClick = { increase }>+</button>
  </div>
};
}
    export default App;
```

शनिवार

* State & fn be made inside App() & before return

explanation

1) either do React.useState or import it
2) useState returns array of size 2 → [variable, function]
3) to destruct array we do
   [count, setcount] = useState(0) → var = 0.
4) this setCount can renders change in variable to screen

5) count ++ replaces count
   can even do new replace time ;
   eg
   [time, setTime] = useState(now);
   & setTime(newTime) ;
   ↓ no need to do time = newTime X

8 सोमवार

OBJECT & Array Destructing

eg const animal = ["cat", "dog"];
to destructure
const [a, b] = animals ;
so a = "cat" , b = "dog"

```
const animals = [
    {name: "cat", sound = "meow" },
    { name: "dog", sound = "woof" },
];
```

new const [cat, dog] = animals;
console.log(cat) → {name: "cat", sound: "meow"}
→ return object

const {name, sound} = cat; (destructing)
console.log(sound) → meow. (nesting)

→ for destructuring object → { }
for destructuring array → [ ]

for array [,] = animals[0], [1]
names → need not be same
but for object name & sound → name, snd
Key names be same

बुधवार

- Nested object

```
const animals = [
  {
    name: "cat",
    sound: "meow",
    feeding: { food: 2, water: 3 }
  },
]
```

→ const [cat] = animals;

→ const {name, sound, feeding} = cat;

console.log(feeding.food) or

const {name, sound, feeding: {food, water}} = cat;

* in inline style can't do {{ color : { } }} / var ...

(*) make state & function inside App() fn & before return

Event handling

- here onclick becomes → onClick

```
impost React ,{useState } from "react" ;
Function App(){

  const [headText, setHead ] = useState ("Hello");
const [isMouseOver, setMouseOver] = useState (false);

function handleClick(){
   SetHead ("Submitted"); }
function handleMouseOver () {
    SetMouseOver (true); }
```

12 शुक्रवार

```
    return (
      <div className ="Container">
    <h1> { headText } </h1>
    <button
      style={{ backgroundColor : isMouseOver? "black" : "white" }}
      onClick = { handleClick }
      onMouseOver = { handleMouseOver } > Submit </button>
    </div>
    );
  }
  export default App;
```

→ can add onMouseOut = { handleMouseOut }

```
function handleMouseOut () {
    SetMouseOver (false);
}
```

Forms

see control or class component

Form during submission refresh themselves

```
function App() {
const [name, setName] = useState("");
const [headText, setHead] = useState("");

function handleChange(event) {
 setName(event.target.value); }

function handleClick(event) {
  setHead(name);
  event.preventDefault();
}
```

* name variable of another state can be directly use in other state

// prevent form reload

```
return(
   <div className="container">
   <h1> Hello {headText} </h1>
   <form onSubmit={handleClick}>
   <input
      onChange={handleChange}
      type="text"
      placeholder="What is your name?"
      value={name}
      />
   <button type="submit"> Submit </button>
   </form>
   );
}
```

* if on form so handle both Enter & submit button

रविवार 14

⊙ Class Components vs Functional components

```
import React, {Components} from "react";

class App extends React.Component {
    render() {
        return <h1> Hello </h1>
    }
}
    export default App;
```

```
import React from "react";

function App() {
    return <h1> Hello </h1>;
}
export default App;
```

⇒ We prefer functional components. We use to use class components before hooks concept was introduce to use state.
but now after hooks use functional components (best way).

Changing Complex State

means retrive old data & add new one also eg

Hello Palash Bajpai

Palash

Bajpai

Submit

one way can do

Hello {fname} {lname}

& write 2 function & 2 input

Capo also do

```
function App() {
    const [fullName, setFullName] = useState({
      fName = "",
      lName = ""
    });
function handleChange(event) {
    const newValue = event.target.value;
  setFullName(prevValue => {
    if (event.target.name === "fname") {
      return(
          fName : value,
          lName : prevValue.lName
          );
```

```
      else if (name === "lname") {
      return {
          fname : prevValue.fName,
          lname : value };
      }
    });
  }
& [h1> Hello {fullName.fName} {fullName.lName} </h1>

& <input  name = "fName"
          onChange = {handleChange}
          placeholder = "first Name"
  />
```

Spread operator (...)

to copy array a in b

eg a = ["apple", "bannana"]
b = ["peach", "kiwi", ...a]

so b = ["peach", "kiwi", "apple", "bannana"]

can also use for objects

- Hence this can be used to store previous Value

eg Todo list

```
function App() {

const [inputText, setInputText] = useState("");
const [items, setItems] = useState([]);

function handleChange(event) {
    const newValue = event.target.value;
    setInputText(newValue);
}

function addItem() {
    setItems(prevItems => {
        return [...prevItems, inputText];
    });
    setInputText("");
}
```

```
return (
  <div className="container">
  <div className='heading'>
  <h1> To-Do List </h1>
  </div>
  <div className="form" >
  <input onChange={handleChange} type='text"
                value={inputText} />
<button onClick={addItem}>
    <span> Add </span>
</button>
 </div>
  <div>
   <ul> { items.map(todoItem => {
              <li> {todoItem} </li>
          ))}
     </ul>
   </div>
   </div>
  );
}
```

State in class Component

(also called stateless component)

```
class StatefulComp extends React.Component
{
  constructor (props){
    super (props);

  this.state ={
    name : "Palash",
  }
  }
  render () {
    return (
      <div>
      <h1>{this.state.name}</h1>
      </div>
    );
  }
};
```

"To bring changes before render

```
handleClick () {
this.setState({
  name : "React Rocks!"
  });
}
```

```
<button onClick = {
  this.handleClick}>
  click me </button>
```

24 बुधवार

To change state according to previous state

```
this.setState ([state, props) => ({
    counter : state.counter + props.increment
}));
```

or

```
this.setState (state => ({
    counter : state.counter + 1
}));
```