1. react :
   a) JS lirary to make web and native application
   b) follows componet based arch
   c) declarative (most of built in code)
   d) provide virtual DOM
   e) we can add REACT any point of time in our project.
2. Understanding textContent, innerHTML and createElement

```
<body>
   <h1 id="my_h1">My text will change, not tag</h1>
   <div id="my_div"> <p>I will vanish and h2 tag will appear</p> </div>
   <div id="root">
      <p>you will find h3 tag inside this div</p>
   </div>
   <script>
      //text content
      let my_h1=document.getElementById("my_h1")
      my_h1.textContent = "HELLO REACT"

      //innerHTML
      let my_div=document.getElementById("my_div")
      my_div.innerHTML = "<h2>Hello React</h2>"

      //child node creation
      let root=document.getElementById("root")

      let my_h3=document.createElement("h3")
      my_h3.textContent = "new h3 tag with some content"

      root.appendChild(my_h3)
   </script>
</body>
```

3. Creating element with react

```
<body>
   <div id="root"></div>
   <script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>
   <script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>
   <script>
      //creating element using react
      let my_h1=React.createElement("h1",null,"this h1 is created using React")

      //getting div by its id to load above element inside it
      let root = document.getElementById("root")

      //rendering/displaying created element into div
      ReactDOM.render(my_h1,root)
   </script>
</body>
```

4. After running above code, check console, you will get error because of ReactDOM.render()

```
      //rendering/displaying created element into div
      //ReactDOM.render(my_h1,root) //this will gives erroc coz, ReactDOM.render() is not suppoerted in R17

      let reactRoot=ReactDOM.createRoot(root)
      reactRoot.render(my_h1)

   </script>
```

5. Creating component using react
   a. We cant render/display multiple elements at a time using reactRoot.render() so we need to create component. Component is a collection of elements. We can create either class based or function based component.
   b. Component name must be start with Capital letter

```
<body>
   <div id="root"></div>
   <script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>
   <script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>
   <script>
      //creating element using react
      function MyComponent()
      {
         return React.createElement("h1",null,"this h1 is created inside component using React")

      }
      //getting div by its id to load above element inside it
      let root = document.getElementById("root")

      let reactRoot=ReactDOM.createRoot(root)
      //rendering component inside reactRoot
      reactRoot.render(React.createElement(MyComponent))

   </script>
</body>
```

6. JSX:

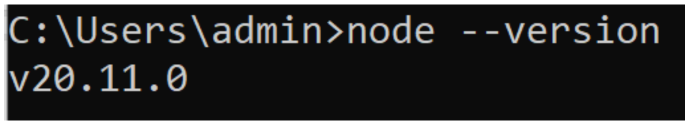
   const element = <h1>Hello, world!</h1>;

   a. This funny tag syntax is neither a string nor HTML.
   b. It is called JSX, and it is a syntax extension to JavaScript. We recommend using it with React to describe what the UI should look like. JSX may remind you of a template language, but it comes with the full power of JavaScript.

   c. JSX produces React “elements”. We will explore rendering them to the DOM in the next section. Below, you can find the basics of JSX necessary to get you started.
   d. We can use babel library to transpile JSX code onto React code
      i. Get babel cdn : https://babeljs.io/setup#installation then click on in the browser button
      ii. Creating element using JSX and babel

```
<body>
   <div id="root"></div>
   <script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>
   <script crossorigin src="https://unpkg.com/react-dom@18/umd/react-
dom.development.js"></script>

   <!-- adding babel cdn -->
   <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>

   <script type="text/babel">
      function MyComponent()
      {
         return <h1>This h1 is created in JSX</h1>;
      }

      let root = document.getElementById("root")

      let reactRoot=ReactDOM.createRoot(root)
      reactRoot.render( <MyComponent/> )

   </script>
</body>
```

7. Why component name must start with capital letter?

8. Manually converting JSX code into React code
    a. Check node version
        i. 

```
C:\Users\admin>node --version
v20.11.0
```

    b. If not found then download and install node
    c. Create folder
    d. Open that folder in vs code
    e. Create node package manager in it by running
        i. >npm init –y
    f. Install babel
        i. >npm i -D @babel/core @babel/cli @babel/preset-env @babel/preset-react
    g. Create 2 folders src and dist
        i. Create index.js file in src folder and add following code :

```
function MyComponent()
      {
          return <h1>This h1 is created in JSX</h1>;
      }

let root = document.getElementById("root")

let reactRoot=ReactDOM.createRoot(root)
reactRoot.render( <MyComponent/> )
```

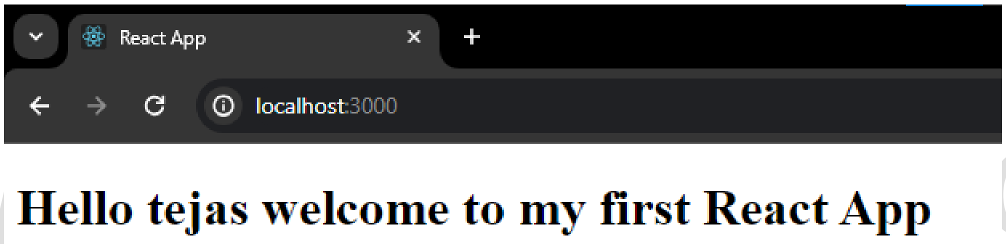
        ii. Open terminal and run following command to convert JSX into React
            > npx babel --watch src --out-dir dist --presets=@babel/preset-env,@babel/preset-react
            By running above command, you will get index.js file in dist folder
        iii. Create index.html in project folder (not in src or dist) and add following code

```
<body>
   <div id="root"></div>
   <script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>
   <script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>
   <!-- error -->
   <!-- <script src="src/index.js"></script> -->

   <!-- no error  -->
   <script src="dist/index.js"></script>
</body>
```

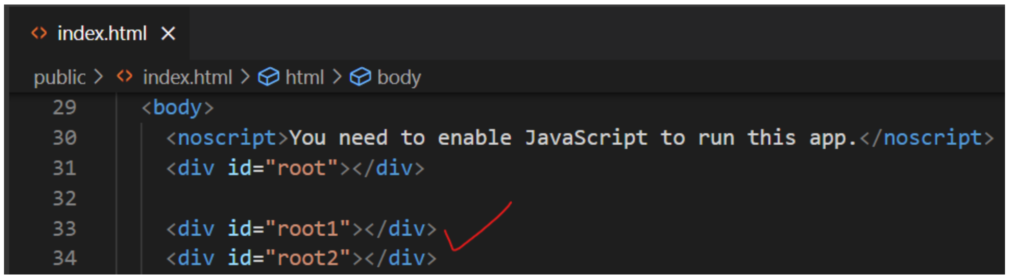
        iv. Run above index.html
        v. DONE !

9. Creating first react app
   a. Open cmd and
      i. >npx create-react-app first-app
      ii. >cd first-app
      iii. …….first-app>code . *to open project in vs code*
      iv. …….first-app>npm start *to start server*
   b. Delete all files from src folder except index.js
   c. Update code inside index.js as –

```
import ReactDOM from 'react-dom/client';

function Display()
{
  let my_name = "tejas"
  return <h1>Hello {my_name} welcome to my first React App</h1>
}
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Display/>);
```



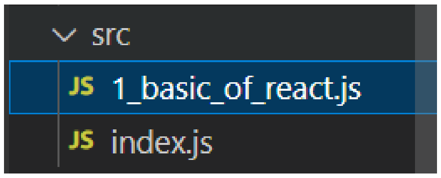
Hello tejas welcome to my first React App

10. Understanding diff between actual and virtual dom
    a. Create 2 div tag with id in index.html as
        i. 

index.html ×

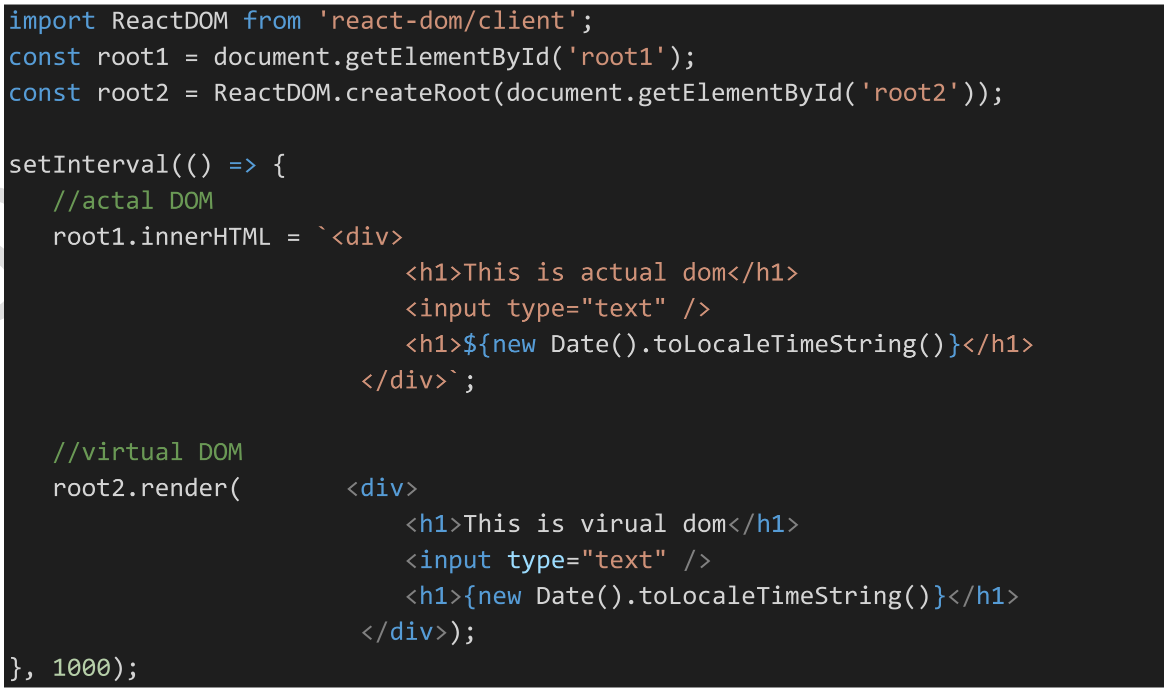
public > index.html > html > body

```html
<body>
  <noscript>You need to enable JavaScript to run this app.</noscript>
  <div id="root"></div>

  <div id="root1"></div>
  <div id="root2"></div>
```

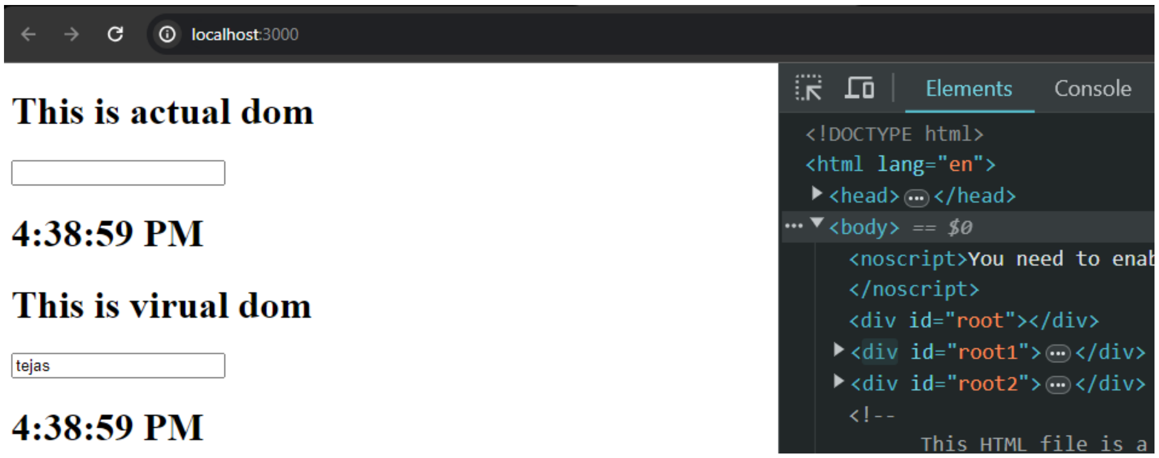
    b. I have created new index.js file and previous code I have saved with name - 1_basic_of_react.js this is because everytime we need index.js as entry point for our project
        i. 

```
src
  1_basic_of_react.js
  index.js
```

    c. Code for index.js to understand diff between actual and virtual DOM



```js
import ReactDOM from 'react-dom/client';
const root1 = document.getElementById('root1');
const root2 = ReactDOM.createRoot(document.getElementById('root2'));

setInterval(() => {
   //actal DOM
   root1.innerHTML = `<div>
                          <h1>This is actual dom</h1>
                          <input type="text" />
                          <h1>${new Date().toLocaleTimeString()}</h1>
                       </div>`;

   //virtual DOM
   root2.render(       <div>
                          <h1>This is viral dom</h1>
                          <input type="text" />
                          <h1>{new Date().toLocaleTimeString()}</h1>
                       </div>);
}, 1000);
```



localhost:3000

This is actual dom

4:38:59 PM

This is virual dom

tejas

4:38:59 PM

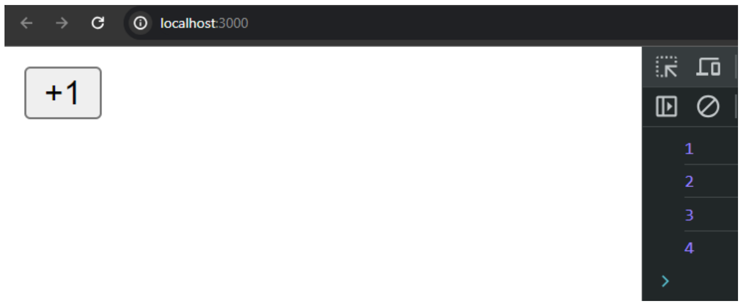
```html
<!DOCTYPE html>
<html lang="en">
 ▶<head>…</head>
…▼<body> == $0
    <noscript>You need to enat
    </noscript>
    <div id="root"></div>
   ▶<div id="root1">…</div>
   ▶<div id="root2">…</div>
    <!--
        This HTML file is a
```

11. Event handling in react (button click)
    a. Create index.js and add following code :

```
import ReactDOM from 'react-dom/client';

const generate_random = () =>{
   console.log(Math.random());
}

function Button()
{
  //html code to handle onclick on button : <button onclick="demo()"></button>
  //jsx code to handle button :

  //return <button onClick={generate_random}>click me to get random number</button>

  return <button onClick={()=>{
                                     console.log(Math.random());
                                  }}>click me to get random number</button>
}
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Button/>);
```

12. Counter app (display on console)
    a. Don't create new index.js file just remove above code and add following

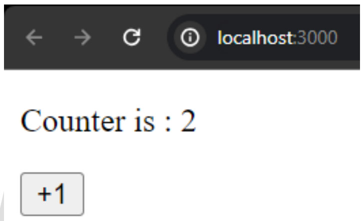
```
import ReactDOM from 'react-dom/client';
function Button()
{
   let counter = 0

   const setCounter = () =>{
                                counter+=1
                                console.log(counter);
                             }
   return <button onClick={setCounter}>+1</button>
}
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Button/>);
```



```
localhost:3000
+1
1
2
3
4
```

13. Counter app (display on paragraph tag)
    a. Here we will understand useState() hook (Don't create new index.js file just remove above code and add following)

```
import { useState } from 'react';
import ReactDOM from 'react-dom/client';
function Button()
{
   let [counter,setCounter]=useState(0)

   const updateCounter = ()=>{
                                 setCounter(counter+1)
                             }
   return <div>
            <p>Counter is : {counter}</p>
            <button onClick={updateCounter}>+1</button>
         </div>
}
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Button/>);
```



Counter is : 2

14. Displaying multiple components at a time
    a. Create new index.js file and save pervious code.

```
import React from 'react';
import ReactDOM from 'react-dom/client';

function Button()
{
   return <button>Click me</button>
}

function Display()
{
   const my_name = "Tejas"
  return <h1>Hello {my_name} </h1>
}
const root = ReactDOM.createRoot(document.getElementById('root'));
// root.render([<Button/>,<Display/>]);

// root.render( <div> <Button/> <Display/> </div> );

// root.render(<React.Fragment> <Button/> <Display/> </React.Fragment>);

root.render(
   <>
      <Button/>,<Display/>
   </>
);
```

15. Parent component that return other components (child components)
    a. Create new index.js and add following code

```
import React from 'react';
import ReactDOM from 'react-dom/client';

function Button()
{
   return <button>Click me</button>
}

function Display()
{
   const my_name = "Tejas"
  return <h1>Hello {my_name} </h1>
}

function App()
{
   return <>
      <Button/>
      <Display/>
   </>
}
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<App/>);
```

16. Passing data between components
    a. Create new index.js

```
import React, { useState } from 'react';
import ReactDOM from 'react-dom/client';

function Button(props)
{
   return <button onClick={props.function_call}>Click me</button>
}

function Display(props)
{
   const counter_value = props.data
  return <h1>Counter is :  {counter_value} </h1>
}

function App()
{
   let [counter,setCounter]=useState(0)

   const updateCounter = ()=>{
                                setCounter(counter+1)
                            }
   return <>
      <Display data={counter}/>
      <Button function_call={updateCounter}/>
   </>
}
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<App/>);
```

17. Creating new file for each component