React:

1.props in functional component

```
//props in function

import React from 'react';
//like this you can also create class component
// export   we can export this fucntin itself for that see line 10 in App.js

// remember : har jagah variable ko {} me daal ke use karna hai or else wo
string

const Header = (props) => {
    console.log(props);   //you can see in console that we are receiving it
    //props ki value change nhi hoti they are immutable
    return (
        <div>
            <h1>This is {props.name} {props.lastname} </h1>
            {props.children}
        </div>

    )
}

//this line is necessary
export default Header;
```

props in class component

```
//props in class component
import React from "react";

class CHeader extends React.Component{
    render(){
        return <h1>Hello Class {this.props.name}</h1>
    }
}

export default CHeader;
```

state in class component

```
//State in class component

import React from "react";

class Channel extends React.Component{
    constructor(){   //constructor is used to allocate memmory to the object
```

```
        super();     //accesses the constructor of parent ie React component
        //NOW WE WILL MSKE STATE HERE
        this.state={               //accesing state object{}
            msg:"we are using State here"
        }
    }

    call=()=>{
        this.setState({            //remember to use
this.setState({property:value});
            msg:"hence state changed"
        })
    }
    render(){
        return(
            <div>{this.state.msg}
            <button onClick={this.call}>Change state</button>
            </div>
        )
    }
}

export default Channel;
```

2.Forms and events

Forms

```
import React from "react";


class Form extends React.Component{
    constructor(props){
        super(props);
        this.state={
            username:"",
            address:""
        }
    }
    handleUsername=(event)=>{
        this.setState({
            username:event.target.value
        });
    }

    handleAddress=(event)=>{
```

```
        this.setState({    //dont forget () here
            address:event.target.value
        })
    }
    handlebutton=(event)=>{
        event.preventDefault();
        alert(`${this.state.username} ${this.state.address}`);
    }
    render(){return(
        <form>
            <div>
            <label for="name">Username</label>
            <input type="text"  value={this.state.username}
onChange={this.handleUsername}></input>
            </div>
            <div>

            <label>Enter address</label>
            <input type="textarea" value={this.state.address}
onChange={this.handleAddress}></input>
            </div>

            <button onClick={this.handlebutton}>Click to save</button>
        </form>
    );}
}

export default Form;
```

Events:

Functional

```
//event handling in fucntional component


import React from "react";
//fucntinal component me this access nhi hota

const Eventhandling=()=>{
    const FunctionClick=()=>{
        console.log("working");
    }
    // use function instead of using function() because we are not calling
fucntion but using it
    return(
        <div><button onClick={FunctionClick}>Click me</button></div>
```

```
    )
}

export default Eventhandling;
```

class

```
//eventhandling in class

import React from "react"

//using props in class components
class CEventhandling extends React.Component{
    constructor(props){           //pass props if using props
        super(props);                //pass props if using props

        this.state={                          //remember use = here
            msg:"Hello boi",
            roll:this.props.roll
        })

    }
    //only difference in onclick this.fucntion is used
    FunctionClick=()=>{
        this.setState({
            msg:"hello gurls"
        })
        console.log("working in class component",this.state.roll);
    }
    // use function instead of using function() because we are not calling
fucntion but using it
    render(){return(
        <div><h1>{this.state.msg}  {this.state.roll}</h1><button
onClick={this.FunctionClick}>Click Class Component</button></div>
    )
    }
}

export default CEventhandling;
```

3.router

```
Router
use npm i -D react-router-dom first

App.js Code:
import './App.css';
```

```jsx
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Layout from "./Layout";
import Home from "./Home";
import Blogs from "./Blogs";
import Contact from "./Contact";
import NoPage from "./NoPage";

function App() {
  return (
    // <div className="App">
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Layout />}>
          <Route index element={<Home />} />
          <Route path="blogs" element={<Blogs />} />
          <Route path="contact" element={<Contact />} />
          <Route path="*" element={<NoPage />} />
        </Route>
      </Routes>
    </BrowserRouter>
    //</div>
  )
}
export default App;
```

Layout.js code:

```jsx
import { Outlet, Link } from "react-router-dom";
const Layout = () => {
  return (
    <>
      <nav>
        <ul>
          <li>
            <Link to="/">Home</Link>
          </li>
          <li>
            <Link to="/blogs">Blogs</Link>
          </li>
          <li>
            <Link to="/contact">Contact</Link>
          </li>
        </ul>
      </nav>

      <Outlet />
    </>
  )
```

```
};
export default Layout;

Home.js code:
import React from 'react'

const Home = () => {
  return (
    <div>
      <h1>This is Home</h1>
    </div>
  )
}
export default Home

Blogs.js code:
import React from 'react'
const Blogs = () => {
  return (
    <div>
      <h1>This is Blogs</h1>
    </div>
  )
}
export default Blogs
```

9.Repl use node and write

```
10. const fs=require('fs');


fs.readFile("happy.txt",(err,data)=>{
    if(err) console.log("error :"+ err);
    else console.log(data.toString());
});

//this overwrites the data
fs.writeFile("index.txt","hello world",(err,data)=>{
    if(!err) console.log("Success");
    else console.log("error");
});

//appends the data
fs.appendFile("index.txt","\n this is appended ",(err,data)=>{
    if(!err){
        console.log("success");
```

```
    }
    else console.log(err);
})

fs.rename("index.txt","happy.txt",(err,data)=>{
    if(!err){
        console.log("done");
    }
});

fs.open('index.txt', 'w', function (err, file) {
    if (err) console.log("error");
    console.log('Saved!');
  });

fs.unlink("index.txt",(data,err)=>{
    if(err) console.log("error");
    console.log("saved");
})
```

DOM:

```
//document object model
//here everythign in document can be treated as object for example the window
obeject

console.log(window);

//window.alert("how are you"); //so alert is a part of wndow object but its
not necessary to use window object here only alert(something) will also do


// document is used to select things/elements from document

//single elements

// for selecting single element use

console.log(document.getElementById("my-form"));
const form =document.getElementById("my-form");   //assinging it to a variable
console.log(form);

// jqury :javascript library that helps select other thing than id likes
classes,tags etc;

// queryselector works same as jquery meaning we can also select classes,tag
wtih it
//use dot for class ".class" and # for id "#id"
```

```
console.log(document.querySelector(".container"));   //imp:use dot here
console.log(document.querySelector("h1"));   //Recomended //select only one h1
and its the first one

//Multiple items
console.log(document.querySelectorAll(".item")); //Recomended  //gives us
array like nodelist and we can run array methods on it like for each

console.log(document.getElementsByClassName("item")); // here no dot is use
because it only grabs classes //it gives us html collection which should be
converted to array unlike nodelist


const items=document.querySelectorAll(".item");

items.forEach((x)=>console.log(x));  //here we can traverse throudh entire
list

const ul=document.querySelector(".items");

//ul.remove();   //this method removes the entire ul
// ul.firstElementChild.remove();     // this remove the first li
ul.firstElementChild.textContent="umar";  //we can also dyanamically change
text
ul.children[1].innerText="Shaikh";
ul.lastElementChild.innerHTML="<h1>hello<h1>";  //we can also dynamically add
html


//Styling

const btn=document.querySelector(".btn");

btn.style.background="red";


// with these all things we can have events in functions and can dyanamically
change things

//----------------------------------Events-------------------------------------
-------

// somepara.addEventListener("someevetn",function with parameter e (event))

btn.addEventListener("click",(e)=>{
    e.preventDefault();     //calling this method because....
    // console.log("click");    //...as this flashes really fast so we use
above statement
```

```
    console.log(e);  //Prints entire event object but in that 'target' is very
important
    console.log(e.target);  //prints entire tag
    console.log(e.target.className);   // prints class associated with that
tag
    //like this you can get a lot of different attributes
});


btn.addEventListener("click",(e)=>{
    e.preventDefault();
    document.querySelector("#my-form").style.background="#ccc";
    // adding a class to a tag after clicking button
    document.querySelector("body").classList.add("bg-
dark");   //.remove("class") can also be used;
    document.querySelector(".items").lastElementChild.innerHTML="<h1>New
Look</h1>";
});

btn.addEventListener("mouseover",(e)=>{
    e.preventDefault();
    document.querySelector(".items").firstElementChild.innerText="Changed on
hover";
});

/*
btn.addEventListener("mouseout",(e)=>{
    document.querySelector(".items").firstElementChild.innerText="Changed out
on hover";
});
*/

const button=document.querySelector(".btn");

button.style.background="red";

ul.firstElementChild.innerText="hello";
ul.firstElementChild.innerHTML="hello";
```

Promises,fetch,callback

Callback

```
let Posts=[
    {
        title:"title1",
        body:"body1"
    },
```

```
    {
        title:"title2",
        body:"body2"
    }
];

function getPosts(){

    setTimeout(() => {
        let output="";
        Posts.forEach((post)=>
        {
            output+=`<li>${post.title}</li><p>${post.body}</p>`;
        })
    document.querySelector("body").innerHTML=output;
    }, 500);
}
/* ------------------------------------------------
createPost=(post)=>{

    Posts.push(post); //this doesnot wait for setTimeout

    setTimeout(() => {     //this doesnot get print because our dom was
already painted since getPosts executed time is less than this
        Posts.push(post);
    }, 1000);
}

getPosts();
createPost({tile:"post3",body:"body"});
------------------------------------------------- */

// so if make getposts a callback and insert it in Createpost as parameter to
occur then...

createPost=(post,callback)=>{
    setTimeout(() => {
        Posts.push(post);
        callback();      //calling getposts here
    }, 3000);
}
//so all three will be executed

createPost({title:"title3",body:"body3"},getPosts)
```

Promise

```
let Posts = [
    {
        title: "title1",
        body: "body1"
    },
    {
        title: "title2",
        body: "body2"
    }
];

getPosts = () => {
    let output = "";
    setTimeout(
        () => {
            Posts.forEach((post) => {
                output += `<li>${post.title}</li>`;
            });
            document.body.innerHTML = output;
        }, 1000);

}

function createPost (post) {
    return new Promise((resolve, reject) => {     //returning a promise, which
will be evaluated by .then and then callback will execute see below
        setTimeout(() => {
            Posts.push(post);
            let error = false;
            if (!error) {
                resolve();
            }
            else reject("SOmething went wrong!!");
        }, 2000);

    });
}
//promise.then*
createPost({ title: "title3", body: "body3"
}).then(getPosts).catch(err=>console.log(err));

/* syntax of promis.then
myPromise.then(
    function(value) {myDisplayer(value);},
    function(error) {myDisplayer(error);}
  );
  */
```

```
  //since it doesnt return in json format we mapped the response to json using
then
const
promise1=fetch("https://jsonplaceholder.typicode.com/users").then(res=>res.jso
n());

promise1.then((values)=>console.log(values));

// promise1.all(promise1,promise2,promise3...).then(somefucntion);

//ASYNC

// add async before fucntion if you want to use await

async function init(){
    await createPost({ title: "title4", body: "body4" }); //here await waits
for create post to occur and then below getposts is executed
    getPosts();
}

init();


//async  /await /fetch

// much cleaner way to handle promises without using .then
async function Getusers(){
    let users=await fetch("https://jsonplaceholder.typicode.com/users");

    let data=await users.json();

    console.log(data);
}

Getusers();
```

Loops

```
//----------------------------------------LOOPS-------------------------------
----------------//

for(let i=0;i<10;i++){
    console.log(`for loop Number ${i}`);     //remember using ticks instead of
comma
}

let i=10;
while(i>=0){
```

```
    console.log(`While loop no ${i}`);
    i--;
}

// iterating through an array

for(let i=0;i<todoes.length;i++){    //remember we can use length property for
string as well as array
    console.log(todoes[i].task);
}

for(let x of todoes){                 //same as for(auto x:arr)
    console.log(x.task);
}

// ----------------------------------------FOREACH_LOOP---------------------
--------------------------

// we use another loop for array which contain a function as parameter
// function passed to a functin is called as callback function

// syntax:   array.forEach(function(x){ console.log(x.property) });

todoes.forEach(function(x){    //this looks better with array function
    console.log(x.id);
});

// ----------------------------------------MAP-----------------------------
--------------------------

// map returns a particular property of array as an array, eg returning all
tasks property of todoes as array

// since it returns an array we assign it to a varaible

//syntax:  const arr=array.map(function(x){ return x.property;});

const todotask=todoes.map(function(x){
    return x.task;
});

console.log(todotask);    //gives array of task property of all objects


//-------------------------------------------------FILTER-------------------
----------------------

// if we want to filter some property based on values
// eg filtering iscomleted based on true then we use filter
```

```
//syntax: const arr=array.filter(function(x){return x.property==value; });

// remember == is to match two variables and === is to match variables as well as with datatype

const taskCompletedobject=todoes.filter(function(x){
    return x.isCompleted==true;
});

console.log(taskCompletedobject);   //gives array of objects which has iscompleted property as true value

// to have only a property instead of printing entire object just map that filter

const taskCompleted=todoes.filter(function(x){
    return x.isCompleted==true;
}).map(function(x){
    return x.task;
})

console.log(taskCompleted);     //prints only task property which have isCompleted property as true value, instead of entire object
```

Iterator and generator

```
function* fibo(){
    let a=0;
    let b=1;
    while(true){
        yield a;
        let temp=a;
        a=b;
        b=b+temp;
    }
}
const sequence=fibo();
console.log(sequence.next().value);
console.log(sequence.next().value);
console.log(sequence.next().value);
console.log(sequence.next().value);
console.log(sequence.next().value);
console.log(sequence.next().value);
console.log(sequence.next().value);
console.log(sequence.next().value);
```

```
const favouratesMovies=[
    "hello",
    "next"
]

const iterator=favouratesMovies[Symbol.iterator]();
```