Promises –

1. Explain what a JavaScript Promise is and describe its primary purpose.

Ans:

Promise is an object which represents the eventual completion or failure of an async operations. Async operations will always returns promise.purpose of promise is to manage async operations I a better way.

2. Describe the states of a Promise and what each state signifies.

Ans:

Promise has 3 states:

1. Pending: initial state

2. Fulfilled: promise solved or successful async operation

3. Rejected: failure

API –

1. Explain what an API (Application Programming Interface) is in the context of web development and how JavaScript interacts with it.

Ans:

API consists of a set of rules and protocols. by which 2 modules or different software systems to communicate with each other. JavaScript interacts with APIs by sending requests (GET, POST, PUT, PATCH, DELETE) to servers and receiving responses that can be used in web applications.

2. Describe the difference between synchronous and asynchronous API calls in JavaScript. Why are asynchronous calls generally preferred?

Ans:

synchronous calls are generally preferred because they improve the performance of the application, preventing it from lock conditions while waiting for a response.

Synchronous API calls: (one-by-one)These calls wait for the operation to complete before moving to the next line of code. This can make the application unresponsive if the call takes a long time.

Asynchronous API calls: These calls do not wait for the operation to complete. Instead, they allow the program to continue running other code while waiting for the response.

3. What is the Fetch API in JavaScript? Provide an example of how to make a GET request using the Fetch API and explain the code.

Ans

GET call is used to retrieve the data from server or API

```
Const API = “http://api.com”;

.fetch(API)

.then(response => response.json)

.then(data=>

{

Console.log(data));

}

.catch(err=>

{

Consol.log(“error”,err);

})
```

Use Effect

1. Explain the purpose of the useEffect hook in React.

Ans:

The useEffect hook in React is used to perform side effects in functional components. Side effects are operations like fetching data, directly updating the DOM, and setting up subscriptions. useEffect helps to manage these operations efficiently within React components.

2. Describe the syntax of the useState hook and explain the meaning of its return value

Ans:

```
useEffect( ( ) => {

//code

} , [ dependencies ] );
```

It fetch the operation when component mounts

[] -> dependency array is same as componentdidmount in react

3. How can you initialize state with the useState hook? Provide an example with a detailed explanation.

Ans:

```
//import usestate from react

Import { useState  }from “react”;

//counter

Export default function Counter()

{

Const [ count, setCount] = userState( 0 );

Function handleclick()

{

Setcount = (count+1);
}

Return(

<>

<p> number : {count} <p/>

<button onclick(handlecount) >click </button>

</>

}
```

4. What is the significance of the setter function returned by the useState hook? Illustrate with an example how you can update the state

Ans:

The setter function returned by the `useState` hook is important because it lets you update the state. When you call this function with a new value, React re-renders the component with the updated state.

```
import { useState } from 'react';

Export default function NameChanger() {

  const [name, setName] = useState('yash');
```

```
const handleChange = (e) => {
  setName(e.target.value);
};

return (
  <div>
    <p>Your name is: {name}</p>
    <input type="text" value={name} onChange={handlechange} />
  </div>
);
}
```

So, the setter function (setName in this case) is crucial for changing the state. Whenever the state changes, React updates the UI to reflect the new state

Files in React app –

1. Explain the typical structure of a React project. What are the main directories and files you would expect to find, and what is the purpose of each?

Ans:

There are many files in react project like public directory,src dierectory,then config files like package.json and package.lock.json, node_modules and some testing files also and reacdme.md

Structure:

my-react-app/

├── node_modules/

├── public/

│ ├── index.html

│ └── favicon.ico

├── src/

│ ├── assets/

│ │ └── logo.png

│ ├── components/

```
│   │   ├── Header.js
│   │   └── Footer.js
│   ├── styles/
│   │   ├── App.css
│   │   └── Header.css
│   ├── App.js
│   ├── index.js
│   └── serviceWorker.js
├── .gitignore
├── package.json
└── README.md
```

public directory:

index.html: The main HTML file. This is where your React app gets injected. It typically contains a element, where the React app will render.

Other assets: This directory can also contain other static assets like images, favicon, and any other files that won't change during the application's runtime.

src directory:

index.js: The entry point for the React application. It renders the main App component into the DOM element with id="root".

App.js: The root component of the application. Other components are typically nested within this component.

App.css: The CSS file for styling the App component.

components directory: This directory contains reusable components. Each component usually has its own file (e.g., Header.js, Footer.js).

styles directory: This directory contains CSS files or styling solutions for the components (e.g., App.css, Header.css).

assets directory: This directory contains images, icons, and other static assets used in the application.

node_modules directory:

This directory contains all the dependencies and packages installed via npm or yarn. You usually don't interact directly with this directory.

Configuration files:

package.json: This file contains metadata about the project, including dependencies, scripts, and other configuration settings. It's essential for managing the project's dependencies and running various scripts.

.gitignore: Lists files and directories that should be ignored by version control systems like Git.

README.md: Provides information about the project, how to set it up, and how to use it. It's useful for anyone looking at the project for the first time.