1. Explain using code examples what is meant by props and state in React JS?

Prop and state are 2 different concepts that allow you to manage data in React

Prop (properties) passes data through parent components to child components. It is read only and shouldn’t be modified by the child component.

function ChildComponent(props) {

return <h1>{props.greeting}</h1>;

}

function ParentComponent() {

return <ChildComponent greeting="Hello, World!" />;

}

In this example Parent passes greeting to Child which is used to display the greeting.

State is a way of maintaining data that can change over time and affect the component's rendering. State is managed within the component and can be changed using the setState function (in class components) or the useState hook.

import React, { useState } from 'react';

function Counter() {

const [count, setCount] = useState(0);

return (

<div>

<p>You clicked {count} times</p>

<button onClick={() => setCount(count + 1)}>

Click me

</button>

</div>

);

}

In this example, the Counter component has a count state variable that's initially set to 0. When the button is clicked, the setCount function is called to increment the count state variable, which causes the component to re-render with the updated count

1. In functional programming, what does the term functor mean? Can you give an example in JavaScript?

In functional programming, a Functor is a type that implements a map method, which applies a function to the value(s) in the functor and returns a new functor. In JavaScript, Arrays are a common example of a functor.

const numbers = [1, 2, 3];

const square = x => x \* x;

const squaredNumbers = numbers.map(square); // [1, 4, 9]

In this example, numbers is an array (functor) of numbers. The map method is used to apply the square function to each number in the array, resulting in a new array (functor) of squared numbers.

1. We have looked at three kinds of asynchronous programming mechanisms, namely callbacks, promises and streams. Mention one advantage and one disadvantage of each type.

Callbacks:

* Advantage: Simple to understand and use for small tasks.
* Disadvantage: Can lead to "callback hell" with nested callbacks, making code hard to read and manage.

Promises:

* Advantage: Makes asynchronous code easier to read and manage by avoiding callback hell.
* Disadvantage: Error handling can be tricky, as errors need to be caught at each step in the chain.

Streams:

* Advantage: Efficient for handling large data, as you don't need to load the entire data into memory.
* Disadvantage: More complex to understand and use, especially handling errors and stream control flow.

1. With the aid of a diagram and example code, describe the Cascading Style Sheets (CSS) Box. Model and show how it can be used to space DOM elements

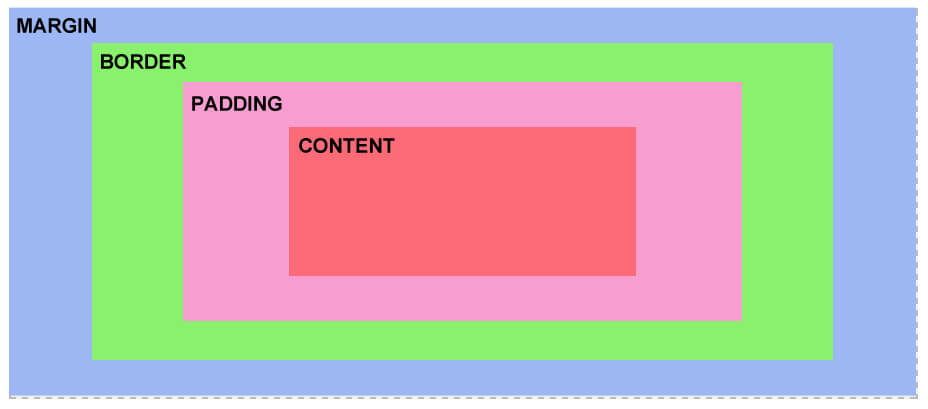
The CSS Box Model is a rectangular layout paradigm for HTML elements that consists of the following areas, from innermost to outermost:

Content: The actual content of the box, where text and images appear.

Padding: Clears an area around the content. The padding is transparent.

Border: A border that goes around the padding and content.

Margin: Clears an area outside the border. The margin is transparent.



example of how you can use the CSS Box Model to space DOM elements

div {

width: 300px;

border: 25px solid green;

padding: 25px;

margin: 25px;

}

In this example, each div will have a content width of 300px, a padding of 25px around the content, a border of 25px around the padding, and a margin of 25px around the border. This will space out the div elements on the page.

1. Detail how the browser loads and bootstraps a rich web application from an initial URL.

When a browser loads from an initial URL, It follows these steps.

DNS Lookup: domain name into IP Address

HTTP Request: browser sends HTTP request to server

Server Response: responds with css, html and javascript

Parse HTML: browser parses html to build DOM

Load Assets: browser loads CSS and Javascript files

Javascript Bootstrapping: If it’s a rich web application, javascript will typically make additional requests to load data (via APIs) or additional app code

Render: browser renders app on screen, in rich web this is usually javascript manipulating the DOM.