**1. Explain using code examples what is meant by props and state in**

**React JS?**

Props and state both influence the render method's output, but they are not the same thing. Props are supplied to a component, whereas state is controlled within each individual component; in other words, props allow data to be sent from one component to another.

State is an updateable structure that can change, for instance, in response to an event that the user initiated. State cannot be changed by another component in the application; it can only be edited or accessed within the component itself. State holds information about the components which can change as different events are invoked within the application.

The below component has a state object called mood which has a value of “happy”. This state can be used throughout the render method and modified where needed.

class Person extends React.Component {

constructor(props) {

super(props);

this.state = {mood: "happy"};

}

render() {

return (

<div>

<h1>I am feeling {this.state.mood}</h1>

</div>

);

}

}

Props can be passed around from one component to another. See below example which demonstrates this.

function Sweet(props) {

return <h2>I am a { props.type }!</h2>;

}

function Sweets() {

return (

<>

<h1>What sweets are in my bag?</h1>

<Sweet type="Jelly" />

</>

);

}

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<Sweets />);

**2. In functional programming, what does the term functor mean? Can you give**

**an example in JavaScript?**

A data object known as a functor implements the map operation and can include elements of any data type. The map function accepts a function as an input and calls it for each item, creating a new functor of the same size each time. It allows values to be extracted from a container, altered in some way and then placed back into a container. For example, say we want map over an array, we can so by using a functor. See example below:

let array = [1,2,3]

console.log(array.map( num => num+3 ))

**3. 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 - One of the advantages of utilizing a callback function is that it enables you to wait for the outcome of a prior function call before proceeding with the execution of a subsequent function call.

Disadvantage – Nested callbacks are hard to read and can often make debugging very challenging because we have a nested layer of callbacks depending on a previous callback and it can get complex very fast

Promises

Advantage – Better option than callbacks when handling multiple asynchronous operations. Callbacks can lead a common problem called callback hell – using promises eliminates this issue.

Disadvantage – Promises do not have an API that allows the developer to work safely with data. Little is done by the API to stop programmers from writing hazardous code.

Streams

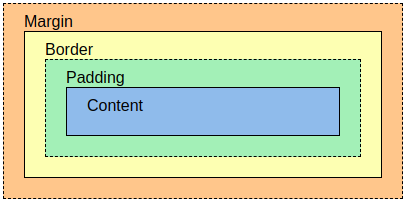
Advantage – You do not need to know the size of the data – chunks of the data are transmitted at a time. Processing can be done on these chunks as they arrive into the application. We don’t have to wait for all of the data to load before we process it.

Disadvantage – Once a stream is consumed it cannot be used again otherwise an IllegalStateException will be thrown

**4. 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 used by developers for designing websites. Every element on a HTML has some form of box around which can be modified to be bigger or smaller or perhaps to create a layout. There are four main components of the box model:

* Margin - specified gap outside the border
* Border - encapsulates the content and padding of the element
* Padding - creates space around an element
* Content - e.g. text, images

See example below which demonstrates the box model

h1 {

width: 50px;

height: 50px;

padding: 10px;

margin: 5px

border: 1px solid;

}

**5. Detail how the browser loads and bootstraps a rich web application from an**

**initial URL**

When a user loads a rich web app from a URL the following steps occur:

* When the user navigates to a URL in the browser, the browser consults the Domain Name Servers (DNS) to find the server where the website is located.
* The browser then establishes a TCP/IP connection with the server and the designated port associated with that URL in order to retrieve the files needed to load the website.
* Once the connection is established, an application server or HTTP proxy listening at the designated IP address accepts the connection from the browser.
* The browser sends a HTTP request over the open TCP/IP connection, and the application server parses the request and responds to the browser with the necessary files.
* The application server delivers the requested files to the browser over the same open TCP/IP connection.
* Some of the files that will be retrieved from the server are HTML, JavaScript as well as CSS