**1. What exactly do you mean when you say "prop drilling," and how do you**

**avoid it?**

Ans. We know that components in React can be passed some parameter namely prop.

Prop Drilling basically means passing the data through several nested children components, in a bid to deliver this data to a deeply-nested component.

The problem with Prop Drilling is that whenever data from the Parent component will be needed, it would have to come from each level, Regardless of the fact that it is not needed there and simply needed in last.

We avoid prop drilling by using Context API or by not breaking the code, and breaking it when necessary.

**2. In React JS, how do you add validation to props?**

**Ans.**

* **PropTypes.any :** This means the prop can be of any data type.
* **PropTypes.bool:** This means the prop should be a boolean.
* **PropTypes.number:** This means the prop should be a number.
* **PropTypes.string:** This means the prop should be a string.
* **PropTypes.func:** This means the prop should be a function.
* **PropTypes.array:** This means the prop should be an array.
* **PropTypes.object:** This means the prop should be an object.
* **PropTypes.symbol:** This means the prop should be a symbol.
* **PropTypes.instanceOf:** This means the prop should be an instance of a particular JavaScript class.
* **PropTypes.isRequired:** This means the prop should be provided.
* **PropTypes.oneOf():** This means the props should be one of several types of specified values.
* **PropTypes.element:** This means the props must be an element.

import React from 'react';

import PropTypes from 'prop-types';

class App extends React.Component {

render() {

return (

<div>

<h1>Hello!!</h1>

<h2>ReactJS Props validation</h2>

<table>

<tr>

<th>Type</th>

<th>Value</th>

<th>Valid</th>

</tr>

<tr>

<td>Array</td>

<td>{this.props.propArray}</td>

<td>{this.props.propArray ? "true" : "False"}</td>

</tr>

<tr>

<td>Boolean</td>

<td>{this.props.propBool ? "true" : "False"}</td>

<td>{this.props.propBool ? "true" : "False"}</td>

</tr>

<tr>

<td>Function</td>

<td>{this.props.propFunc(5)}</td>

<td>{this.props.propFunc(5) ? "true" : "False"}</td>

</tr>

<tr>

<td>String</td>

<td>{this.props.propString}</td>

<td>{this.props.propString ? "true" : "False"}</td>

</tr>

<tr>

<td>Number</td>

<td>{this.props.propNumber}</td>

<td>{this.props.propNumber ? "true" : "False"}</td>

</tr>

</table>

</div>

);

}

}

// Prop types for our Component

App.propTypes = {

propArray: PropTypes.array.isRequired,

propBool: PropTypes.bool.isRequired,

propFunc: PropTypes.func,

propNumber: PropTypes.number,

propString: PropTypes.string,

}

// Default Props for our Component

App.defaultProps = {

propArray: [1, 2, 3, 4, 5],

propBool: true,

propFunc: function (x) { return x \* 10 },

propNumber: 1,

propString: "iNeuron",

}

export default App;

**3. Is it possible to use classes in NodeJS?**

**Ans.** Yes (ES6)

Eg. class Polygon {

constructor(height, width) {

this.area = height \* width;

}

}

**4. What is the purpose of super(props)?**

**Ans.**  Super(props) is used to pass the props to parent constructor.

import React from "react";

class Person extends React.Component {

constructor(props) {

super(props);

console.log(this.props);

}

render() {

console.log(this.props);

return null;

}}

export default Person;

**5. Why are the Express app and server separated?**

**Ans.** Express 'app' and 'server' must be kept separate as by doing this, you will be separating the API declaration from the network-related configuration which benefits in the below listed ways: It allows testing the API in-process without having to perform the network calls. Faster testing execution