### 1.  Why can't browsers read JSX?

Browsers cannot read JSX directly because they can only understand JavaScript objects, and JSX is not a regular JavaScript object. Thus, we need to transform the JSX file into a JavaScript object using transpilers like Babel and then pass it to the browser.

### 2. Why we use JSX?

* It is faster than regular JavaScript because it performs optimization while translating the code to JavaScript.
* Instead of separating technologies by putting markup and logic in separate files, React uses components that contain both.
* t is type-safe, and most of the errors can be found at compilation time.
* It makes easier to create templates.

### 3. What do you understand by Virtual DOM?

A Virtual DOM is a lightweight JavaScript object which is an in-memory representation of real DOM. It is an intermediary step between the render function being called and the displaying of elements on the screen. It is similar to a node tree which lists the elements, their attributes, and content as objects and their properties. The render function creates a node tree of the React components and then updates this node tree in response to the mutations in the data model caused by various actions done by the user or by the system.

### 4. How React's ES6 syntax is different from ES5 syntax?

The React's ES6 syntax has changed from ES5 syntax in the following aspects.

// ES5

var React = require('react');

// ES6

**import** React from 'react';

**exports vs. export**

// ES5

module.exports = Component;

// ES6

export **default** Component;

**component and function**

// ES5

var MyComponent = React.createClass({

    render: function() {

**return**(

          <h3>Hello JavaTpoint</h3>

        );

    }

});

// ES6

**class** MyComponent **extends** React.Component {

     render() {

**return**(

          <h3>Hello Javatpoint</h3>

        );

    }

}

**Props**

// ES5

var App = React.createClass({

    propTypes: { name: React.PropTypes.string },

    render: function() {

**return**(

           <h3>Hello, {**this**.props.name}!</h3>

        );

    }

});

// ES6

**class** App **extends** React.Component {

    render() {

**return**(

          <h3>Hello, {**this**.props.name}!</h3>

        );

    }

}

state

var App = React.createClass({

    getInitialState: function() {

**return** { name: 'world' };

    },

    render: function() {

**return**(

          <h3>Hello, {**this**.state.name}!</h3>

        );

    }

});

// ES6

**class** App **extends** React.Component {

    constructor() {

**super**();

**this**.state = { name: 'world' };

    }

    render() {

**return**(

          <h3>Hello, {**this**.state.name}!</h3>

        );

    }

}

### 5.  Explain the purpose of render() in React.

It is mandatory for each React component to have a render() function. Render function is used to return the HTML which you want to display in a component. If you need to rendered more than one HTML element, you need to grouped together inside single enclosing tag (parent tag) such as <div>, <form>, <group> etc. This function returns the same result each time it is invoked.

**Example:** If you need to display a heading, you can do this as below.

**import** React from 'react'

**class** App **extends** React.Component {

   render (){

**return** (

         <h1>Hello World</h1>

      )

   }

}

export **default** App