1. Is JSX mandatory for React?

* JSX is not a requirement for using React. Using React without JSX is especially convenient when you don’t want to set up compilation in your build environment.
* Each JSX element is just syntactic sugar for calling React.createElement(component, props, ...children).

1. Is ES6 mandatory for React?

* Reactjs is the best frontend library ever created. It is made by Facebook to perform several tasks in the frontend itself.
* ES6 is the standardization of javascript for making code more readable and more accessible.
* If we don’t use ES6 in react, there is an alternative to perform.
* We use **create-react-class**instead of ES6.

1. How can I write comments in JSX?

* JSX provides us with the ability to write comments.
* Comments aid readability, understandability, maintainability, etc. of a codebase.
* JSX comments begin and end with curly braces {}.

1. What is <React.Fragment><React.Fragment /> and <></> ?

* A common pattern in React is for a component to return multiple elements.
* Fragments let you group a list of children without adding extra nodes to the DOM.
* <></> is shorthand syntax for Raect fragment.
* You can use <></> the same way you’d use any other element except that it doesn’t support keys or attributes.
* key is the only attribute that can be passed to Fragment.

1. What is Virtual DOM?

* The virtual DOM (VDOM) is a programming concept where an ideal, or “virtual”, representation of a UI is kept in memory and synced with the “real” DOM by a library such as ReactDOM
* In React world, the term “virtual DOM” is usually associated with [React elements](https://reactjs.org/docs/rendering-elements.html) since they are the objects representing the user interface.
* React, however, also uses internal objects called “fibers” to hold additional information about the component tree.
* They may also be considered a part of “virtual DOM” implementation in React.
* Shadow DOM and Virtual DOM both are different. The Shadow DOM is a browser technology designed primarily for scoping variables and CSS in web components.
* The virtual DOM is a concept implemented by libraries in JavaScript on top of browser APIs.

1. What is Reconciliation in React?

* The algorithm React uses to diff one tree with another to determine which parts need to be changed.
* React implements a heuristic O(n) algorithm based on two assumptions:
  + Two elements of different types will produce different trees.
  + The developer can hint at which child elements may be stable across different renders with a key prop.
* Whenever the root elements have different types, React will tear down the old tree and build the new tree from scratch. When tearing down a tree, old DOM nodes are destroyed.
* When comparing two React DOM elements of the same type, React looks at the attributes of both, keeps the same underlying DOM node, and only updates the changed attributes.
* When a component updates, the instance stays the same, so that state is maintained across renders. React updates the props of the underlying component instance to match the new element.
* By default, when recursing on the children of a DOM node, React just iterates over both lists of children at the same time and generates a mutation whenever there’s a difference.

1. What is React Fiber?

* React Fiber is an ongoing reimplementation of React's core algorithm.
* The goal of React Fiber is to increase its suitability for areas like animation, layout, and gestures.
* Its headline feature is incremental rendering: the ability to split rendering work into chunks and spread it out over multiple frames.
* Other key features include the ability to pause, abort, or reuse work as new updates come in; the ability to assign priority to different types of updates; and new concurrency primitives.
* The reconciler does the work of computing which parts of a tree have changed; the renderer then uses that information to actually update the rendered app.
* Fiber reimplements the reconciler.
* That's the purpose of React Fiber. Fiber is reimplementation of the stack, specialized for React components. You can think of a single fiber as a **virtual stack frame**.

1. Why do we need keys in React?

* Keys help React identify which items have changed, are added, or are removed.
* Keys should be given to the elements inside the array to give the elements a stable identity.
* By default, when recursing on the children of a DOM node, React just iterates over both lists of children at the same time and generates a mutation whenever there’s a difference.
* If you implement it naively, inserting an element at the beginning has worse performance.
* In order to solve this issue, React supports a key attribute.
* When children have keys, React uses the key to match children in the original tree with children in the subsequent tree.

1. Can we use index as keys in react?

* The best way to pick a key is to use a string that uniquely identifies a list item among its siblings.
* When that’s not the case, you can add a new ID property to your model or hash some parts of the content to generate a key.
* The key only has to be unique among its siblings, not globally unique.
* As a last resort, you can pass an item’s index in the array as a key.
* This can work well if the items are never reordered, but reorders will be slow.
* Reorders can also cause issues with component state when indexes are used as keys.
* Component instances are updated and reused based on their key.
* If the key is an index, moving an item changes it.
* As a result, component state for things like uncontrolled inputs can get mixed up and updated in unexpected ways.

1. What is props in react?

* When React sees an element representing a user-defined component, it passes JSX attributes and children to this component as a single object. We call this object “props”.
* Props are arguments passed into React components.
* Props are passed to components via HTML attributes.
* React Props are like function arguments in JavaScript *and* attributes in HTML.

1. What is Config Driven UI?

* CDD is a way of using modularity to build a loosely coupled set of components that are then composed together using a common interface.
* UI which is rendered based on config is Config Driven UI.
* It is dynamic and super efficient.