1. **Is JSX mandatory for React?**

   No, JSX is not mandatory for React. You can use React without JSX by using the `React.createElement()` function to create elements. However, JSX is generally considered more convenient and readable, and it is the preferred way to write React components.

2. **Is ES6 mandatory for React?**

   No, ES6 is not strictly mandatory for React.
   However, it is highly recommended and widely used in modern React development. ES6 introduces features like arrow functions, classes, and destructuring, making code more concise and readable.

   You can use React with older JavaScript syntax. However, you'll miss out on many benefits and might find it more challenging to write and maintain your code.

   How to use React without ES6:

1. Create React Class: You can use the `create-react-class` module to define React components without using ES6 classes.
2. Transpiling: You can use tools like Babel to transpile your ES6 code into older JavaScript syntax that can be understood by older browsers.

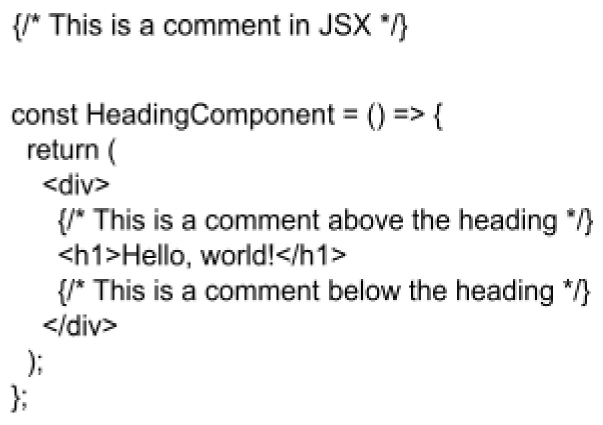
3. **{TitleComponent} vs {<TitleComponent/>} vs {<TitleComponent><TitleComponent/>} in JSX**

   {TitleComponent}: This value describes the TitleComponent as a javascript expression or a variable. The {} can embed a javascript expression or a variable inside it.

   {<TitleComponent/>} : This value represents a Component that is basically returning Some JSX value. In simple terms TitleComponent is a function that is returning a JSX value. A component is written inside the {< />} expression.

   {<TitleComponent><TitleComponent/>} : <TitleComponent/> and **<TitleComponent><TitleComponent/>** are equivalent only when < TitleComponent /> has no child components. The opening and closing tags are created to include the child components.

4. How can I write comments in JSX?



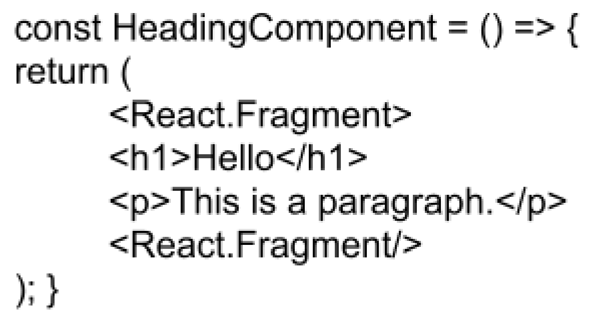
```
{/* This is a comment in JSX */}


const HeadingComponent = () => {
  return (
    <div>
      {/* This is a comment above the heading */}
      <h1>Hello, world!</h1>
      {/* This is a comment below the heading */}
    </div>
  );
};
```

The comment syntax is enclosed within `{/*` and `*/}`.

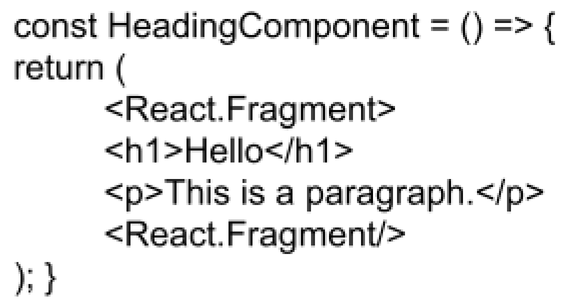
5. What is <React.Fragment><React.Fragment/> and <></>?

<React.Fragment><React.Fragment/>
To group elements without adding an extra node to the DOM**.**



```
const HeadingComponent = () => {
return (
        <React.Fragment>
        <h1>Hello</h1>
        <p>This is a paragraph.</p>
        <React.Fragment/>
); }
```

<></>
A shortcut to write `<React.Fragment>`



```
const HeadingComponent = () => {
return (
        <React.Fragment>
        <h1>Hello</h1>
        <p>This is a paragraph.</p>
        <React.Fragment/>
); }
```

6. **What is Virtual DOM?**
   It's a lightweight, in-memory representation of the actual Document Object Model (DOM), which is the structure and content of a web page.

   How does it work?

   1. **Initial Rendering:** When a React component renders for the first time, React creates a virtual DOM tree representing the UI.

   2. **State Changes**: When the component's state or props change, React creates a new virtual DOM tree.

   3. **Diffing:** React efficiently compares the old and new virtual DOM trees to identify the minimal set of changes required to update the actual DOM.

   4. **DOM Updates:** React applies only the necessary changes to the actual DOM, minimizing the number of expensive DOM manipulations.

7. **What is Reconciliation in React?**

   React Reconciliation is the process by which React efficiently updates the DOM when component state or props change.

   This process involves two key steps:

   1. Creating a Virtual DOM:

   - When a component renders, React creates a virtual DOM representation of the UI, which is a lightweight, in-memory copy of the actual DOM.
   - This virtual DOM is a tree-like structure that represents the hierarchy of components and their corresponding elements.

   2. Diffing Algorithm:

   - When a component's state or props change, React creates a new virtual DOM.
   - The diffing algorithm compares the old and new virtual DOMs to identify the minimum number of changes required to update the actual DOM.
   - This algorithm is highly optimised to efficiently identify differences, such as added, removed, or modified elements.

8. **What is React Fiber?**

   React Fiber is a significant reimplementation of React's core reconciliation algorithm.It's designed to enhance performance, responsiveness, and user experience in complex and demanding applications.

   What Problem Does Fiber Solve?
   Rendering the page, responding to user actions, running Javascript and much more are all handled by the browser's main thread. If at any time our main thread gets blocked, the user's experience can become laggy and slow.

   We can safely move some of these things to another thread safely using Web Workers, however, they can't access the DOM and manipulate it. That means you can only move heavy computations or lengthy network request calls, however, you can't change what's on the screen without ease with a Web Worker. There are workarounds, however, they tend not to be the best practice.

   Since our React code runs on Javascript, the browser's main thread handles executing the rendering loop.

9. **Why do we need keys in React? When do we need keys in React?**

   Whenever we are looping on any list, we have to give a key for every list of children.

   E.g <RestaurentCard key = {restaurent.info.id} />

   Every child of list should be uniquely represented. If we add new child to th list, react will rerender all the childs of the list , because react does not know which child is new.

   React don't know on which it should need to place.

   If we give to them unique id, react will render only that one new child instead of rerendering all childs of the list.

   It simplify optimization process.

10. **Can we use indexes as keys in React?**
    We can use indexes as key in react. But it is not mandatory or recommended. When items are re-ordered, React may mistakenly identify them as different elements, leading to unnecessary re-renders and potentially impacting performance.

11. **What are props in React?**
    Short form for properties. To dynamically send data to a component we use props. Passing a prop to a function is like passing an argument to a function.

Passing Props to a Component

```
<RestaurantCard
  resName="Meghana Foods"
  cuisine="Biryani, North Indian"
/>
```

Receiving props in the Component

Props will be wrapped and send in Javascript object

```
const RestaurantCard = (props) => {
  return(
    <div>{props.resName}</div>
  )
}
```

Destructuring Props

```
const RestaurantCard = ({resName, cuisine}) => {
  return(
    <div>{resName}</div>
  )
}
```

12. What is a Config Driven UI ?

It is a user Interface that is built and configured using a declaration configuration file or data structure, rather than being hardcoded.

Config is the data coming from the api which keeps on changing according to different factors like user, location, etc.