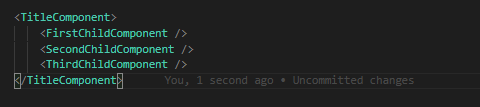
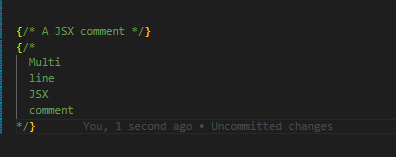
1. Is JSX mandatory for React?
   * No, JSX is not a requirement for using React. Using JSX without React is especially convenient when you don’t want to set-up compilation in your build environment.
   * Each JSX element is just syntactic sugar for calling React.createElement(component, props, ...children). So, anything you can do with JSX can also be done with just plain JavaScript.
   * Example:

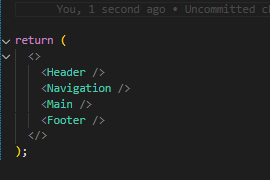


1. Is ES6 mandatory for React?
   * It is not mandatory but it’s highly recommended. React uses many ES6 features, including classes, arrow functions, and variables.
   * If you don't use ES6, you can use the create-react-class module instead of ES6 classes. The API of ES6 classes is similar to createReactClass(), but there are some exceptions.
2. {TitleComponent} vs {<TitleComponent/>} vs {<TitleComponent></TitleComponent>} in JSX.
   * **{TitleComponent}** : This value describes the TitleComponent as a JS expression or a variable or React element. The {} can embed a JS expression or a variable or React element 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. If component is written inside the {< />} expression, it should be returning some JSX value.
   * **{<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.
   * Example:



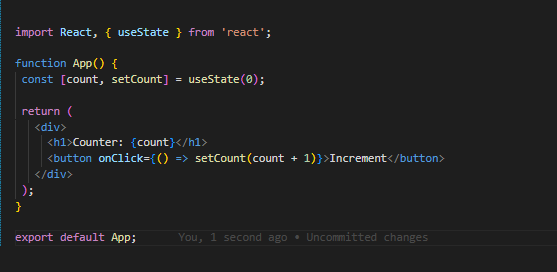
1. How can I write comments in JSX?
   * {/\* \*/} : for single or multiline comments



1. What is <React.Fragment></React.Fragment> and <></>?
   * <React.Fragment></React.Fragment> is a feature in React that allows you to return multiple elements from a React component by allowing you to group a list of children without adding extra nodes to DOM.
   * <></> is the shorthand tag for React.Fragment. The only difference between them is that the shorthand version does not support the key attribute.
   * 
   * 
2. What is virtual DOM?
   * The virtual DOM is an in-memory representation of the real DOM elements. Instead of interacting directly with the real DOM, which can be slow and costly in terms of performance, Reacts creates a virtual representation of the UI components. This virtual representation is a lightweight JS object that mirrors the structure of real DOM.
3. How does virtual DOM works?
   * Step 1 – Initial Rendering:
     + When the app starts, the entire UI is represented as a virtual DOM. React elements are created and rendered into the virtual structure.
   * Step 2 – State and Props Changes:
     + As the state and props change in the app, React re-renders the affected components in the virtual DOM. These changes do not immediately impact the real DOM.
   * Step 3 – Comparison using Diff Algorithm:
     + React then uses a diffing algorithm to compare the current version of the virtual DOM with the previous version. This process identifies the differences(or “diffs”) between the two versions.
   * Step 4 – Reconciliation Process:
     + Based on the differences identified, React determines the most efficient way to update the real DOM. Only the parts of the real DOM that need to be updated are changed, rather than re-rendering the entire UI. This selective updating is quick and performant.
   * Step 5 – Update to the Real DOM:
     + Finally, React applies the necessary changes to the real DOM. This might involve adding, removing, or updating elements based on the differences detected in step 3.

* For example,

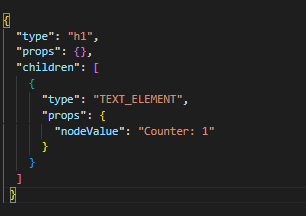
Let’s say we’ve the following counter functionality in the App component:



The virtual DOM representation will look like this:



When the Increase button is clicked once, only the h1 element is changed:



1. Difference between Virtual DOM and Real DOM?
   * The real DOM and the virtual DOM serve similar purposes but operate in distinct ways with significant implications for performance and efficiency.
   * The real DOM is a built-in standard interface in browsers that represents and interacts with HTML elements, from Doctype declaration and the root html element to every other element in it.

This real DOM represents the whole HTML document as a tree structure and allows JavaScript to manipulate and change HTML documents. Sometimes when those changes occur, the whole document might re-render.

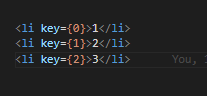
* + This is in contrast to the virtual DOM, which uses a diff algorithm to compare the current and previous versions of updates to the DOM. It only re-renders the parts of the UI that have changed, instead of the whole thing.
  + Benefits of using virtual DOM:
    - Simplified Development:
      * The Virtual DOM lets you write code in a more declarative style, which means that instead of writing detailed instructions on how to update the UI, you can simply describe what the UI should look like, and React takes care of the rest.
      * This is made possible by React's declarative syntax and its component-based architecture.
    - Improved Performance:
      * One of the major advantages of using the virtual DOM is the significant performance improvement it offers. Direct manipulation of the real DOM is slow and can lead to performance issues, especially in complex applications.
    - Enhanced user experience:
      * The Virtual DOM contributes to a better UX by ensuring that UI updates are smooth, responsive, and without full-page refreshes. Users are less likely to experience lag or jank, resulting in a more seamless interaction with the app.
    - Cross-Platform development:
      * The principles of the Virtual DOM are not limited to web development only. React Native – a version of React for building cross-platform mobile apps – uses a similar approach.
      * This increases productivity and reduces development time because you can reuse code across web and mobile platforms.

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Real DOM** | **Virtual DOM** |
| **Definition** | Standard browser API for representing and interacting with HTML documents | In-memory representation of the real DOM |
| **Flexibility** | Directly manipulated via JS or DOM APIs | Abstracted and optimized by the framework |
| **Implementation** | Provided by the browser | Implemented by framework like React and Vue |
| **Performance** | Direct manipulation can be slow and cause performance issues | Already optimized for efficient updates |
| **Usage** | For rendering and interacting with web documents | For efficient UI updates by framework |
| **Updates** | Immediately updates to the UI | Updates are batched and optimized |
| **Use Cases** | General web development and document manipulation | Efficient UI updates in frameworks like React and Vue |

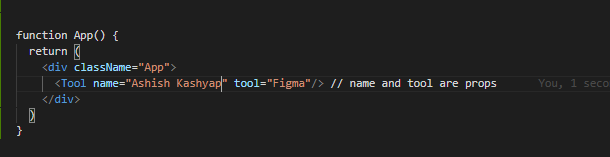
1. What Is Reconciliation in React?
   * Reconciliation is the process through which React updates the Browser DOM and makes React work faster.
   * React use a diffing algorithm so that component updates are predictable and faster. React would first calculate the difference between the real DOM and the copy of DOM (Virtual DOM) when there's an update of components.

React stores a copy of Browser DOM which is called Virtual DOM. When we make changes or add data, React creates a new Virtual DOM and compares it with the previous one. Comparison is done by Diffing Algorithm. React compares the Virtual DOM with Real DOM. It finds out the changed nodes and updates only the changed nodes in Real DOM leaving the rest nodes as it is. This process is called Reconciliation.

1. What is React Fiber?
   * React Fiber is a concept of ReactJS that is used to render a system faster, smoother and smarter. The Fiber reconciler, which became the default reconciler for React 16 and above, is a complete rewrite of React’s reconciliation algorithm to solve some long-standing issues in React.
   * Because Fiber is asynchronous, React can:
     + Pause, resume, and restart rendering work on components as new updates come in
     + Reuse previously completed work and even abort it if not needed
     + Split work into chunks and prioritize tasks based on importance
2. Why we need keys in React? When do we need keys in React?
   * A key is a special attribute you need to include when creating lists of elements in React. Keys are used in React to identify which items in the list are changed, updated, or deleted. In other words, we can say that keys are unique Identifier used to give an identity to the elements in the lists. Keys should be given to the elements within the array to give the elements a stable identity.
   * Example:



1. Can we use index as keys in React?
   * Yes, we can use the index as keys, but it is not considered as a good practice to use them because if the order of items may change. This can negatively impact performance and may cause issues with component state. Keys are taken from each object which is being rendered. There might be a possibility that if we modify the incoming data react may render them in unusual order.
2. What is props in React?
   * Props stands for properties.
   * These are arguments passed into React components.
   * They are used in React to pass data from one component to another (from a parent component to a child component(s)). They are useful when you want the flow of data in your app to be dynamic.
   * Example:



1. What is Config Driven UI?
   * Config Driven UI are based on the configurations of the data application receives. It is rather a good practice to use config driven UIs to make application for dynamic. It is a very common & basic approach to interact with the User. It provides a generic interface to develop things which help your project scale well. It saves a lot of development time and effort. A typical login form, common in most of the Apps. Most of these forms also get frequent updates as the requirements increase in terms of Form Validations, dropdown options,.. or design changes.
   * For example:

In Swiggy.com, we’ve different carousels / restaurant card for different location. There might be some restaurants or offers that is present in one region (Bangalore) but not in other (Kolkata). So, it’s UI will keep changing depending on location.

* This is called Config driven UI (UI is driven by data / config).
* It is basically controlling your UI on basis of data/config and this comes from backend.