React

React is a JavaScript library for building UI components.It is created by Meta.

How does React Work?

React creates a VIRTUAL DOM in memory.

Instead of manipulating the browser's DOM directly, React creates a virtual DOM in memory, where it does all the necessary manipulating, before making the changes in the browser DOM.

React only changes what needs to be changed. React finds out what changes have been made, and changes **only** what needs to be changed.

The createRoot Function

The `createRoot()` function takes one argument, an HTML element.

The purpose of the function is to define the HTML element where a React component should be displayed.

The render Method

The `render()` method is then called to define the React component that should be rendered.

What is Component?

Components are independent and reusable bits of code.

Components are like functions that return HTML elements.

Components come in two types, Class components and Function components and always return JSX code

Component is Tag

When creating a component, the component's name *MUST* start with an upper case letter.

A class component must include the `extends React.Component` statement. This statement creates an inheritance to React.Component, and gives your component access to React.Component's functions.

The component also requires a `render()` method, this method returns HTML

JSX

React embraces the fact that rendering logic is inherently coupled with other UI logic: how events are handled, how the state changes over time, and how the data is prepared for display.

What is JSX?

JSX stands for JavaScript XML.

JSX allows us to write HTML in React.

JSX makes it easier to write and add HTML in Rea

JSX converts HTML tags into react elements.

class = className

The `class` attribute is a much used attribute in HTML, but since JSX is rendered as JavaScript, and the `class` keyword is a reserved word in JavaScript, you are not allowed to use it in JSX.

Expressions in JSX

With JSX you can write expressions inside curly braces `{ }`.

One Top Level Element

A common pattern in React is for a component to return multiple elements.

But The code must be wrapped in *ONE* top level element.

So if you like to write *two* paragraphs, you must put them inside a parent element, like a `div` element

JSX will throw an error if the code is not correct, or if the parent element is missed.

What is Fragment?

Fragments let you group a list of children without adding extra nodes to the DOM.

```
return (
    <React.Fragment>
      <ChildA />
      <ChildB />
      <ChildC />
    </React.Fragment>
  );
```

There is also a new short syntax for declaring them: `<></>`

Props

Props are arguments passed into React components.

Props are passed to components via HTML attributes.

Event Handling

Handling events with React elements is very similar to handling events on DOM elements. There are some syntax differences:

- React events are named using camelCase, rather than lowercase.
- With JSX you pass a function as the event handler, rather than a string.

Your event handlers will be passed instances of `SyntheticEvent`, a cross-browser wrapper around the browser's native event. It has the same interface as the browser's native event, including `stopPropagation()` and `preventDefault()`, except the events work identically across all browsers.

If you find that you need the underlying browser event for some reason, simply use the `nativeEvent` attribute to get it. The synthetic events are different from, and do not map directly to, the browser's native events. For example in `onMouseLeave` `event.nativeEvent` will point to a `mouseout` event. The specific mapping is not part of the public API and may change at any time.

Supported Events

React normalizes events so that they have consistent properties across different browsers.

Chekout the the reference for events: https://legacy.reactjs.org/docs/events.html#reference

Conditional Rendering

There are several ways to do this:

- if Statement
- Logical && Operator
- Ternary Operator

Rendering Multiple Components(List)

As you know for transforming lists in JavaScript we use map() array method.It is the preferred method for list in React as well.

You can build collections of elements and include them in JSX using curly braces `{}`.

```
const numbers = [1, 2, 3, 4, 5];
const listItems = numbers.map((number) =>
{number}
)
```

Keys

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:

```
const numbers = [1, 2, 3, 4, 5];
const listItems = numbers.map((number) =>
 <li key={number.toString()}>    {number}  </li>
);
```

Keys Must Only Be Unique Among Siblings

Conditional Rendering

There are several ways to do this:

- if Statement
- Logical && Operator
- Ternary Operator

Rendering Multiple Components(List)

As you know for transforming lists in JavaScript we use map() array method.It is the preferred method for list in React as well.

You can build collections of elements and include them in JSX using curly braces `{}`.

```
const numbers = [1, 2, 3, 4, 5];
const listItems = numbers.map((number) =>
{number}
)
```

Keys

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:

```
const numbers = [1, 2, 3, 4, 5];
const listItems = numbers.map((number) =>
 <li key={number.toString()}>    {number}  </li>
);
```

Keys Must Only Be Unique Among Siblings