When does re-rendering in React happens?

In React, a re-render of a component will occur when there is a ***change in the component's state or props***. If you don't use state in your component and only rely on local variables, React will not automatically trigger a re-render when those local variables change.

When you update a local variable in a component without using state or props, React is unaware of this change. As a result, it will not re-render the component to reflect the updated local variable.

To cause a re-render in React, you need to either update the component's state (using `setState()` for class components or `useState()` for functional components) or pass new props to the component from its parent.

Here's an example to illustrate this behavior:

```jsx
import React, { useState } from 'react';

const MyComponent = () => {
  // Local variable, not state
  let count = 0;

  const handleClick = () => {
    count += 1; // Modifying the local variable directly, but React won't re-render
    };

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={handleClick}>Increment</button>
    </div>
  );
};

export default MyComponent;
```

In the above example, when you click the "Increment" button, the `handleClick` function updates the `count` variable. However, React doesn't detect this change as it's a local variable and not part of the component's state or props. As a result, the displayed count will not change, and the component won't re-render.

If you want to make the component re-render and display the updated count, you should use state, like this:

```jsx
import React, { useState } from 'react';

const MyComponent = () => {
  // Using state instead of a local variable
  const [count, setCount] = useState(0);

  const handleClick = () => {
    setCount(prevCount => prevCount + 1); // Updating state triggers a re-render
  };

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={handleClick}>Increment</button>
    </div>
  );
};

export default MyComponent;
```

By using the `useState()` hook, the component now has state, and updating the `count` state with `setCount()` will trigger a re-render, updating the displayed count on the page.

What is React State?

In React, "state" refers to the internal data of a component that can change over time. It represents the dynamic information that affects the component's behavior and rendering. When the state of a component is updated, React automatically triggers a re-render to reflect the changes in the user interface. State allows React components to maintain and manage their own data throughout their lifecycle, making them interactive and responsive to user interactions and other dynamic events.