JSX expressions allow you to embed JavaScript expressions within curly braces `{}` directly within JSX. This enables dynamic content to be inserted into JSX elements. JSX expressions can include variables, function calls, and any valid JavaScript expression.

Here's an example of JSX expressions in action:

```jsx
import React from 'react';

function App() {
  const name = 'John';
  const age = 30;

  // JSX expression with variables and JavaScript expressions
  return (
    <div>
      <h1>Hello, {name}!</h1>
      <p>You are {age} years old.</p>
      <p>Your age plus 5 is {age + 5}.</p>
      <p>Random number: {Math.floor(Math.random() * 10)}</p>
    </div>
  );
}

export default App;
```

In this example:

- We have a functional component `App`.
- Inside the `App` component, we define two variables `name` and `age`.
- We use JSX expressions to dynamically insert the values of these variables into the JSX elements: `{name}` and `{age}`.
- We also use JavaScript expressions within JSX to perform calculations and generate dynamic content, such as `{age + 5}` to display the age plus 5, and `{Math.floor(Math.random() * 10)}` to generate a random number between 0 and 9.

When this component is rendered by React, it will display:

```
Hello, John!
You are 30 years old.
Your age plus 5 is 35.
Random number: [a random number between 0 and 9]
```

This demonstrates how JSX expressions allow for dynamic content and computation within JSX elements, enabling developers to create dynamic and interactive user interfaces in React.

Examples:

1. **Embedding Variables**:
```jsx
import React from 'react';

function Greeting() {
  const name = 'John';
  return <h1>Hello, {name}!</h1>;
}

export default Greeting;
```

In this example, the value of the `name` variable is embedded within the JSX expression using curly braces `{}`.

2. **Embedding JavaScript Expressions**:
```jsx
import React from 'react';

function RandomNumber() {
  const randomNumber = Math.floor(Math.random() * 100);
  return <p>Random Number: {randomNumber}</p>;
}

export default RandomNumber;
```

Here, a JavaScript expression `Math.floor(Math.random() * 100)` generates a random number between 0 and 99, which is embedded within the JSX expression.

3. **Conditional Rendering**:
```jsx
import React from 'react';

function Message({ isLoggedIn }) {
  return (
    <div>
```

```
      {isLoggedIn ? (
        <h1>Welcome back!</h1>
      ) : (
        <h1>Please log in.</h1>
      )}
    </div>
  );
}

export default Message;
```

In this example, a ternary operator is used within the JSX expression to conditionally render different elements based on the value of the `isLoggedIn` prop.

4. **Mapping Arrays**:
```jsx
import React from 'react';

function List({ items }) {
  return (
    <ul>
      {items.map((item, index) => (
        <li key={index}>{item}</li>
      ))}
    </ul>
  );
}

export default List;
```

Here, the `map()` function is used to iterate over an array of items, and for each item, a JSX expression is used to create a `<li>` element with the item's content.

These examples demonstrate the flexibility and power of JSX expressions in React, allowing for dynamic content, conditional rendering, and iteration over arrays directly within JSX.

JSX and HTML differences

While JSX syntax resembles HTML, there are some key differences between JSX and HTML:

1. **Tag Names**: In JSX, tag names are case-sensitive. Component names in JSX must start with a capital letter, while HTML tag names are case-insensitive.

   ```jsx
   // JSX
   <MyComponent />

   // HTML
   <div></div>
   ```

2. **Attribute Names**: In JSX, attribute names are written in camelCase, whereas in HTML, they are typically written in lowercase.

   ```jsx
   // JSX
   <input type="text" className="input" />

   // HTML
   <input type="text" class="input" />
   ```

3. **Style Attribute**: In JSX, the `style` attribute accepts an object with camelCased property names, whereas in HTML, the `style` attribute accepts CSS as a string.

   ```jsx
   // JSX
   <div style={{ color: 'red', fontSize: '16px' }}>Hello</div>

   // HTML
   <div style="color: red; font-size: 16px;">Hello</div>
   ```

4. **Comments**: JSX allows JavaScript-style comments (`{/* */}`), while HTML uses HTML-style comments (`<!-- -->`).

   ```jsx
   {/* This is a JSX comment */}
   <div>Hello</div>

   <!-- This is an HTML comment -->
   <div>Hello</div>
   ```

5. **Class Attribute**: In JSX, the `class` attribute is written as `className` due to the reserved nature of `class` in JavaScript, while in HTML, it's simply `class`.

   ```jsx
   // JSX
   <div className="container">Hello</div>

   // HTML
   <div class="container">Hello</div>
   ```

6. **Self-Closing Tags**: JSX requires self-closing tags for elements without children, while in HTML, it's optional.

   ```jsx
   // JSX
   <img src="example.jpg" alt="Example Image" />

   // HTML
   <img src="example.jpg" alt="Example Image">
   ```

7. **Event Handlers**: In JSX, event handlers are written inline using camelCase, while in HTML, they are written in lowercase and enclosed in quotes.

   ```jsx
   // JSX
   <button onClick={handleClick}>Click Me</button>

   // HTML
   <button onclick="handleClick()">Click Me</button>
   ```

Overall, while JSX resembles HTML, it's important to keep these differences in mind when working with React components and JSX syntax.