Building Components

Terms

Fragment
Immutable
Props
State hook

Summary

- In React apps, a component can only return a single element. To return multiple elements, we wrap them in a fragment, which is represented by empty angle brackets.
- To render a list in JSX, we use the 'array.map()' method. When mapping items, each item must have a unique key, which can be a string or a number.
- To conditionally render content, we can use an 'if' statement or a ternary operator.
- We use the state hook to define state (data that can change over time) in a component. A hook is a function that allows us to tap into built-in features in React.
- Components can optionally have props (short for properties) to accept input.
- We can pass data and functions to a component using props. Functions are used to notify the parent (consumer) of a component about certain events that occur in the component, such as an item being clicked or selected.
- We should treat props as immutable (read-only) and not modify them.
- When the state or props of a component change, React will re-render the component and update the DOM accordingly.

- In React apps, a component can only return a single element. To return multiple elements, we wrap them in a fragment, which is represented by empty angle brackets.

- To render a list in JSX, we use the 'array.map()' method. When mapping items, each item must have a unique key, which can be a string or a number.

- To conditionally render content, we can use an 'if' statement or a ternary operator.

- We use the state hook to define state (data that can change over time) in a component. A hook is a function that allows us to tap into built-in features in React.

- Components can optionally have props (short for properties) to accept input.

- We can pass data and functions to a component using props. Functions are used to notify the parent (consumer) of a component about certain events that occur in the component, such as an item being clicked or selected.

- We should treat props as immutable (read-only) and not modify them.

- When the state or props of a component change, React will re-render the component and update the DOM accordingly.

CREATING A COMPONENT

```
const Message = () => {
  return <h1>Hello World</h1>;
}

export default Message;
```

RENDERING A LIST

```
const Component = () => {
  const items = ['a', 'b', 'c'];
  return (
    <ul>
      {items.map((item) => (
        <li key={item}>item</li>
      ))}
    </ul>
  );
};
```

CONDITIONAL RENDERING

```
{items.length === 0 ? 'a' : 'b'}
{items.length === 0 && 'a'}
```


HANDLING EVENTS

```
<button onClick={() => console.log('clicked')}></button>;
```

DEFINING STATE

```
const [name, setName] = useState('');
```

PROPS

```
interface Props {
  name: string;
}

const Component = ({ name }: Props) => {
  return <p>{name}</p>
};
```

PASSING CHILDREN

```
interface Props {
  children: ReactNode
}

const Component = ({ children }: Props) => {
  return <div>{children}</div>
};
```

