React Online Marathon

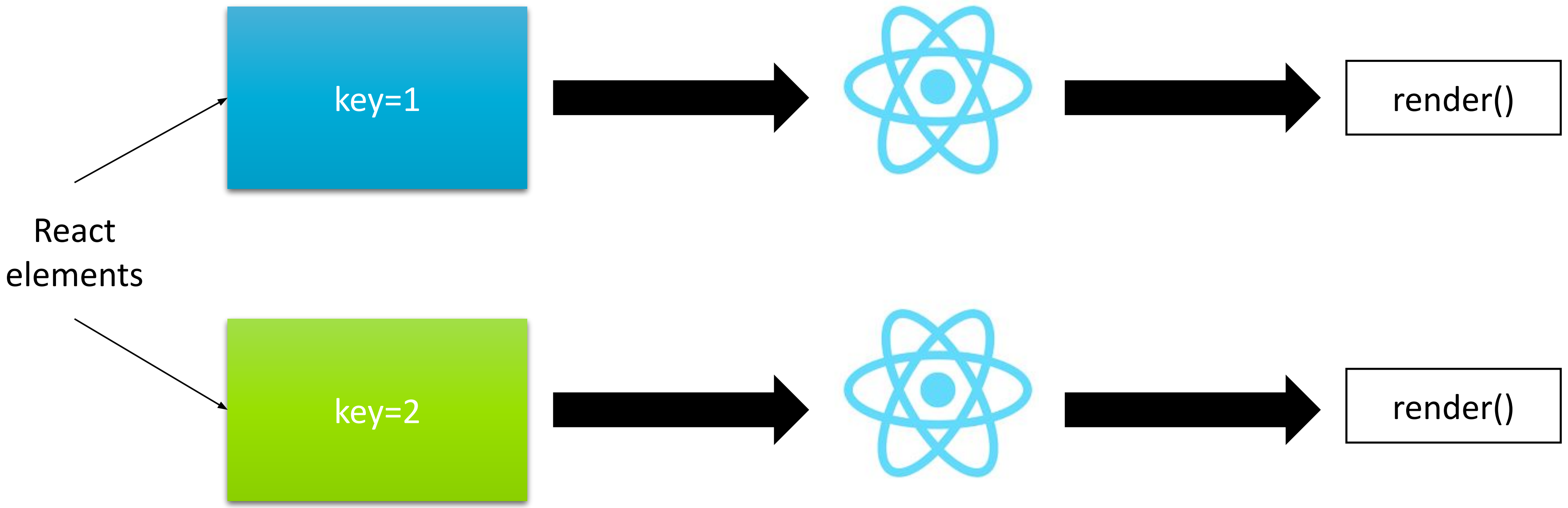
softserve

Key

softserve

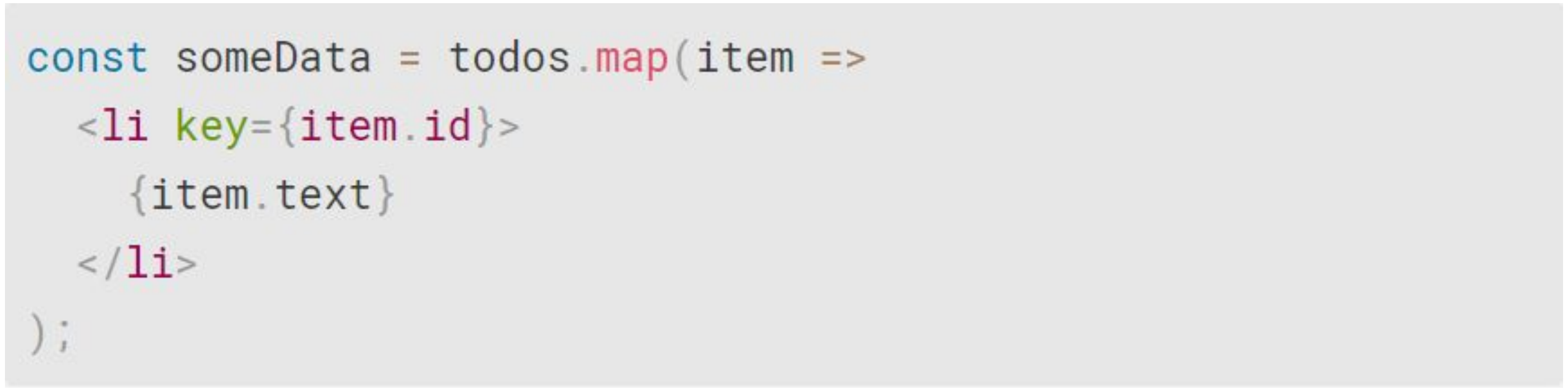
Usually you would render lists inside a component

A key is a special string attribute which helps React to identify components uniquely



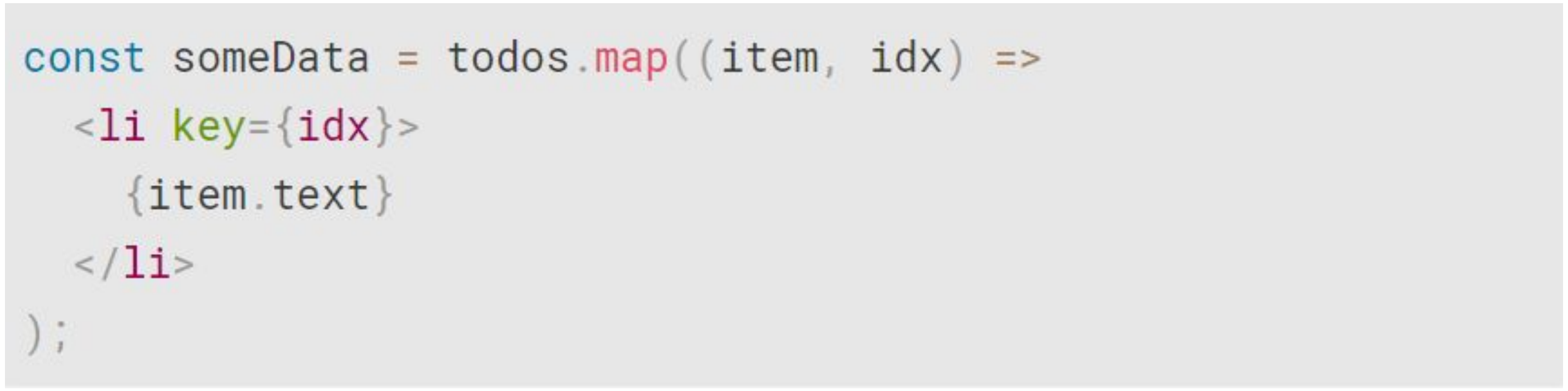
softserve

Most often you would use IDs from your data as keys:



```
const someData = todos.map(item =>
  <li key={item.id}>
    {item.text}
  </li>
);
```

When you don’t have stable IDs for rendered items, you may use the item index as a key as a last resort:

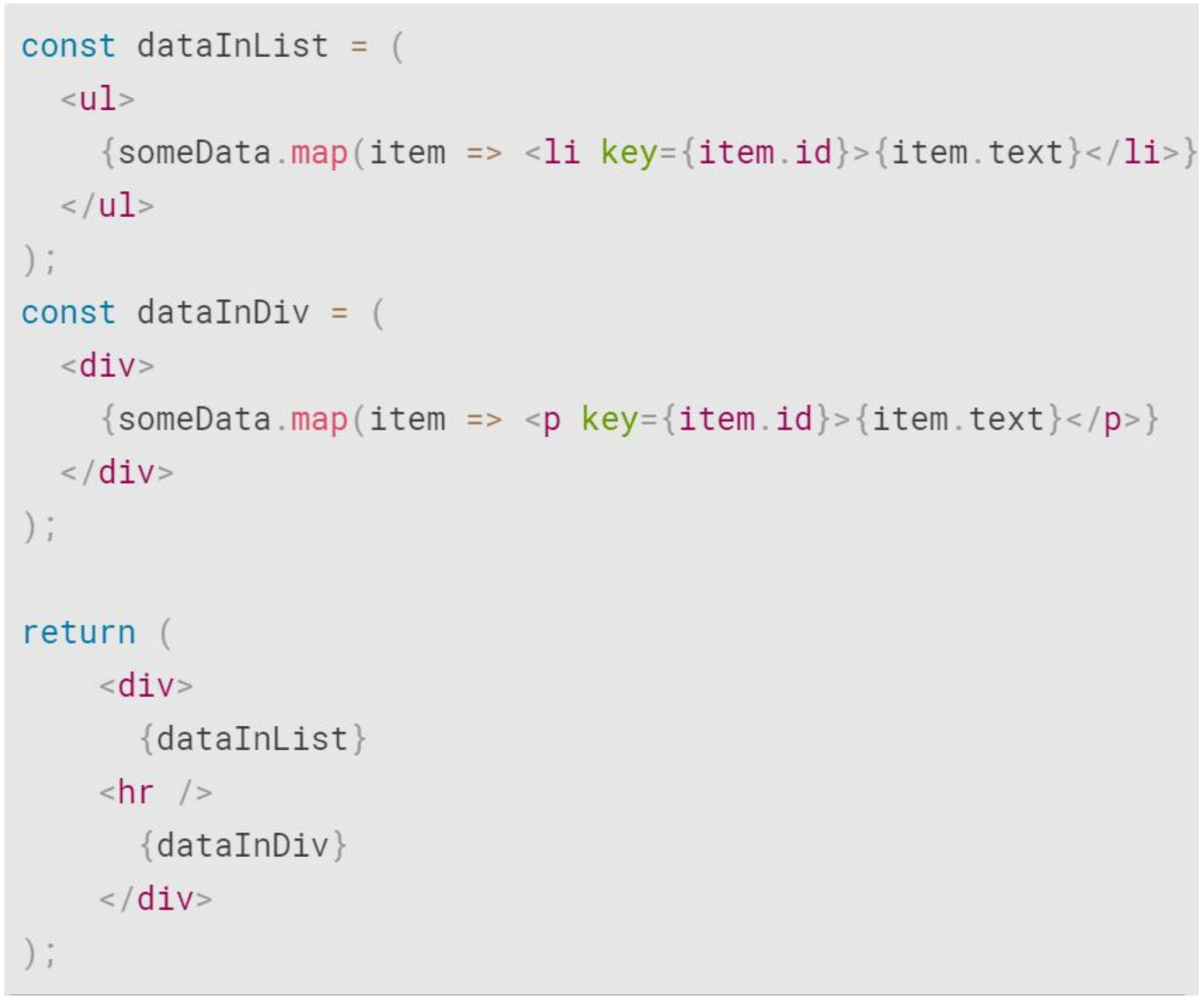


```
const someData = todos.map((item, idx) =>
  <li key={idx}>
    {item.text}
  </li>
);
```

But it is not recommended to use indexes for keys if the order of items may change

softserve

Keys used within arrays should be unique among their siblings - they don’t need to be globally unique



```
const dataInList = (
  <ul>
    {someData.map(item => <li key={item.id}>{item.text}</li>}
  </ul>
);
const dataInDiv = (
  <div>
    {someData.map(item => <p key={item.id}>{item.text}</p>}
  </div>
);

return (
    <div>
      {dataInList}
    <hr />
      {dataInDiv}
    </div>
);
```

softserve

React Online Marathon

softserve

State

A state is a special instance property and can be defined as an object contains data specific to this component that may change over a lifetime of the component. The state is user-defined:

```
Class MyClass extends React.Component
{
    constructor(props)
    {
        super(props);
        this.state = { someData: 'some value' };
    }
}
```

State should never be updated directly, if we update the state of any component like the following

```
this.state.someData = 'new value';
```

the webpage will not re-render itself because React State will not be able to detect the changes made

setState() enqueues changes to the component state and tells React that this component and its children need to be re-rendered with the updated state

```
this.setState({someData: "new value"});
```

softserve

React Online Marathon

softserve

Props

softserve

Props are basically kind of object. They used to pass data between components

```
const element = <ChildComponent someData='Some value' />;
```

Whether you declare a component as a class

```
class ChildComponent extends React.Component {
  render() {
    return <div>{this.props.someData}</div>;
  }
}
```

softserve

or a function

```
function ChildComponent(props) {
  return <div>{props.someData}</div>;
}
```

it must never modify its own props - props data is read-only

softserve

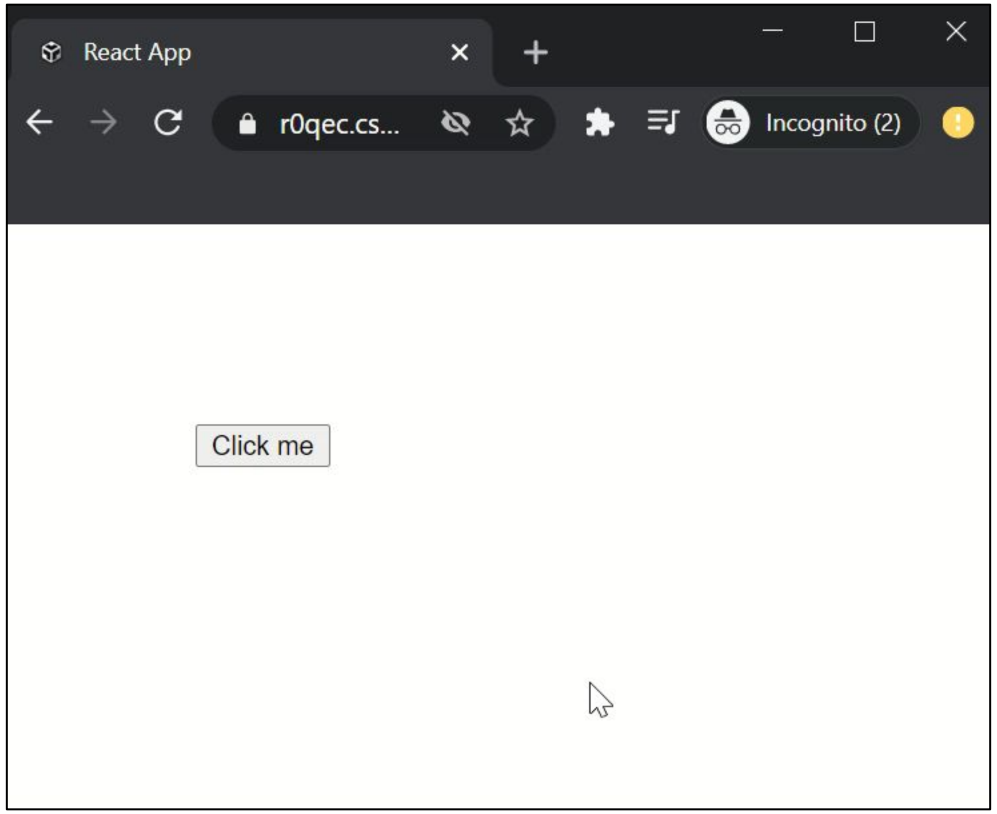
React Online Marathon

softserve

Events

softserve

Events are the triggered reactions to specific actions like mouse hover, mouse click, key press, etc.



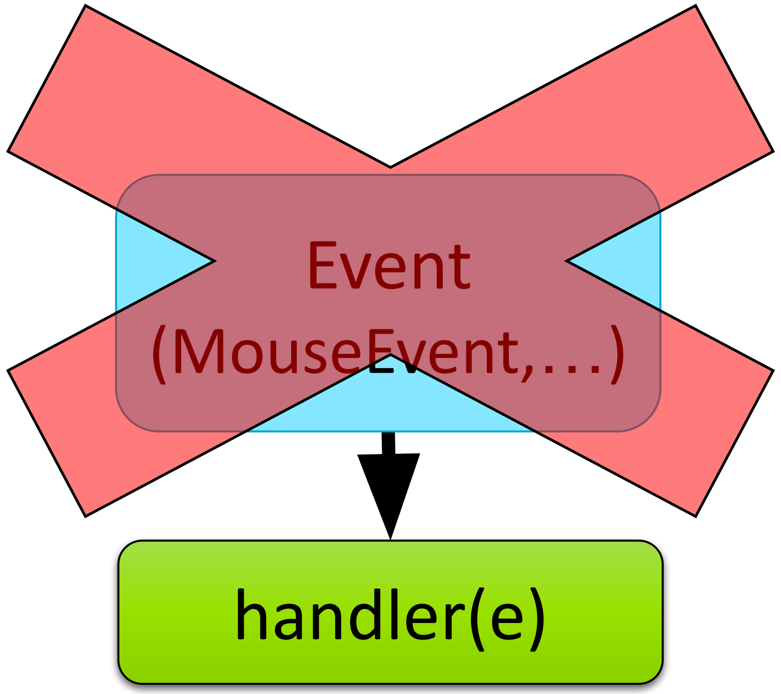
softserve

Handling events with React elements is very similar to handling events on DOM elements.

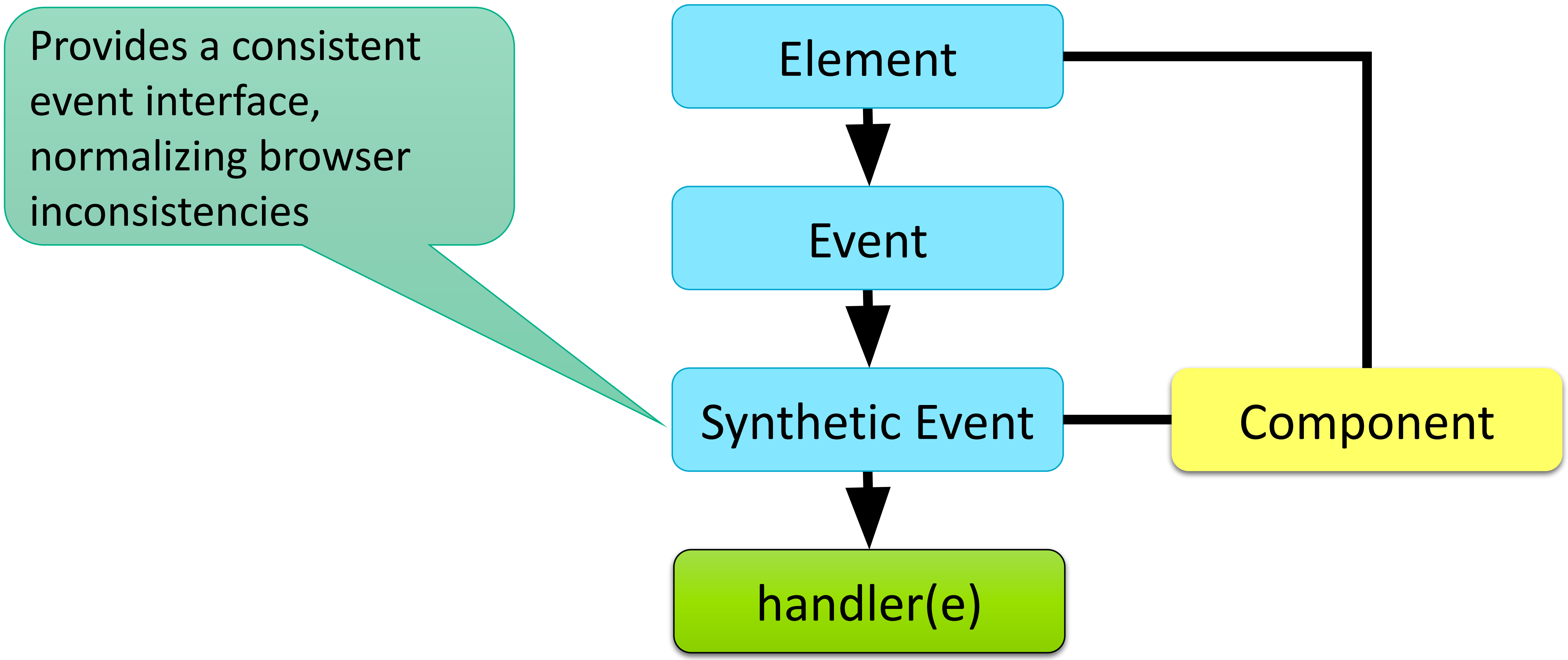
```
<button onClick={onOkClick}>
  Ok
</button>
```

React events are named using camelCase, instead of a lowercase

With JSX you pass a function as the event handler, rather than a string



softserve


Provides a consistent event interface, normalizing browser inconsistencies
Element
Event
Synthetic Event
Component
handler(e)

softserve