Fragments

A common pattern in React is for a component to return multiple elements. Fragments let you group a list of children without adding extra nodes to the DOM.

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

There is also a new short syntax for declaring them.

Motivation

A common pattern is for a component to return a list of children. Take this example React snippet:

```
class Table extends React.Component {
  render() {
    return (
      <table>
        <tr>
          <Columns />
        </tr>
      </table>
    );
  }
}
```

`<Columns />` would need to return multiple `<td>` elements in order for the rendered HTML to be valid. If a parent div was used inside the `render()` of `<Columns />`, then the resulting HTML will be invalid.

```
class Columns extends React.Component {
  render() {
    return (
      <div>
        <td>Hello</td>
        <td>World</td>
      </div>
    );
  }
}
```

results in a `<Table />` output of:

INSTALLATION
MAIN CONCEPTS
ADVANCED GUIDES
Accessibility
Code-Splitting
Context
Error Boundaries
Forwarding Refs
Fragments
Higher-Order Components
Integrating with Other Libraries
JSX In Depth
Optimizing Performance
Portals
Profiler
React Without ES6
React Without JSX
Reconciliation
Refs and the DOM
Render Props
Static Type Checking
Strict Mode
Typechecking With PropTypes
Uncontrolled Components
Web Components
API REFERENCE
HOOKS
TESTING
CONCURRENT MODE (EXPERIMENTAL)
CONTRIBUTING
FAQ

```
<table>
  <tr>
    <div>
      <td>Hello</td>
      <td>World</td>
    </div>
  </tr>
</table>
```

Fragments solve this problem.

Usage

```
class Columns extends React.Component {
  render() {
    return (
      <React.Fragment>
        <td>Hello</td>
        <td>World</td>
      </React.Fragment>
    );
  }
}
```

which results in a correct `<Table />` output of:

```
<table>
  <tr>
    <td>Hello</td>
    <td>World</td>
  </tr>
</table>
```

Short Syntax

There is a new, shorter syntax you can use for declaring fragments. It looks like empty tags:

```
class Columns extends React.Component {
  render() {
    return (
      <>
        <td>Hello</td>
        <td>World</td>
      </>
    );
  }
}
```

You can use `<></>` the same way you'd use any other element except that it doesn't support keys or attributes.

Keyed Fragments

Fragments declared with the explicit `<React.Fragment>` syntax may have keys. A use case for this is mapping a collection to an array of fragments — for example, to create a description list:

INSTALLATION
MAIN CONCEPTS
ADVANCED GUIDES
Accessibility
Code-Splitting
Context
Error Boundaries
Forwarding Refs
Fragments
Higher-Order Components
Integrating with Other Libraries
JSX In Depth
Optimizing Performance
Portals
Profiler
React Without ES6
React Without JSX
Reconciliation
Refs and the DOM
Render Props
Static Type Checking
Strict Mode
Typechecking With PropTypes
Uncontrolled Components
Web Components
API REFERENCE
HOOKS
TESTING
CONCURRENT MODE (EXPERIMENTAL)
CONTRIBUTING
FAQ

```jsx
function Glossary(props) {
  return (
    <dl>
      {props.items.map(item => (
        // Without the `key`, React will fire a key warning
        <React.Fragment key={item.id}>
          <dt>{item.term}</dt>
          <dd>{item.description}</dd>
        </React.Fragment>
      ))}
    </dl>
  );
}
```

`key` is the only attribute that can be passed to `Fragment`. In the future, we may add support for additional attributes, such as event handlers.

Live Demo

You can try out the new JSX fragment syntax with this CodePen.

Edit this page

INSTALLATION
MAIN CONCEPTS
ADVANCED GUIDES
Accessibility
Code-Splitting
Context
Error Boundaries
Forwarding Refs
Fragments
Higher-Order Components
Integrating with Other Libraries
JSX In Depth
Optimizing Performance
Portals
Profiler
React Without ES6
React Without JSX
Reconciliation
Refs and the DOM
Render Props
Static Type Checking
Strict Mode
Typechecking With PropTypes
Uncontrolled Components
Web Components
API REFERENCE
HOOKS
TESTING
CONCURRENT MODE (EXPERIMENTAL)
CONTRIBUTING
FAQ