Web Components

React と Web Components は異なる課題を解決する為に構築されました。Web Components はコンポーネントをパッケージ化して、高い再利用性を与えます。一方で React は DOM とデータを同期させる為の宣言型のライブラリを提供しています。この 2 つの目標は互いを補完しあっています。あなたは開発者として、Web Components 内で React を使用することも、React 内で Web Components を使用することも、あるいはその両方を行うこともできます。
React を使用するほとんどの人は Web Components を使用しませんが、Web Components を用いたサードパーティ製の UI コンポーネントを使用したい時などには活用できるかもしれません。

React で Web Components を使用する

```
class HelloMessage extends React.Component {
  render() {
    return <div>Hello <x-search>{this.props.name}</x-search>!</div>;
  }
}
```

補足:
Web Components はよく命令型の API を公開しています。例えば、`video` という Web Component が `play()` や `pause()` といった関数を公開しているかもしれません。Web Component が使う命令型の API にアクセスするには DOM ノードと直接やり取りするために ref を使う必要があります。サードパーティ製の Web Components を使用している場合は、Web Component のラッパとして機能する React のコンポーネントを作成するのがベストな選択でしょう。
Web Component から発された Event は React のレンダーツリーを正しく伝わってこない可能性があります。 React コンポーネント内でイベントに適切に対応するにはそのためのイベントハンドラを与える必要があります。

よくある混乱のひとつとして、Web Components が “className” の代わりに “class” を使用しているケースがあります。

```
function BrickFlipbox() {
  return (
    <brick-flipbox class="demo">
      <div>front</div>
      <div>back</div>
    </brick-flipbox>
  );
}
```

Web Components で React を使用する

```
class XSearch extends HTMLElement {
  connectedCallback() {
    const mountPoint = document.createElement('span');
    this.attachShadow({ mode: 'open' }).appendChild(mountPoint);

    const name = this.getAttribute('name');
    const url = 'https://www.google.com/search?q=' + encodeURIComponent(name);
    ReactDOM.render(<a href={url}>{name}</a>, mountPoint);
  }
}
customElements.define('x-search', XSearch);
```

補足:
Babel を使ってクラス変換を行うと上記のコードは**機能しません**。詳細や議論はこの issue を参照してください。 この問題を解決するには custom-elements-es5-adapter をあなたの web component の前に読み込む必要があります。

このページを編集する