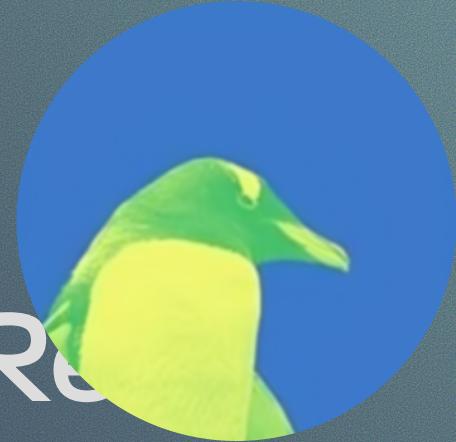


how to use "key" in React

Reactにおけるkeyの効果的な使い方について



ken7253

Frontend developer

技術記事を書いたりするのが趣味。

最近はNext.jsを使ったアプリケーションを書いています。

インターフェイス設計やアクセシビリティ・SSG関連の技術に興味があります。



 <https://github.com/ken7253>

 <https://zenn.dev/ken7253>

 <https://dairoku-studio.com>

Reactのkeyについて

リストを `map()` とかでレンダリングするときに使う `key` について。

```
const list = ['リンゴ', 'バナナ', 'ゴリラ'];

return (
  <ul>
    {
      list.map((v) => <li key={v}>{v}</li>)
    }
  </ul>
)
```

実際に起きた出来事

```
import { CheckList } from "../CheckList"; // 複数のチェックボックスを管理するコンポーネント
type Process = "before" | "after"; // 進行状況

export const App = () => {
  const [process, setProcess] = useState<Process>('before');

  return (
    <>
    {
      process === 'before'
        ? <CheckList label="開始前チェックリスト" /> // input[type="checkbox"]が複数並んだコンポーネント
        : <CheckList label="終了後チェックリスト" />
    }
    <button onClick={
      // クリックされた場合次のチェックリストに進む
      () => setProcess((prev) => prev === 'before' ? 'after' : 'before')
    } />
    </>
  )
}
```

コンポーネントを切り替えたはずなのにチェック状態が維持されてしまう

問題があった箇所

コンポーネントを出し分ける部分の書き方に問題があった。

```
{  
  process === 'before'  
    ? <CheckList label="開始前チェックリスト" /> // input[type="checkbox"]が複数並んだコンポーネント  
    : <CheckList label="終了後チェックリスト" />  
}
```

- 自分：`CheckList` コンポーネント自体を再レンダリングしてほしい
- React：差分のある `CheckList[label]` のみを更新します

対処方法

key を付けたら治った。

```
{  
  process === 'before'  
    ? <CheckList key="before" label="開始前チェックリスト" /> // input[type="checkbox"]が複数並んだコンポーネント  
    : <CheckList key="after" label="終了後チェックリスト" />  
}
```

なぜ **key** を付けてみようと思ったのか

- Reactのレンダリングにおける差分検知の仕組みを知っていたから
- 個人的には対処療法のつもりだった

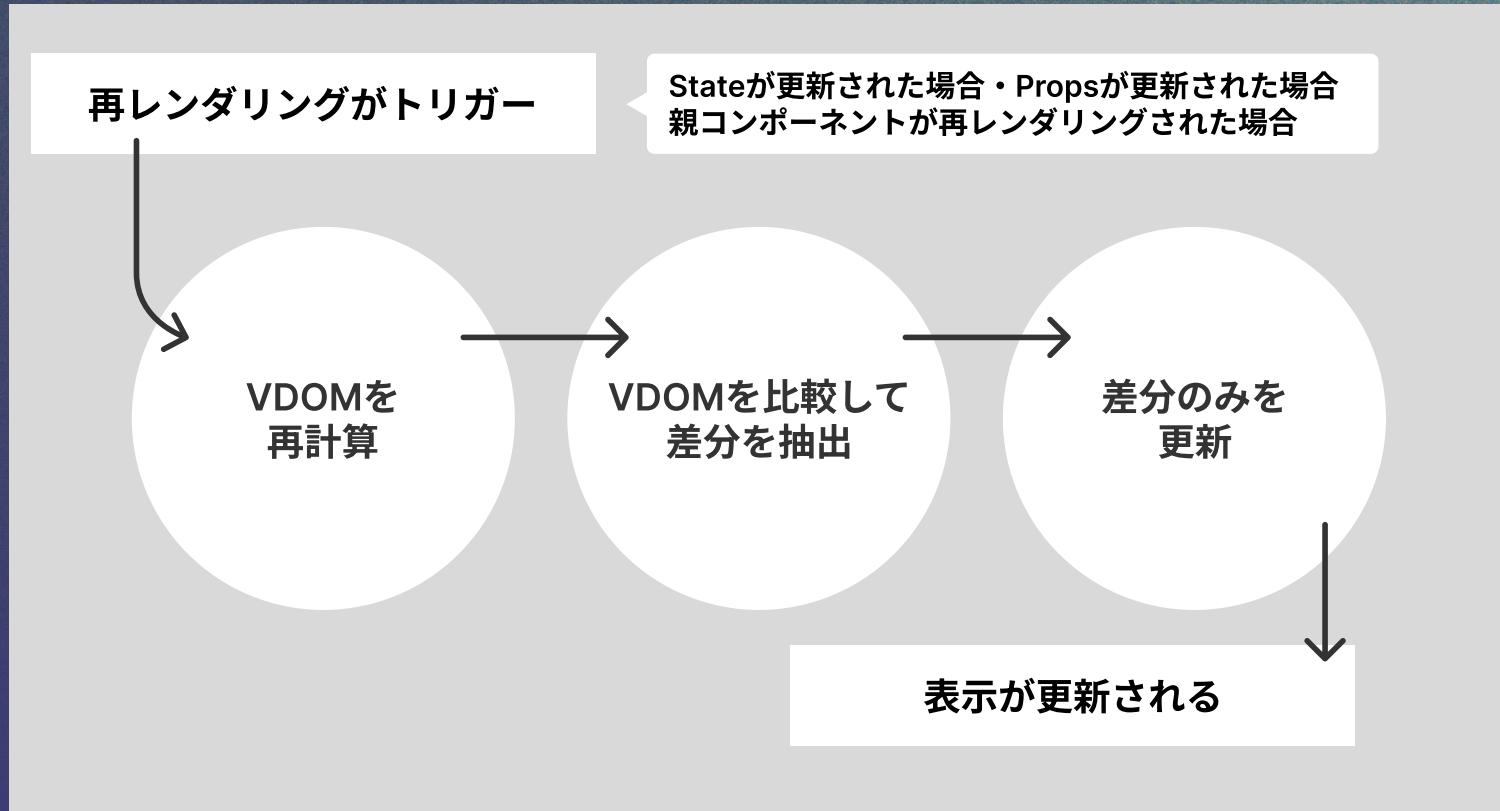
keyのもう一つの使い方

公式ドキュメントにも記載がある通り、同じようなコンポーネントを出し分ける場合に
Stateをリセットする用途としても利用できる。

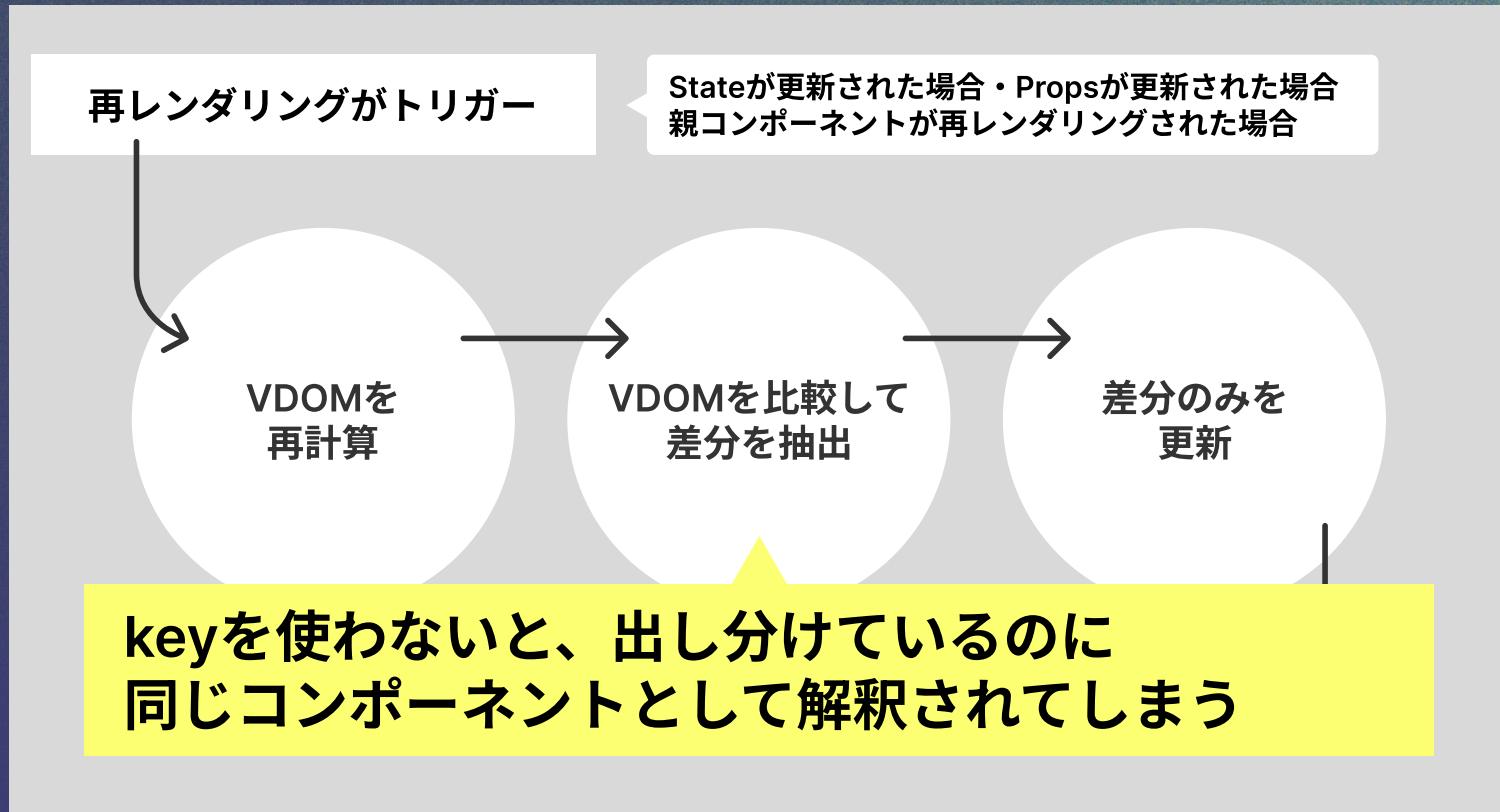
[state の保持とリセット – React](#)

Reactのレンダリングの仕組みについて

Reactのレンダリングの仕組みについて



Reactのレンダリングの仕組みについて



伝えたいこと

- `key` は識別子として使える
- Reactはレンダリングの仕組みを知ることが大切