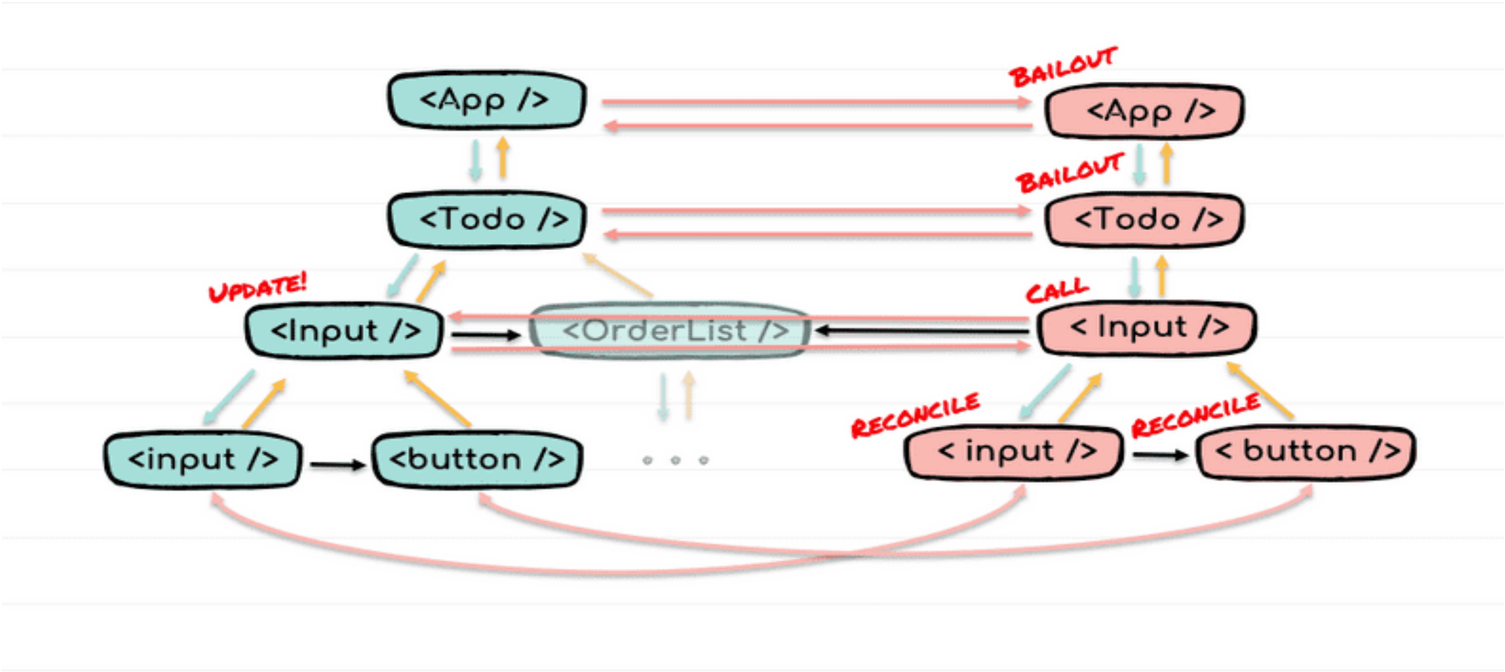
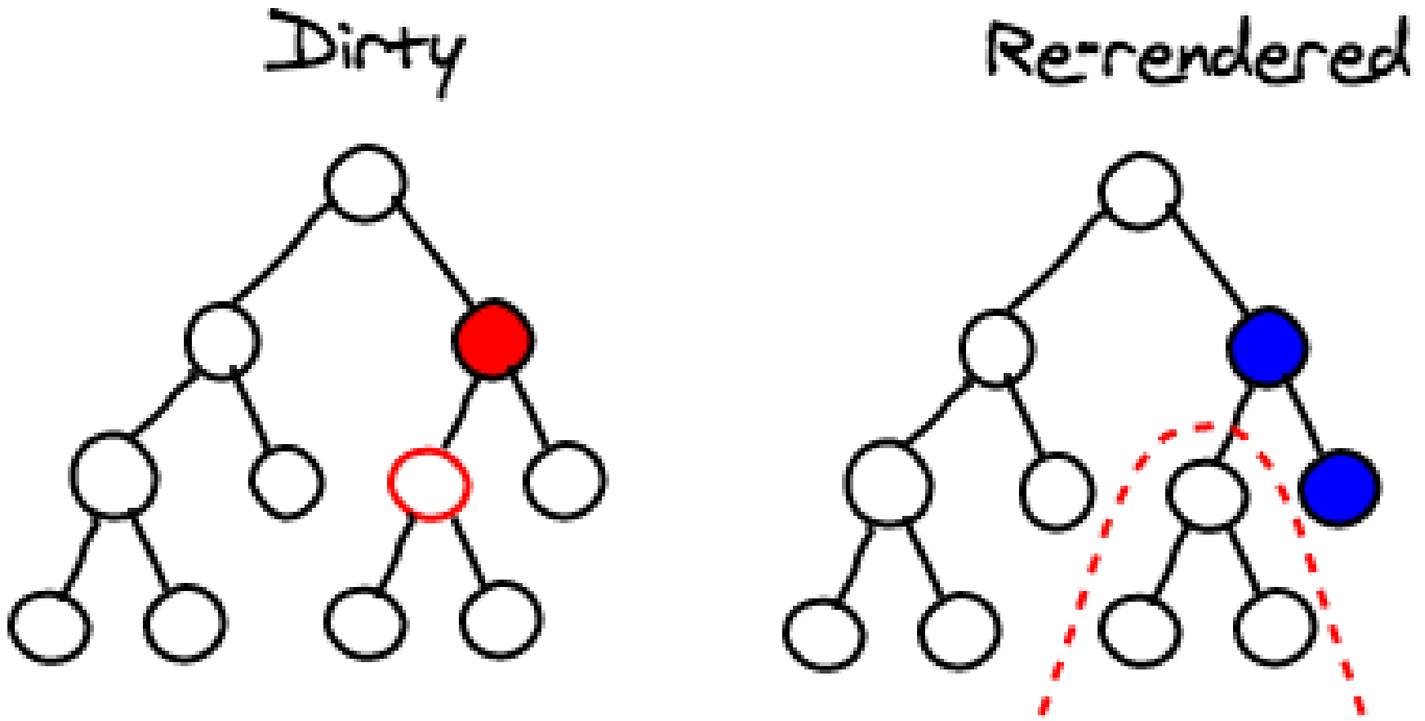
React

Reconiliation



Reconciliation 이란?



Virtual DOM을 실제 DOM과 동일하게 만드는 데 필요한 최소 변경 횟수를 알아내는 과정

재조정 또는 비교조정이라고 부른다

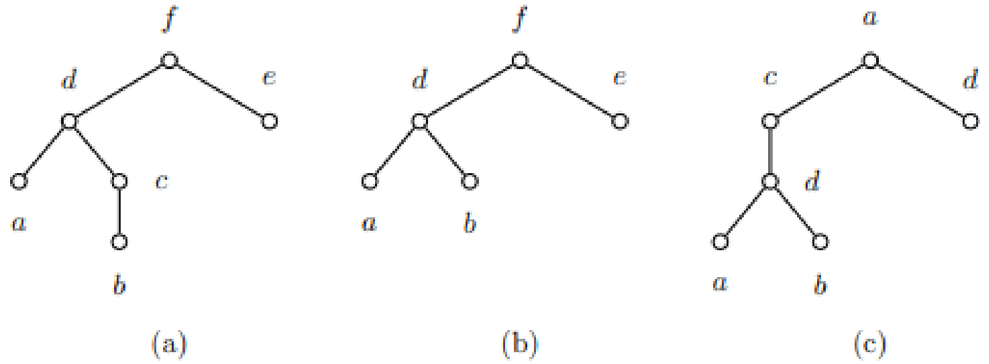
Diffing 알고리즘

A Survey on Tree Edit Distance and Related Problems

Philip Bille*
The IT University of Copenhagen
Glentevej 67, DK-2400 Copenhagen NV, Denmark.
Email: beetle@itu.dk.

Abstract

We survey the problem of comparing labeled trees based on simple local operations of deleting, inserting, and relabeling nodes. These operations lead to the tree edit distance, alignment distance, and inclusion problem. For each problem we review the results available and present, in detail, one or more of the central algorithms for solving the problem.



하나의 트리를 다른 트리로 변형시키는 가장 작은 조작 방식을 알아내는 알고리즘

n개의 엘리먼트를 갖는 트리에 대해 O(n^3)의 복잡도를 가진다

Diffing 알고리즘

n의 크기가 그리 크지 않은 웹 환경에서도 O(n^3)의 복잡도의 비용이 너무 크기 때문에
리액트에서는 O(n)의 휴리스틱 알고리즘을 사용한다

React Diffing Algorithm

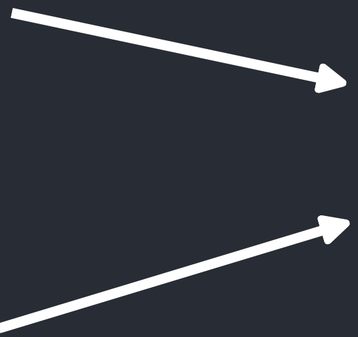
01 다른 타입을 가진 두 엘리먼트는 다른 트리를 만들어 낼 것이다

02 개발자가 제공한 key prop을 이용해, 여러번의 렌더링 속에서도
변경되지 말아야 할 자식 엘리먼트가 무엇인지를 알아낼 수 있을 것이다

Diffing 알고리즘

```
<div>
   <Counter />
</div>

<span>
   <Counter />
</span>
```



1. 다른 타입을 가진 두 엘리먼트는 다른 트리를 만들어 낼 것이다

엘리먼트의 타입이 다르기 때문에 다른 트리라고 간주한다
자식 엘리먼트가 같음에도 불구하고 모든 노드를 파괴하고
새로운 노드와 트리를 생성한다

Diffing 알고리즘

같은 타입이고 속성만 다른 경우에는 새로운 엘리먼트를 생성하지 않고
이전 엘리먼트의 속성값만 변경해준다

```
<div className="before" title="stuff" />

<div className="after" title="stuff" />

<div style={{color: 'red', fontWeight: 'bold'}} />

<div style={{color: 'green', fontWeight: 'bold'}} />
```

Diffing 알고리즘

```
<ul>
  <li>first</li>
  <li>second</li>
</ul>


<ul>
  <li>first</li>
  <li>second</li>
  <li>third</li>
</ul>
```

새로운 자식 엘리먼트가 추가될 때 마지막에 추가되는 경우에는
앞선 두 엘리먼트가 그대로임을 쉽게 알 수 있기 때문에
새로운 엘리먼트만 렌더링해주면 된다

Diffing 알고리즘

```
<ul>
  <li>Duke</li>
  <li>Villanova</li>
</ul>

<ul>
  <li>Connecticut</li>
  <li>Duke</li>
  <li>Villanova</li>
</ul>
```

맨 뒤가 아닌 곳에 새로운 자식 엘리먼트가 추가되어
기존의 순서가 달라지는 경우에는 자식 엘리먼트의 순서대로
엘리먼트를 비교해 리렌더링을 생략할 수 없다

Diffing 알고리즘

2. 개발자가 제공한 key prop을 이용해, 여러번의 렌더링 속에서도
변경되지 말아야 할 자식 엘리먼트가 무엇인지를 알아낼 수 있을 것이다

```
<ul>
  <li key="2015">Duke</li>
  <li key="2016">Villanova</li>
</ul>
```

key값을 이용해 이전 엘리먼트를 식별하고 비교를 진행한다

```
<ul>
  <li key="2014">Connecticut</li>
  <li key="2015">Duke</li>
  <li key="2016">Villanova</li>
</ul>
```

key에 index를 사용하면 안되는 이유

```
return (
  <ul>
    {items.map((item, index) => (
      <li key={index}>{item}</li>
    ))}
  </ul>
);
```

key 값이 item과 무관하게 매번 변하고 있기때문에
기존에 있던 엘리먼트들을 식별할 수 없다

key를 사용하지 않는것처럼 동작한다

key에 index를 사용하면 안되는 이유

```
<ul>
  <li key="0">Duke</li>
  <li key="1">Villanova</li>
</ul>

<ul>
  <li key="0">Connecticut</li>
  <li key="1">Duke</li>
  <li key="2">Villanova</li>
</ul>
```

```
<li key={item.id}>{item.name}</li>
```

key 값이 item과 무관하게 매번 변하고 있기때문에 기존에 있던 엘리먼트들을 식별할 수 없다

key를 사용하지 않는것처럼 동작한다

실제 개발할 때 리스트의 item들은 DB상의 ID를 가지고 있는 경우가 많기 때문에 이를 key 값으로 활용하면 된다

React의 Virtual DOM 그리고 선언형 프로그래밍

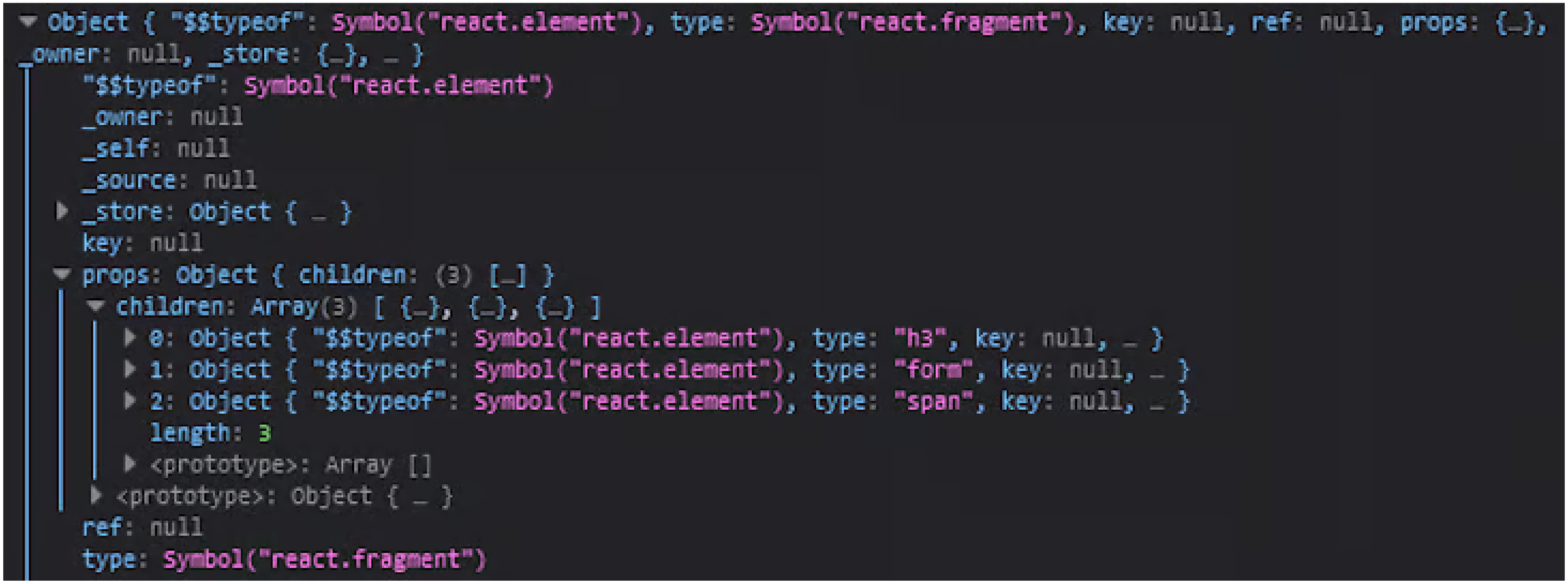
```jsx
import React from 'react';

function MyComponent() {
  return (
    <div>
      <h1>Hello, World!</h1>
      <p>Welcome to my website.</p>
    </div>
  );
}

export default MyComponent;
```

```jsx
function MyComponent() {
  return React.createElement(
    'div',
    null,
    React.createElement(
      'h1',
      null,
      'Hello, World!'
    ),
    React.createElement(
      'p',
      null,
      'Welcome to my website.'
    )
  );
}
```

React의 Virtual DOM 그리고 선언적 프로그래밍



React의 Virtual DOM 그리고 선언적 프로그래밍

명령형 프로그래밍

How

"OO님 디자인팀에서 디자인해준
페이지를 기반으로 페이지를 만든다음에,
데이터베이스에서 전체 유저 리스트에 대해,
AWS ses를 통해 메일을 보내는
스크립트를 만들어 실행해주세요."

선언형 프로그래밍

What

"OO님 프로모션 메일 보내주세요."

React의 Virtual DOM 그리고 선언형 프로그래밍

1. JSX

```jsx
function MyComponent() {
  return (
    <div>
      <h1>Hello, World!</h1>
      <p>Welcome to my website.</p>
    </div>
  );
}
```

```jsx
function MyComponent() {
  return React.createElement(
    'div',
    null,
    React.createElement(
      'h1',
      null,
      'Hello, World!'
    ),
    React.createElement(
      'p',
      null,
      'Welcome to my website.'
    )
  );
}
```

2. Reconciliation

```
ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
)
```

참고자료

https://goidle.github.io/react/in-depth-react-reconciler_1/

https://reactjs-org-ko.netlify.app/docs/reconciliation.html

https://velog.io/@yeonbot/선언적-프로그래밍-리액트