

컴포넌트 제대로 만들기

@ React 사용자를 위한 리액트 부트캠프

이현섭

mail@hyunseob.me



반갑습니다, 이현섭입니다.

- GDG Korea WebTech Organizer
- Toss Frontend Developer
- 웹 개발 4년
- hyunseob.github.io

**컴포넌트를 잘 만드는 것,
왜 중요할까요?**

**컴포넌트는 리액트 앱에서
가장 중요한 구성요소입니다.**

잘 만든 리액트 앱이란,

- 작고 단단한 컴포넌트를 만들고
- 이렇게 만들어진 컴포넌트를 유기적으로 연결한 것

잘 만든 리액트 앱이란,

- 작고 단단한 컴포넌트를 만들고
- 이렇게 만들어진 컴포넌트를 유기적으로 연결한 것

컴포넌트란 무엇일까요?

Conceptually, components are like JavaScript **functions**. They accept arbitrary inputs (called “props”) and return **React elements describing what should appear on the screen**.

이 강의에서 다룰 것들:

- State를 분리하고 컴포넌트를 추상화하기
- React.memo와 PureComponent
- Controlled / Uncontrolled 컴포넌트
- Functional Component with Hooks
- Portal Component
- Component Composition

이 강의에서 다루지 않을 것들:

- Higher Order Component(HOCs)
- 스타일링 기법 (CSS Modules, CSS-in-JS)
- State Management (Redux, MobX)

HOC는 왜 다루지 않나요?

- Hooks API
- HOC는 더 이상 좋은 접근법이 아닙니다.
- “Making Sense of React Hooks” by Dan Abramov

코딩 전 잠깐!

- StackBlitz에 가입해주세요. (GitHub 로그인 가능)
- 가급적 모든 코드를 실습해보세요.
- 주위에 못 따라오는 분이 있다면 도와주세요.
- 못 쫓아오더라도 제가 미리 짜둔 코드가 제공됩니다.

Let's code: Form

- [StackBlitz](#)

Let's code: Form

- State 없음
- render 초기 1회만 실행
- 만약 Reset이 필요하다면?

Let's code: Form with State

- email과 password를 State로 가져와 제어하기
- “초기화”버튼 만들어보기
- [StackBlitz](#)

Uncontrolled / Controlled

- 첫번째 예제가 Uncontrolled, 두번째 예제가 Controlled
- 상태를 “제어”하지 않기 때문에 Uncontrolled라고 부름
- 근본적으로 Uncontrolled 보다는 Controlled 추천

Let's code: <Input/>

- <Input/> 컴포넌트 만들기
- type, placeholder, value, onChange를 Props로 넘겨받음
- render 횟수 측정하기
- [StackBlitz](#)

Optimize: Uncontrolled

- 전의 코드에서 Fork
- <Input/> 에서 value props를 받지 않도록 만들기
- <Input/> 내부 State를 이용해 render, onChange 구현하기
- render 횟수 측정하기
- [StackBlitz](#)

Let's code: Reset with key

- key prop을 이용해 “초기화” 동작하게 만들기
- [StackBlitz](#)
- [fully uncontrolled component with a key](#)

Optimize: PureComponent

- 아까 Fork하기 전으로 돌아가서 (Controlled Component)
- <Input/>을 PureComponent로 고치기
- render 횟수 측정하기
- [StackBlitz](#)

Optim

nt

- 아까 For
- <Input/
- render
- [StackBl](#)



The image is a screenshot of a Twitter thread on a dark background. It features two tweets from Dan Abramov (@dan_abramov) dated January 16, 2017. The first tweet is a PSA about React.PureComponent. The second tweet is a follow-up explaining the reasoning behind the PSA. The interface includes engagement metrics like replies, retweets, and likes, as well as a translation link and the source of the tweet.

Dan Abramov @dan_abramov · 2017년 1월 16일
PSA: React.PureComponent can make your app slower if you use it everywhere.

12 67 149

Dan Abramov @dan_abramov
Think about it. If component's props are shallowly unequal more often than not, it re-renders anyway, but it also had to run the checks.

[트윗 번역하기](#)
오전 1:25 · 2017년 1월 16일 · [Twitter Web Client](#)

5 리트윗 35 마음에 들어요

PureComponent Bad Practice

- 이벤트 핸들러들을 인라인 함수로 고치기
- render 횟수 측정하기
- [StackBlitz](#)

PureComponent Bad Practice

- 이벤트 핸들러들은 이라이 하스르 고치기
- render 횟수 측정
- [StackBlitz](#)



```
(() => null) === (() => null)
```

PureComponent Bad Practice

- 이벤트 핸들
- render 함수
- [StackBlitz](#)

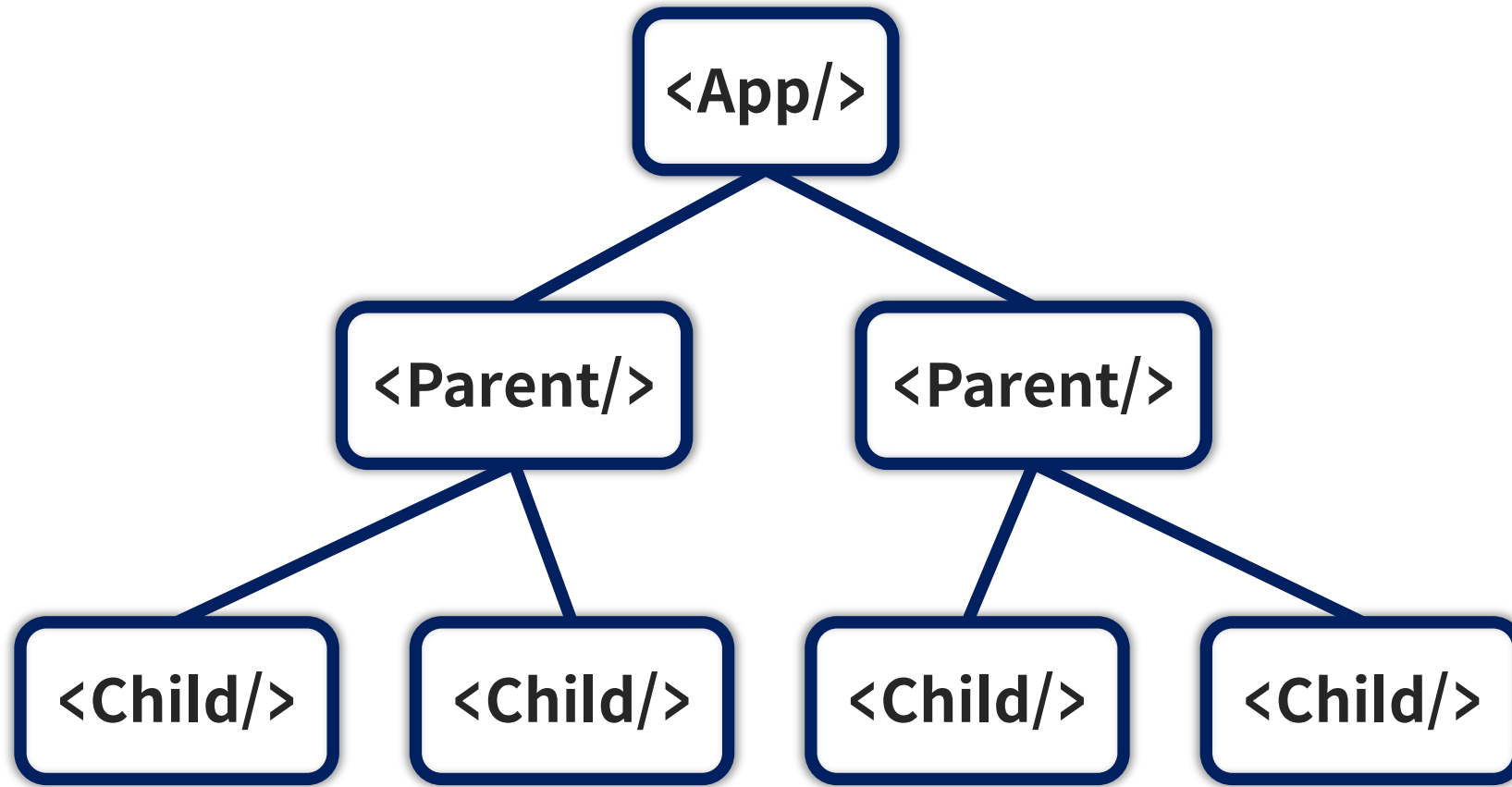


```
class Parent extends React.Component {  
  render() {  
    return (  
      <Child foo={{ bar: 'baz' }} />  
    )  
  }  
}
```

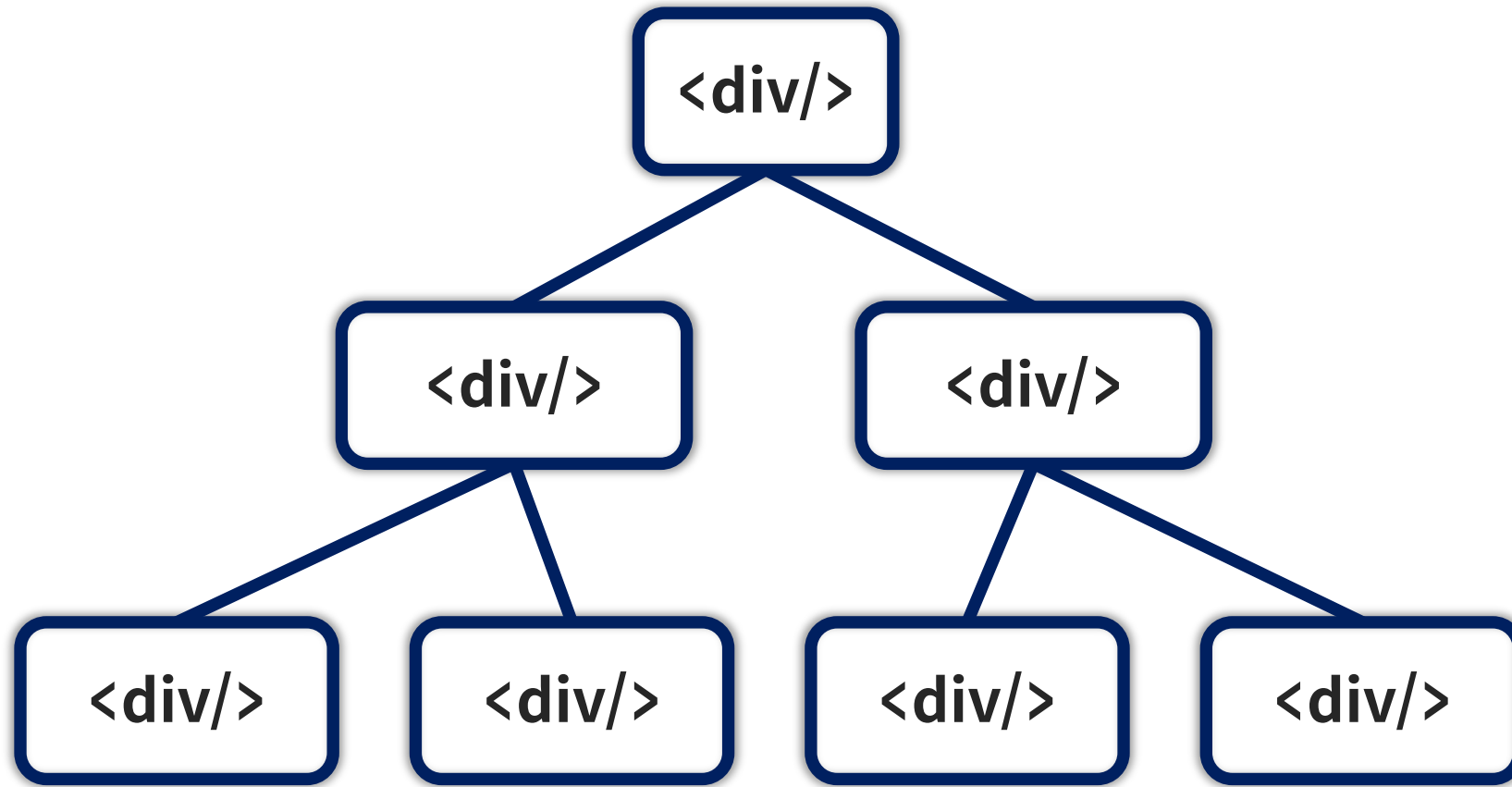
Recap

- Uncontrolled 컴포넌트는 사용자가 상태를 제어하지 않는 컴포넌트다.
- Uncontrolled 컴포넌트는 key Props를 이용해 초기화 할 수 있다.
- Controlled 컴포넌트는 사용자가 상태를 제어할 수 있는 컴포넌트다.
- PureComponent를 이용해 render를 최적화 할 수 있다.
- PureComponent를 잘못 사용하면 일반 컴포넌트보다 성능이 나쁘다.

The React Render Tree



The DOM Tree



The Problem

- [StackBlitz](#)
- position: fixed가 모두 해결하지 못한다.
- 그 외 position: absolute, Stacking Context 때문에 문제가 생긴다.

Portal Component

- index.html 파일에 Dialog container 만들기
- Dialog를 Portal Component로 만들기
- [StackBlitz](#)
- 논리적으로는 하위 컴포넌트, 시각적으로는 상위에 위치

Component Composition

- React 모듈의 단위는 컴포넌트
- 합성(Composition)은 컴포넌트를 재사용하기 위한 강력한 방법
- [Composition vs Inheritance](#)

Composition: children

- Dialog 컴포넌트를 children Props를 받도록 변경하기
- Dialog children으로 이것저것 넘겨보기
- [StackBlitz](#)

Composition: Templating

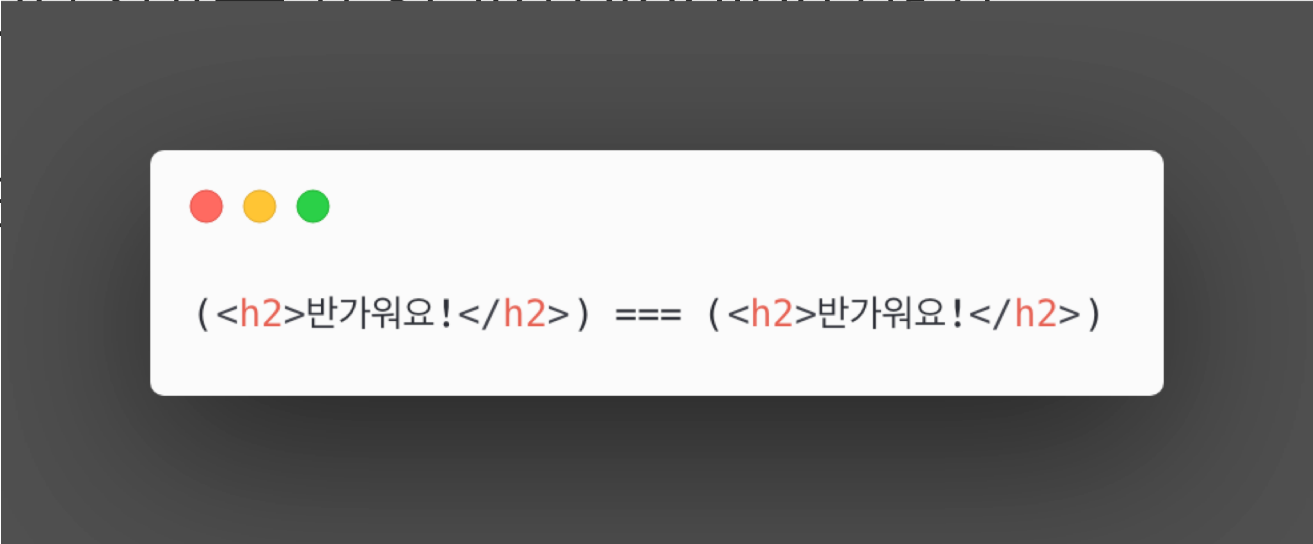
- Dialog 컴포넌트 Props에서 children을 제거하기
- Dialog 컴포넌트에 header, body, footer Props 추가하기
- 그 사이사이에 <hr/>로 구분해보기
- 만든 Props에 이것저것 넘겨보기
- [StackBlitz](#)

Composition: PureComponent

- 부모에 <input/> 만들고 State binding 하기
- Dialog를 PureComponent로 교체하기
- <input/>에 타이핑하면서 render 횟수 측정하기
- [StackBlitz](#)

Composition: PureComponent

- 부모에 `<input/>` 만들고 State binding 하기
- Dialog를 PureComponent로 만들기
- `<input/>`에 `onChange` 이벤트 연결
- [StackBlitz](#)



(`<h2>반가워요!</h2>`) === (`<h2>반가워요!</h2>`)

Optimize: Composition

- 클래스 프로퍼티로 header, body, footer로 넘길 컴포넌트 옮기기
- <input/>에 타이핑하면서 render 횟수 측정하기
- [StackBlitz](#)

Composition: Specialization

- WelcomeDialog.js 파일 생성하기
- header, body, footer WelcomeDialog로 옮기기
- WelcomeDialog를 PureComponent로 만들기
- Dialog의 render 함수 측정하기
- [StackBlitz](#)

Recap

- Portal Component는 논리적으로 하위지만 시각적으로는 상위에 위치해야 할 때 사용한다.
- React에서는 “합성”을 통해 컴포넌트를 재사용한다.
- 합성을 지원하는 컴포넌트의 경우 PureComponent가 성능이 더 나쁠 수 있다.
- 합성을 지원하는 컴포넌트의 성능 최적화를 위해 “Specialization” 할 수 있다.

Functional Component

- 컴포넌트는 “근본적으로” 함수다.
- 컴포넌트는 함수로도 작성할 수 있다.
- React 16.8 버전부터는 Hooks가 추가되어 함수형 컴포넌트만 사용해서 앱을 구성할 수 있다.
- Input을 함수형 컴포넌트로 바꾸기
- Input을 React.memo로 감싸기
- [StackBlitz](#)

Hooks: useState

- JoinForm에서 render만 남기고 다 지우기
- JoinForm을 함수형 컴포넌트로 바꾸기
- useState를 사용해서 상태를 렌더링하기
- onChange에는 인라인 함수를 넘기기
- [StackBlitz](#)

Hooks: useCallback

- useCallback을 이용해 onChange이벤트 핸들러 만들기
- handleSubmit 만들기
- handleReset 만들기
- [StackBlitz](#)

Hooks: useMemo

- Input 컴포넌트에서 render 횟수 측정하기
- Input에 accessory라는 composition용 Props 추가하기
- accessory에 동적으로 버튼 렌더하기
(value가 빈값이 아닐때만 보이도록)
- useMemo로 accessory 만들어 넘겨주기
- [StackBlitz](#)

Hooks: useRef

- Input 컴포넌트에서 useRef를 통해 DOM 객체 얻어오기
- [StackBlitz](#)

Hooks: useEffect

- useEffect를 통해 Input이 마운트 되었을때 자동으로 focus가게 하기
- Input에 autoFocus라는 Props를 추가해 이 값이 true일 때만 focus 하게 하기
- [StackBlitz](#)

Hooks: Cleanup

- autoFocus 로직 모두 지우기
- email 값이 “abcd” 일 때만 Input을 렌더링 하도록 변경하기
- Input에 useEffect hook 안의 return 문에 함수를 만들어 넘겨주기
- 그 함수 안에서 로그 찍어보기
- [StackBlitz](#)

Hooks: Custom hook

- useState를 통해 State를 생성하고,
- useCallback을 통해 onChange 이벤트를 핸들링하는 Hook 만들기
- 이렇게 만들어진 Hook을 합치기
- [StackBlitz](#)

Hooks: Custom hook(2)

- window가 resize 될 때마다 State가 변하는 Hook 만들기
- 그 hook을 컴포넌트에서 사용해서 로그 출력하기
- [StackBlitz](#)

Rules of Hooks

- Hook은 항상 컴포넌트의 Top Level Scope에서 호출되어야 함
- 즉, Hook을 if 문, for 문, 콜백 안에서 사용하면 안됨
- render 호출시마다 항상 같은 순서로 hook을 호출해야 하기 때문
- Hook을 React 함수 (함수형 컴포넌트, Custom hook)이 아닌 다른 곳에서 호출해서는 안 됨
- [ESLint](#)

Rules of

- Hook은 항상 컴포넌트
- 즉, Hook을 if 문, for 문
- render 호출시만
- Hook을 React 함수
- 다른 곳에서 호출
- [ESLint](#)

```
// 이렇게 하면 안돼요! 🙅  
function Input() {  
  // ...  
  if (a === true) {  
    useEffect(() => {  
      // ...  
    })  
  }  
  
  states.forEach(() => {  
    states.push(useState())  
  })  
}
```

호출되어야 함

됨

출해야 하기 때문

hook)이 아닌

Recap

- Hook으로 거의 모든 컴포넌트를 클래스 없이 구현할 수 있다.
- React.memo는 함수형 컴포넌트의 PureComponent다.
- useState로 함수형 컴포넌트의 State를 만들 수 있다.
- useCallback, useMemo를 통해 렌더링 최적화를 할 수 있다.
- useRef는 렌더링에 영향을 미치지 않는 값을 보관할 용도로 사용한다.
- useEffect는 클래스 컴포넌트의 Lifecycle hook과 유사하다.
- 이미 나와있는 Hook을 통해서 직접 Hook을 만들 수 있다.
- 이렇게 만든 Hook은 로직을 추상화하고, 재사용하는데 유리하다.

감사합니다.

 mail@hyunseob.me

 [@HyunSeob_](https://twitter.com/HyunSeob_)

 [hyunseob.lee.7](https://facebook.com/hyunseob.lee.7)

 [HyunSeob](https://github.com/HyunSeob)