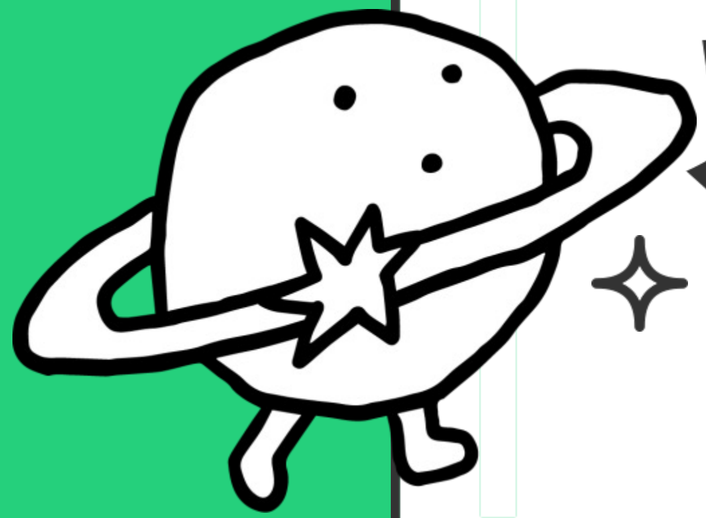
헤일리의 테코톡

커스텀 훅



우아한테크코스

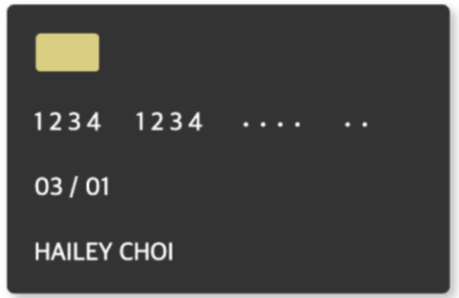
커스텀 훅의 규칙

- 조건문, 반복문 등에서 호출될 수 없고 컴포넌트 최상단에서만 호출 가능하다.
- React 컴포넌트 함수 내에서만 호출 되어야 한다.
- 함수 이름의 접두어는 반드시 'use' 로 지정해야 한다.

커스텀훅을 사용하면서 고민하고 느꼈던 것

반복되는 로직을 분리하여 재사용할 수 있게 하는 훅

Card 컴포넌트



InputField 컴포넌트

결제할 카드 번호를 입력해 주세요

본인 명의의 카드만 결제 가능합니다.

카드번호 (FieldTitle 컴포넌트)

Input 컴포넌트

1234 | 1234 | 1234 | 12ㅇ

숫자만 입력해주세요.

카드 유효기간을 입력해 주세요

월/년도(MMYY)를 순서대로 입력해 주세요.

유효기간 (FieldTitle 컴포넌트)

3 | 1ㅇ

숫자만 입력해주세요.

카드 소유자 이름을 입력해 주세요

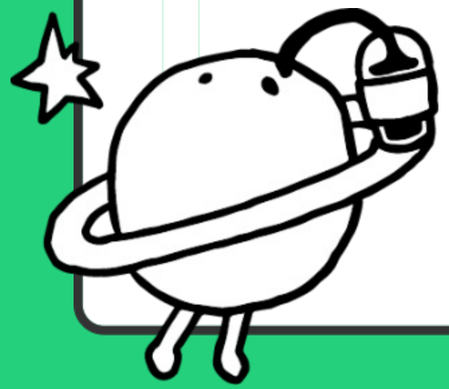
소유자 이름 (FieldTitle 컴포넌트)

HAILEY CHOIa

영대문자로만 입력해주세요.

InputForm 컴포넌트

그런데 만약 재사용되는 경우가 아니면
커스텀 훅을 사용하는 의미가 있나?



재사용하지 않아도
커스텀 훅을 만드는데 **의미가 있다!**

커스텀 훅이란?

“

커스텀 훅은 이름이 use 로 시작하는 자바스크립트 함수이며,
다른 리액트 훅을 호출할 수 있다.

”

“

커스텀 훅은 React의 특별한 기능이라기보다
기본적으로 Hook의 디자인을 따르는 관습이다.

”

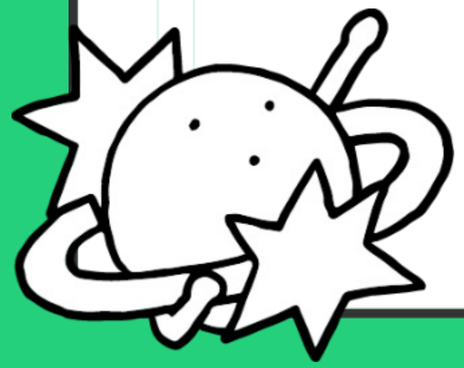
– 리액트 공식 문서–

분리한 로직 속에 리액트 훅이 있다?

> '커스텀 훅'

분리한 로직 속에 리액트 훅이 없다?

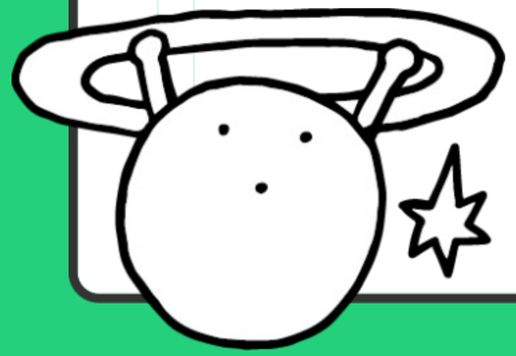
> '(JS) 함수'



```
function ExampleComponent() {
  const [username, setUsername] = useState('');
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [confirmPassword, setConfirmPassword] = useState('');
  const [isUsernameValid, setIsUsernameValid] = useState(true);
  const [isEmailValid, setIsEmailValid] = useState(true);
  const [isPasswordValid, setIsPasswordValid] = useState(true);
  const [isConfirmPasswordValid, setIsConfirmPasswordValid] = useState(true);

  const handleUsernameChange = (event) => {
    const value = event.target.value;
    setUsername(value);
    setIsUsernameValid(value.length >= 3);
  };

  const handleEmailChange = (event) => {
    const value = event.target.value;
    setEmail(value);
    setIsEmailValid(/^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(value));
  };

  const handlePasswordChange = (event) => {
    const value = event.target.value;
```

커스텀 훅이
컴포넌트의 특정 기능을 분리하고 추상화 하면서,
컴포넌트의 역할을 더 명확하게 한다

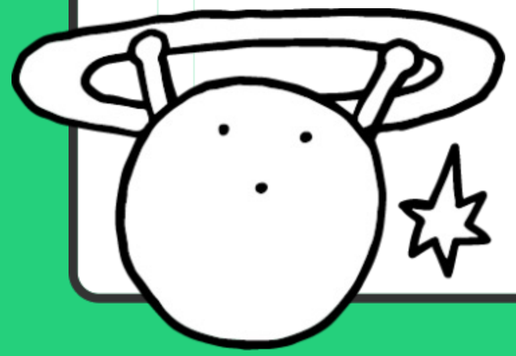
적절한 추상화 레벨을 지키고 있는가?



적절한 추상화 레벨을 지키는 것은?

필요한 정보를 제공하면서 불필요한 세부 사항을 가려내는 것

시스템을 이해하기 쉽게 만들고
개발 및 유지보수를 효율적으로 할 수 있게 한다.



```jsx
// 적절하지 않은 추상화 레벨을 갖는 커스텀 훅
const useForm = () => {
  const [formState, setFormState] = useState({
    name: '',
    age: '',
    gender: '',
  });

  const handleChange = (e) => {
    const { name, value } = e.target;
    setFormState((prevFormState) => ({
      ...prevFormState,
      [name]: value,
    }));
  };

  return {
    formState,
    handleChange,
  };
};
```

```jsx
// 입력 폼 컴포넌트
const InputForm = () => {
  const { formState, handleChange } = useForm();

  return (
    <form>
      <label>
        Name:
        <input
          type="text"
          name="name"
          value={formState.name}
          onChange={handleChange}
        />
      </label>
      <label>
        Age:
        <input
          type="number"
          name="age"
          value={formState.age}
          onChange={handleChange}
        />
      </label>
      <label>
        Gender:
```

적절한 추상화를 지키기 위해

- 코드 분리와 캡슐화
- 단일 책임 원칙 준수
- 재사용성과 결합도
- 추상화 수준 선택

```jsx
// 적절한 추상화 레벨을 유지하는 커스텀 훅
const useInput = (initialValue) => {
    const [value, setValue] = useState(initialValue);

    const handleChange = (e) => {
      setValue(e.target.value);
    };

    return {
      value,
      onChange: handleChange,
    };
  };
```

```jsx
// 입력 폼 컴포넌트
const InputForm = () => {
  const nameInput = useInput('');
  const ageInput = useInput('');
  const genderInput = useInput('');

  return (
    <form>
      <label>
        Name:
        <input type="text" {...nameInput} />
      </label>

      <label>
        Age:
        <input type="number" {...ageInput} />
      </label>

      <label>
        Gender:
```

커스텀 훅 작성시 생각해보기

1	2	3	4
적절한 추상화 지키기	불필요한 사용 방지하기	단일 책임 원칙 지키기	사이드 이펙트 고려하기

```
useYourImagination()
```

그래서 커스텀훅을 언제 사용하는가?

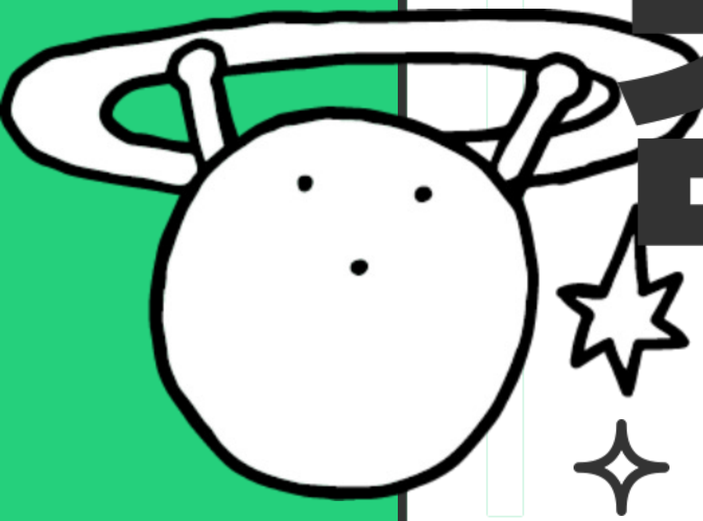
반복되는 로직을
재사용하고 싶을 때

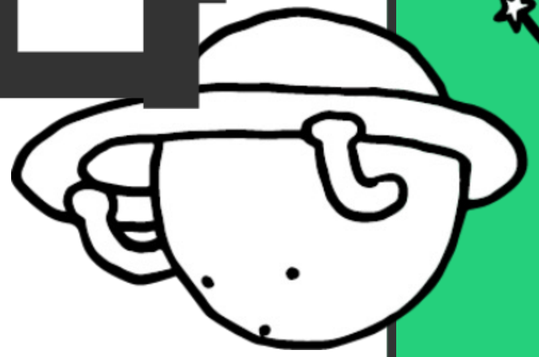
+

분리하는게 더 좋은
복잡한 로직이 있을 때

헤일리의 테코톡

감사합니다





헤일리의 테코톡

Q&A