

Multi Step Form, 중앙 집권에서 자율 조직으로

Multi Step Form 구현을 위한 Bottom-Up Approach

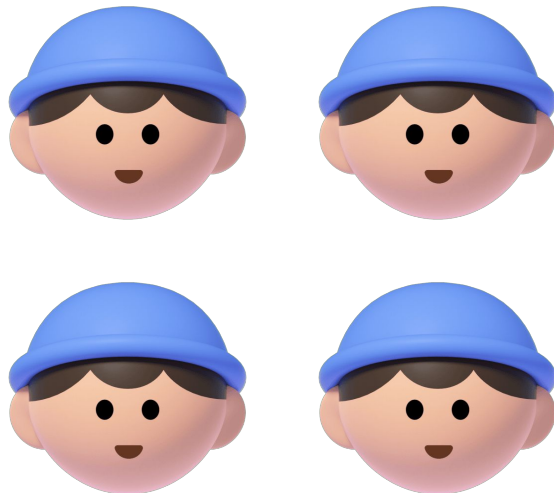
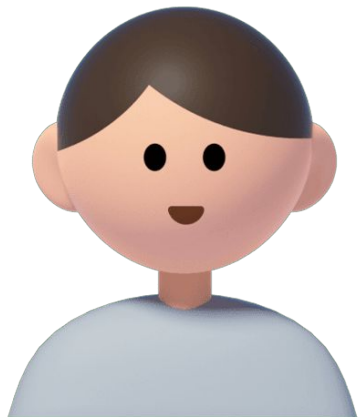
토스코어 Product Platform Team / 박종호

Frontend Diving Club 6th (2025.02.07)

Metaphor is a programmer's weapon



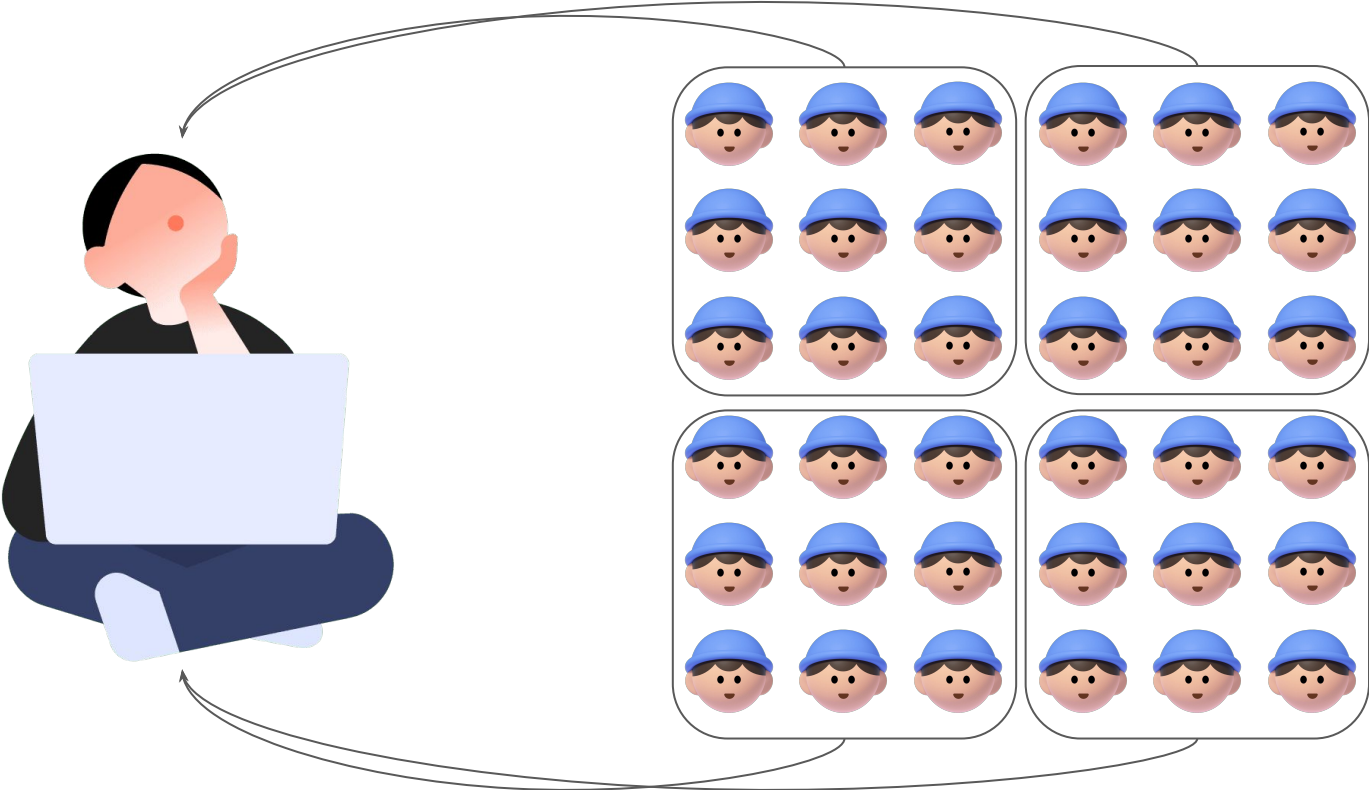
조직 구조 이야기



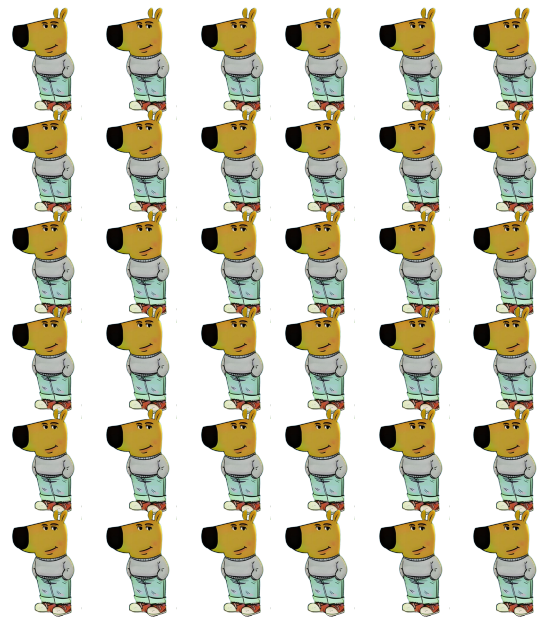
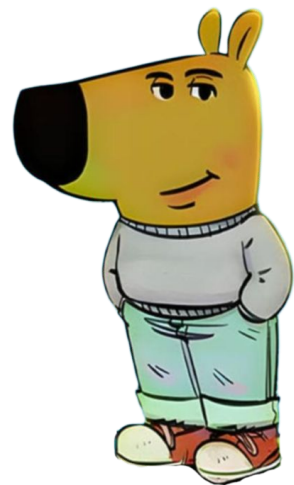
조직 구조 이야기



조직 구조 이야기



조직 구조 이야기



Multi-Step Form?

Multi Step Form

>>

⚙

표준동의모듈 수정하기

1

2

3

기본 정보필수 동의 화면선택 동의 화면

필수 약관 및 동의문 선택

그룹 생성

연결된 약관 및 동의문 추가

필수

[5910] 본인 확인 서비스 이용약관(중계기관)

매번 동의 받기

필수

[5911] 본인 확인 서비스 고유식별정보 처리 동의사항

매번 동의 받기

필수 동의 화면 UI 설정

타이틀

서비스 이용에
꼭 필요한 동의만 추렸어요

[[김토스]]를 쓰면 사용자 이름을, [[서비스명]]을 쓰면 서비스 이름을 변수로 사용할 수 있어요.
그 외 커스텀 변수를 사용하고 싶다면 저장 후에 #토스서비스메니저-request에 문의해주세요.

설명

B U ↻

통합 회원 정책이 바뀌었어요. 토스 앱과 토스뱅크 앱을 연동하려면 동의해주세요.

이전 버튼
(유효성 검사
안함)

다음 버튼
(현재 단계의 유효성 검사)

제출 버튼
(모든 단계의 유효성 검사)

< 이전

다음 >

나가기

저장

필수 동의 화면

2:07

<

내 주택담보대출
더 낮은 금리로 갈아탈 수 있어요
신용점수에 영향 없이 알아봐요.



서비스 이용에
꼭 필요한 동의만 추렸어요

통합 회원 정책이 바뀌었어요. 토스 앱과 토스뱅크
정보를 연동하려면 동의해주세요.

필수 본인 확인 서비스 이용약관(중계기관)

필수 본인 확인 서비스 고유식별정보 처리 동의
사항

필수 동의하기

다음에

Multi Step Form

```
1 const [payload, setPayload] = useState<FormPayload>({  
2   email: 'jane.doe@johndoehub.com',  
3   password: 'janedoe',  
4   name: 'Jane Doe',  
5   age: 14,  
6   gender: 'male',  
7 });
```

payload 선언

Multi Step Form

```
1 const [payload, setPayload] = useState<FormPayload>({
2   email: 'jane.doe@johndoehub.com',
3   password: 'janedoe',
4   name: 'Jane Doe',
5   age: 14,
6   gender: 'male',
7 });
```

payload 선언

```
1 {match(currentStep)
2   .with(Step.UserName, () => {
3     const [firstName, lastName] = payload.name.split(' ');
4
5     return (
6       <UserNameStep
7         value={{ firstName, lastName }}
8         onChange={value => {
9           setPayload((prev) => ({ ...prev, name: `${value.firstName} ${value.lastName}` }));
10        }}
11       />
12     );
13   })
14   .with(Step.Account, () => (
15     <AccountStep
16       value=payload
17       onChange={value => {
18         setPayload((prev) => ({ ...prev, ...value }));
19       }}
20     />
21   ))
22   .with(Step.AdditionalInfo, () => (
23     <AdditionalInfoStep
24       value=payload
25       onChange={value => {
26         setPayload((prev) => ({ ...prev, ...value }));
27       }}
28     />
29   ))
30   .exhaustive()
31 });
```

Step 컴포넌트가 변경하는
값을 payload 에 업데이트

Multi Step Form



```
1 const [payload, setPayload] = useState<FormPayload>({
2   email: 'jane.doe@johndoehub.com',
3   password: 'janedoe',
4   name: 'Jane Doe',
5   age: 14,
6   gender: 'male',
7 });
```

payload 선언



```
1 {match(currentStep)
2   .with(Step.UserName, () => {
3     const [firstName, lastName] = payload.name.split(' ');
4
5     return (
6       <UserNameStep
7         value={{ firstName, lastName }}
8         onChange={value => {
9           setPayload(prev => ({ ...prev, name: `${value.firstName} ${value.lastName}` }));
10        }}
11       />
12     );
13   })
14   .with(Step.Account, () => (
15     <AccountStep
16       value=payload
17       onChange={value => {
18         setPayload(prev => ({ ...prev, ...value }));
19       }}
20     />
21   ))
22   .with(Step.AdditionalInfo, () => (
23     <AdditionalInfoStep
24       value=payload
25       onChange={value => {
26         setPayload(prev => ({ ...prev, ...value }));
27       }}
28     />
29   ))
30   .exhaustive());
```

Step 컴포넌트가 변경하는
값을 payload 에 업데이트



```
1 {!isFirstStep && (
2   <button
3     onClick={() => {
4       match(currentStep)
5         .with(Step.Account, () => setCurrentStep(Step.UserName))
6         .with(Step.AdditionalInfo, () => setCurrentStep(Step.Account))
7         .exhaustive();
8     }}
9   >
10    이전
11  </button>
12 )}
13
14 {!isLastStep && (
15   <button
16     disabled={!isStepValid}
17     onClick={() => {
18       match(currentStep)
19         .with(Step.UserName, () => setCurrentStep(Step.Account))
20         .with(Step.Account, () => setCurrentStep(Step.AdditionalInfo))
21         .exhaustive();
22     }}
23   >
24     다음
25  </button>
26 )}
27
28 <button type="submit" disabled={!isFormValid} style={{ display: 'block' }}>
29   제출
30 </button>
```

버튼 관련
유효성 로직 추가

문제점

```
1  const [firstName, lastName] = payload.name.split(' ');
2
3  return (
4    <UserNameStep
5      value={{ firstName, lastName }}
6      onChange={(value) => {
7        setPayload((prev) => ({ ...prev, name: `${value.firstName} ${value.lastName}` }));
8      }}
9    />
10 );
```

Payload 와 Step 의 인터페이스가 다른 경우,
이를 변환하는 로직을
Form 컴포넌트에서 작성해줘야 함

문제점

```
1  const [firstName, lastName] = payload.name.split(' ');
2
3  return (
4    <UserNameStep
5      value={{ firstName, lastName }}
6      onChange={(value) => {
7        setPayload((prev) => ({ ...prev, name: `${value.firstName} ${value.lastName}` }));
8      }}
9    />
10 );
```

Payload 와 Step 의 인터페이스가 다른 경우,
이를 변환하는 로직을
Form 컴포넌트에서 작성해줘야 함

```
1  const isUserNameStepValid = !(payload.name === '');
2  const isEmailStepValid = !(payload.email === '' || payload.password === '');
3  const isAdditionalInfoStepValid = !(isNaN(payload.age ?? NaN) || payload.gender === null);
4  const isFormValid = isUserNameStepValid && isEmailStepValid && isAdditionalInfoStepValid;
5
6  const [currentStep, setCurrentStep] = useState<Step>(Step.UserName);
7  const isFirstStep = currentStep === Step.UserName;
8  const isLastStep = currentStep === Step.AdditionalInfo;
9  const isStepValid =
10    currentStep === Step.UserName
11      ? isUserNameStepValid
12      : currentStep === Step.Account
13        ? isEmailStepValid
14        : currentStep === Step.AdditionalInfo
15          ? isAdditionalInfoStepValid
16          : true;
```

버튼 유효성 로직을
Form 컴포넌트에서 작성해줘야 함

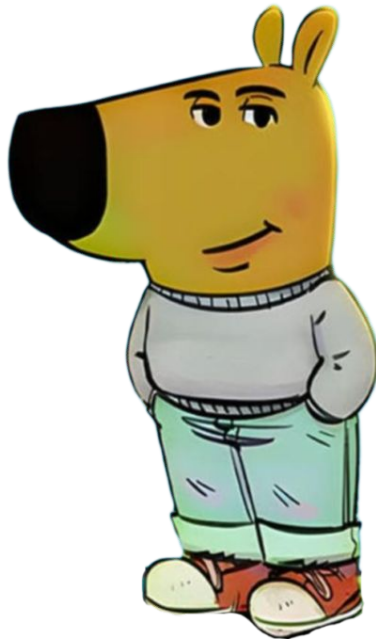
Field, Step 이 더 많아진다면 ?

Form 에 비즈니스 로직이 더 추가된다면 ?



Bottom Up Approach

- 왜 혼자서 모든걸 다 하려고 그래
- 각 팀의 성과가 조직의 성과고
- 각 팀의 성공이 조직의 성공이잖아
- 너는 팀이 필요한 예산과 목표만
관리해



Bottom Up Approach

- 폼 Payload = 모든 Step Payload 를 조합한 값
- 폼이 유효하다 = 모든 Step 이 유효하다

Bottom Up Approach - Step

- StepValue
 - Step 이 가져야 할 값
- serialize
 - Step 의 값이 Payload 에 어떻게 담겨야 하는가?
- validate
 - Step 의 값이 유효한가?



```
1 type StepValue = Record<string, unknown>;  
2  
3 type Serialize = (value: StepValue) => Record<string, unknown>;  
4  
5 type Validate = (value: StepValue) => boolean;
```

Bottom Up Approach - Step

성을 입력해주세요.

이름을 입력해주세요.

다음

제출



```
1  const value = { firstName: '', lastName: '' };  
2  
3  const serialize = ({ firstName, lastName }) => ({ name: `${firstName} ${lastName}` });  
4  
5  const validate = ({ firstName, lastName }) => firstName.length > 0 && lastName.length > 0;
```

Bottom Up Approach - Step

- Jotai 를 사용
- **valueAtom**
 - step payload 가 담긴 atom
- **serializeAtom**
 - valueAtom 에 기반해 payload 에 담길 형태로 변환한 Atom
- **validateAtom**
 - valueAtom 에 기반해 해당 값이 유효한지 판별하는 Atom

```
1 export default step<UserName, FormPayload>(UserNameStep, {
2   stepId: Step.UserName,
3   valueAtom: ({ initialValue }) => {
4     const [firstName = '', lastName = ''] = (initialValue?.name ?? '').split(' ');
5
6     return atom<UserName>({ firstName, lastName });
7   },
8   serializeAtom: ({ valueAtom }) =>
9     atom((get) => {
10       const { firstName, lastName } = get(valueAtom);
11
12       return { name: `${firstName} ${lastName}` };
13     }),
14   validateAtom: ({ valueAtom }) =>
15     atom((get) => {
16       const { firstName, lastName } = get(valueAtom);
17
18       return firstName.length > 0 && lastName.length > 0;
19     }),
20 });
21
```

Bottom Up Approach - Form

valueAtoms

- 모든 step 의
valueAtom, serializeAtom,
validateAtom 을 정의

```
1  const valueAtoms = useMemo(() => {  
2    const valueAtoms: Record<string, StepAtom> = {};  
3  
4    Object.entries(steps).forEach(([stepId, step]) => {  
5      const valueAtom = step.valueAtom({ initialValue });  
6      const serializeAtom = step.serializeAtom({ valueAtom });  
7      const validateAtom = step.validateAtom?.({ valueAtom }) ?? atom(true);  
8  
9      valueAtoms[stepId] = { valueAtom, serializeAtom, validateAtom };  
10   });  
11  
12   return valueAtoms;  
13 }, [initialValue, steps]);
```

Bottom Up Approach - Form

payloadAtom

- valueAtoms 의 값 중
serializeAtom 만 모아서
payload 를 정의

```
1  const payloadAtom = useMemo(() => {  
2    return atom<Payload>((get) => {  
3      const data = Object.values(valueAtoms).reduce((acc, step) => {  
4        return { ...acc, ...get(step.serializeAtom) };  
5      }, {});  
6  
7      return data;  
8    });  
9  }, [valueAtoms]);
```

Bottom Up Approach - Form

validateAtom

- valueAtoms 의 값 중
validateAtom 만 모아서
폼 유효성을 정의



```
1  const validateAtom = useMemo(() => {  
2    return atom<boolean>((get) => {  
3      const data = Object.values(valueAtoms).every((step) => {  
4        return get(step.validateAtom ?? atom(true));  
5      });  
6  
7      return data;  
8    });  
9  }, [valueAtoms]);
```

Bottom Up Approach - Form

```
1 {match(currentStep)
2   .with(Step.UserName, (stepId) => (
3     <StepComponent<UserName> stepId={stepId}>
4       {{{ value, onChange }}} => <UserNameStep value={value} onChange={onChange} />
5     </StepComponent>
6   ))
7   .with(Step.Account, (stepId) => (
8     <StepComponent<Account> stepId={stepId}>
9       {{{ value, onChange }}} => <AccountStep value={value} onChange={onChange} />
10    </StepComponent>
11  ))
12  .with(Step.AdditionalInfo, (stepId) => (
13    <StepComponent<AdditionalInfo> stepId={stepId}>
14      {{{ value, onChange }}} => <AdditionalInfoStep value={value} onChange={onChange} />
15    </StepComponent>
16  ))
17  .exhaustive() }
```

Step 은 자신의 값(valueAtom)만 변경
Payload 는 업데이트 하지 않음

```
1 const payload = useFormPayload();
2 const isValid = useFormValidation();
3
4 const [currentStep, setCurrentStep] = useState<Step>(Step.UserName);
5 const isFirstStep = currentStep === Step.UserName;
6 const isLastStep = currentStep === Step.AdditionalInfo;
7 const isValidStep = useFormStepValidation(currentStep);
```

폼의 유효성= 이전에 만든 validateAtom 으로 판단
스텝의 유효성= 각 스텝의 validateAtom 으로 판단

마무리

There is no silver bullet.



마무리

- 참고 자료

- <https://codesandbox.io/p/sandbox/dazzling-leftpad-4wzs3y>
- <https://codesandbox.io/p/devbox/bottom-up-approach-multi-step-form-xf7kz2>
 - (codesandbox 계정이 있다면)

- 장표

- <https://speakerdeck.com/pumpkiinbell/multi-step-form-decentralized-autonomous-organization>