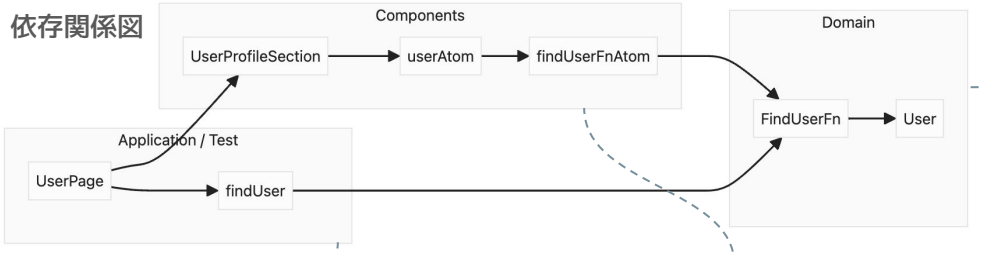
Jotaiで依存性注入したいときの我流レシピ

サンプルコード：https://github.com/kecbigmt/jotai-di-sample

目的：テスト可能性を高める・ドメイン層に外部への依存が入らないようにする

依存関係図



ドメイン層

```ts
// domain/user.ts
export type User = { name: string; imageUrl: string; bio: string };
export type FindUserFn = (userId: string) => Promise<User>;
```

ドメイン層は何にも依存しない

※今回は簡略化のためドメイン層に型しかおいていませんが、
もちろんビジネスロジックも置けます

コンポーネントテスト

```ts
// components/UserProfileSection.spec.tsx
import { UserProfileSection, findUserFnAtom } from './UserProfileSection';

const findUser = async (_userId: string) => ({
    name: 'kecy',
    imageUrl: 'https://kecy.me/image.jpg',
    bio: 'this is a bio text',
});

describe('UserProfileSection', () => {
    test('User name should be displayed in uppercase', async () => {
        const store = createStore();
        store.set(findUserFnAtom, { fn: findUser });

        render(
            <Provider store={store}>
                <UserProfileSection userId="dummy" />
            </Provider>
        );

        await waitFor(() => {
            expect(screen.getByText('KECY')).toBeInTheDocument();
        });
    });
});
```

外部から依存性注入

コンポーネント本体

```ts
// components/UserProfileSection.tsx
import { FindUserFn } from '../domain/user.ts';

export const findUserFnAtom = atom<{ fn: FindUserFn }>({
    fn: () => {
        throw new Error('function must be set');
    },
});

const userAtom = atomFamily(
    (userId: string) => {
        return atom(async (get) => {
            const findUser = get(findUserFnAtom).fn;
            const user = await findUser(userId);
            return {
                ...user,
                name: user.name.toUpperCase(), // ユーザー名は大文字にして表示
            }
        });
    },
);

type UserProfileSectionProps = { userId: string };
export const UserProfileSection = ({ userId }: UserProfileSectionProps) => {

    // userAtomに依存してユーザー情報を表示するコンポーネント

};
```

あとからDIで埋める関数の場所を用意

DIで受け取った関数を利用してデータ取得

atomをコンポーネント内で使用

🤔 ReactでのDI、どうお考えですか？