

# 型ガードの先に何があるのか

@Mita.ts #6

# ken7253

Frontend developer

ブラウザの標準化まわりを追うのが趣味

最近はReactを使ったアプリケーションを書いています。

ユーザーインターフェイスやブラウザが好き。

 <https://github.com/ken7253>

 <https://zenn.dev/ken7253>

 <https://bsky.app/profile/ken7253.bsky.social>

 <https://dairoku-studio.com>



型ガード

型ガードがされている = 良いUIか

型ガードがされている = 良いUIか

```
const UserInfo:FC = () => {
  const { data, status } = useUserData();

  if (status !== 'success' || data == null) {
    return null;
  }

  return <div>{/* ... */}</div>
}
```

悪くはない、 TypeErrorは発生しなさそう。

型ガードがされている = 良いUIか

```
const ErrorNotice = ({ children, isError, onRetry }) => {
  return (
    <>
    {isError &&
      createPortal(
        <dialog open>
          <h2>データの取得に失敗しました</h2>
          <button type="button" onClick={onRetry}>再読み込み</button>
        </dialog>,
        document.body
      )}
    {children}
    </>
  );
};
```

型ガードがされている = 良いUIか

```
const UserInfo:FC = () => {
  const { data, status } = useUserData();

  if (status === 'loading' || data == null) {
    return null;
  }

  return (
    <ErrorNotice isError={status === 'error'}>
      <div>{/* ... */}</div>
    </ErrorNotice>
  )
}
```

## 型ガードがされている = 良いUIか

```
const UserInfo:FC = () => {
  const { data, status } = useUserData();

  if (status === 'loading' || data == null) {
    return null;
  }

  return (
    <ErrorNotice isError={status === 'error'}>
      <div>{/* ... */}</div>
    </ErrorNotice>
  )
}
```

## 型ガードがされている = 良いUIか

```
const UserInfo:FC = () => {
  const { data, status } = useUserData();

  if (status === 'loading' || data == null) {
    return <Loading />;
  }

  return (
    <ErrorNotice isError={status === 'error'}>
      <div>{/* ... */}</div>
    </ErrorNotice>
  )
}
```

型ガードは最低限でしかない

型ガードがあっても  
UIは良くならない

どうする？

## Reactコンポーネントの `null`

- とりあえず返していいものではない
- 何も描画しないという宣言

汎用UIとしての通信エラーダイアログやローディングアイコンは必要

非同期処理が絡む部分は失敗する可能性を常に考えておきたい。

(おまけ) エラー処理

## エラー処理

```
const ErrorNotice = ({ children, isError, onRetry }) => {
  return (
    <>
      {isError &&
        createPortal(
          <dialog open>
            <h2>データの取得に失敗しました</h2>
            <button type="button" onClick={onRetry}>再読み込み</button>
          </dialog>,
          document.body
        )}
      {children}
    </>
  );
};
```

## エラー処理

```
const UserInfo:FC = () => {
  const { data, status, refetch } = useUserData();
  // Loading ...
  return (
    <ErrorNotice isError={status === 'error'} onRetry={refetch}>
      <div>{/* ... */}</div>
    </ErrorNotice>
  )
}
```

- ブラウザでreloadするとすべてのデータを再取得
- リトライボタンでリロードしたらAPI呼び出しのみ