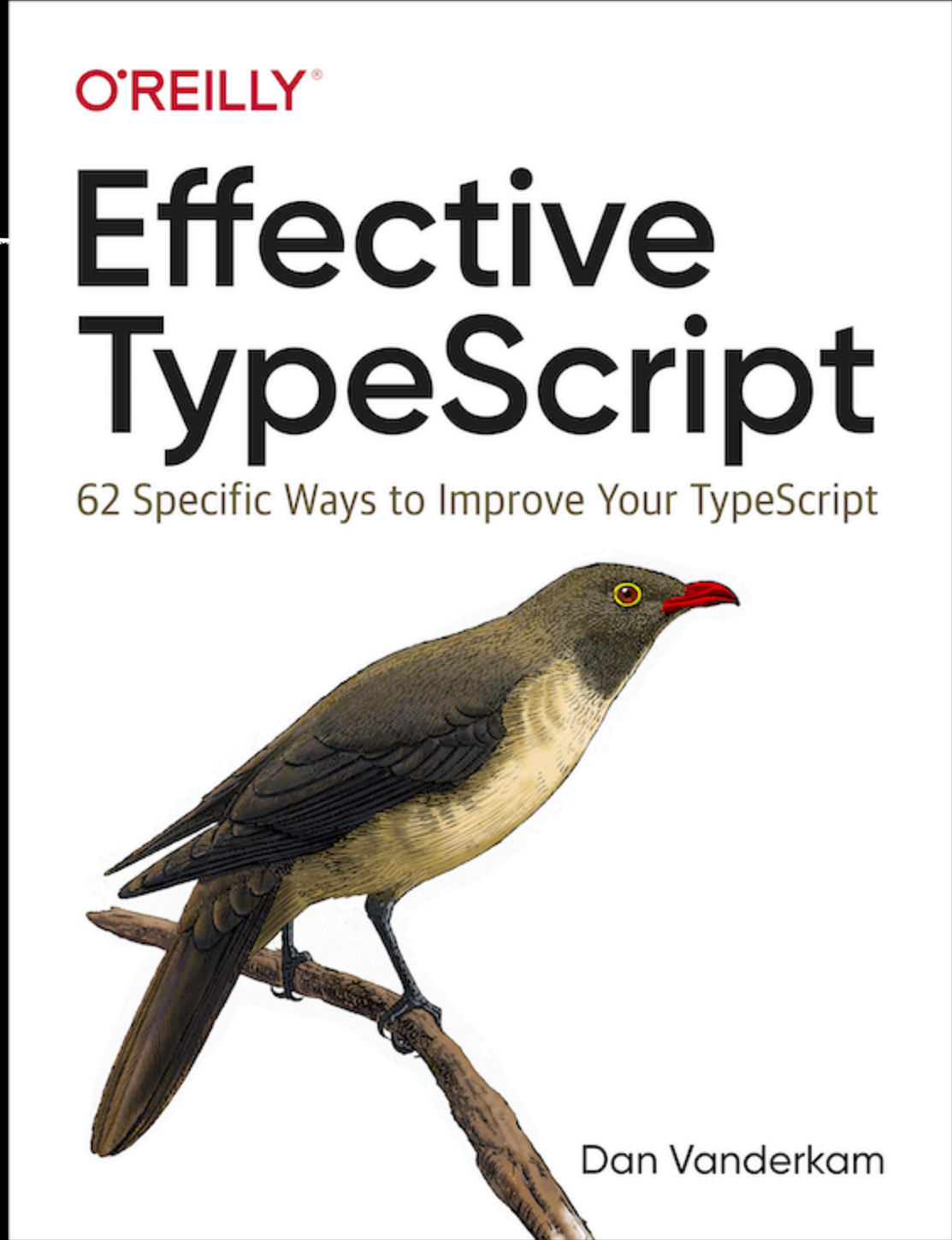
Item 28.

유효한 상태만 표현하는 타입을 지향하기

Effective TypeScript

@Bori-github(이보리)

목 차



1. 유효한 상태와 무효한 상태

타입을 **잘** 설계하면 직관적인 코드 작성이 가능

효과적인 타입을 설계하려면?

⇒ **유효한 상태**만 나타내는 타입을 만들어 내는 것이 중요!

1. 유효한 상태와 무효한 상태

✅ 유효한 상태

- 해당 타입이 가질 수 있는 유효한 값의 범위나 제약 조건
- **타입의 규칙과 제약 조건에 부합**하는 상태

⬇

올바르고 예상된 형태의 데이터를 나타내며,

코드의 실행과 로직에 문제가 없는 상태를 의미

1. 유효한 상태와 무효한 상태

🚫 무효한 상태

- 해당 타입이 가질 수 없는 유효하지 않은 값이나 상태
- 타입의 **규칙과 제약 조건에 어긋나** 프로그램이 예상 동작과는 일치하지 않는 상태

⬇

올바르지 않거나 예상치 못한 형태의 데이터를 나타내며,

코드의 실행과 로직에 문제가 있는 상태

2. 무효한 상태가 나타나는 예시 코드

웹 어플리케이션 페이지를 나타내기 위한 State 타입과 Page 함수

```tsx
interface State {
  data: string;
  isLoading: boolean;
  error?: string;
}

function Page(state: State) {
  if (state.error) {
    return `에러 발생!`;
  } else if (state.isLoading) {
    return `로딩 중..`;
  }
  return `데이터를 나타냅니다. ${state.data}`;
}
```

이때 `error` 값이 존재한다면?

- 어떤 결과가 나타날 지 예상할 수 없음
- 분기 조건이 명확하지 않기 때문

⬇

무효한 상태가 나타남

3. 유효한 상태만 표현하기

유효한 상태만 나타내기 위해 상태에 대한 타입을 더 자세히 표현해야 한다!

```ts
interface State {
  data: string;
  isLoading: boolean;
  error?: string;
}
```

```ts
// 각 상태에 따른 타입 정의
interface RequestPending {
  state: 'pending';
}
interface RequestError {
  state: 'error';
  error: string;
}
interface RequestSuccess {
  state: 'ok';
  data: string;
}

// 요청 상태를 유니온 타입으로 정의
type RequestState = RequestPending | RequestError | RequestSuccess;

interface State {
  requests: RequestState;
}
```

3. 유효한 상태만 표현하기

```tsx
function Page(state: State) {
  const {requests} = state;
  switch (requests.state) {
    case 'pending':
      return `로딩 중`;
    case 'error':
      return `에러 발생!`;
    case 'ok':
      return `데이터를 나타냅니다. ${requests.data}`;
  }
}
```

타입 설계 시, 발생하는 모든 요청 상태를 고려

- 명확한 분기 처리 가능
- 예상된 형태의 데이터를 나타냄

⬇

유효한 상태만 나타남

4. 마무리

- 무효한 상태라는 것은 예상치 못한 동작 또는 데이터를 나타내는 상태를 말합니다.

무효한 상태를 나타내지 않기 위해서는 **유효한 상태만 표현하는 타입을 지향**해야 합니다.

- 효과적인 타입을 설계하기 위해 타입 코드가 길어지거나 표현하기 어려울 수 있습니다.
- 하지만 유효한 상태만 나타내는 타입을 설계한다면 코드 작성이 쉬워지고 타입 체크가 용이해집니다.

감사합니다 :)

Effective TypeScript

@Bori-github(이보리)