Item 29.

Create Rigorous, Use Generous

엄격하게 생성하고 너그럽게 사용하라

’22. 2. 4. (Sat) edited by 신현호(@SWARVY)

Item 29. 사용할 때는 너그럽게, 생성할 때는 엄격하게

- 함수의 매개변수는 타입의 범위가 넓어도 되지만, 결과를 반환할 때는 일반적으로 타입의 범위가 더 구체적이어야 한다.
  - ↳ 타입 좁히기 단원에서도 다룬 적 있다.

예시.

```ts
declare function setCamera(camera: CameraOptions): void;
declare function viewportForBounds(bounds: LngLatBounds): CameraOptions;
```

결과가 바로 전달된다면 편리할 것...!

```ts
interface CameraOptions {
  center?: LngLat;
  zoom?: number;
  bearing?: number;
  pitch?: number;
}
type LngLat =
  { lng: number; lat: number; } |
  { lon: number; lat: number; } |
  [number, number];
```

일부만 변경할수도 있으므로 필드가 모두 선택적인 모습.

```ts
function focusOnFeature(f: Feature) {
  const bounds = calculateBoundingBox(f);
  const camera = viewportForBounds(bounds);
  setCamera(camera);
  const {center: {lat, lng}, zoom} = camera;
                 // ~~~      ... 형식에 'lat' 속성이 없습니다.
                 //      ~~~ ... 형식에 'lng' 속성이 없습니다.
  zoom;  // 타입이 number | undefined
  window.location.search = `?v=@${lat},${lng}z${zoom}`;
}
```

- 결국 이야기하고자 하는 내용은 전체적으로 타입선언시, 생성될때 너무 자유롭다는 것.
- 값을 안전하게 사용하고싶다면 유니온 타입의 요소별로 코드를 분기하는 것. (유니온 타입 사용 시)

유니온 타입의 요소별 분기를 위한 방법은, 기본 형식을 구분하는 것. (-like 활용)

```ts
interface LngLat { lng: number; lat: number; };
type LngLatLike = LngLat | { lon: number; lat: number; } |
                  [number, number];

interface Camera {
  center: LngLat;
  zoom: number;
  bearing: number;
  pitch: number;
}
interface CameraOptions extends Omit<Partial<Camera>, 'center'> {
  center?: LngLatLike;
}
type LngLatBounds =
  {northeast: LngLatLike, southwest: LngLatLike} |
  [LngLatLike, LngLatLike] |
  [number, number, number, number];
```

→ Camera: 완전하게 정의된 Camera
→ CameraOptions: 부분적으로 정의된 Camera.

```ts
interface CameraOptions {
  center?: LngLatLike;
  zoom?: number;
  bearing?: number;
  pitch?: number;
}
```

(=)

```ts
function focusOnFeature(f: Feature) {
  const bounds = calculateBoundingBox(f);
  const camera = viewportForBounds(bounds);
  setCamera(camera);
  const {center: {lat, lng}, zoom} = camera;  // 정상
  zoom;  // 타입이 number
  window.location.search = `?v=@${lat},${lng}z${zoom}`;
}
```

수많은 선택적 속성을 가지는 반환 타입과 유니온 타입은 함수를 (예시에서는 viewportForBounds) 사용하기 어렵게 한다.

- 매개변수 타입의 범위가 넓으면 사용하기 편하다.
- 반환 타입의 범위가 넓으면 불편하다.

→ 사용하기 편리한 API는 매개 변수 타입의 범위는 넓을 수 있으나 반환 타입의 엄격해짐을 알수 있다.