

item 61

# 의존성 관계에 따라 모듈 단위로 전환하기

채희수 / 23.08.27

1. 서드파티 모듈과 외부 API 호출에 대한 @types를 추가한다.

2. 모듈 단위로 점진적 마이그레이션을 한다.

순서는 모듈 간의 의존성 관계를 시각화해보고,  pahen / madge  
다른 모듈에 의존하지 않는 최하단 모듈부터 작업을 시작한다.

\* 작업 주의 사항 \*

마이그레이션 할 때는 타입 정보만 추가하고, **리팩토링하지 않는다.**  
개선이 필요한 부분은 나중에 리팩토링할 수 있도록 목록을 만든다.

# 선언되지 않은 클래스 멤버

```
1 class Greeting {
2   constructor(name) {
3     this.greeting = 'Hello';
4     // ~~~~~ Property 'greeting' does not exist on type 'Greeting'
5     this.name = name;
6     // ~~~~ Property 'name' does not exist on type 'Greeting'
7   }
8   greet() {
9     return this.greeting + ' ' + this.name;
10    // ~~~~~ ~~~~~ Property ... does not exist
11  }
12 }
```

any

Property 'greeting' does not exist on type 'Greeting'. ts(2339)

View Problem (⇧F8) Quick Fix... (⌘.)

## Quick Fix...

- 💡 Declare property 'greeting'
- 💡 Add all missing members
- 💡 Add index signature for property 'greeting'


```
1 class Greeting {
2   greeting: string;
3   name: any;
4   constructor(name) {
5     this.greeting = 'Hello';
6     // ~~~~~ Property 'greeting' does not exist on type 'Greeting'
7     this.name = name;
8     // ~~~~ Property 'name' does not exist on type 'Greeting'
9   }
10  greet() {
11    return this.greeting + ' ' + this.name;
12    // ~~~~~ ~~~~~ Property ... does not exist
13  }
14 }
```

- 누락된 모든 멤버 추가를 선택하면, 선언문이 추가된다.
- 빠른 수정으로 적용된 속성을 훑어보고, any로 추론된 부분을 직접 수정한다.


# 타입이 바뀌는 값

```
1  const state = {};  
2  state.name = 'New York';  
3  // ~~~~ Property 'name' does not exist on type '{}'  
4  state.capital = 'Albany';  
5  // ~~~~~ Property 'capital' does not exist on type '{}'
```

- 객체를 선언 후, 속성을 추가하면 오류가 발생한다.
- > (item23) 한꺼번에 생성하거나, 타입 단언문을 사용할 수 있다.



```
1  const state = {  
2    name: 'New York',  
3    capital: 'Albany'  
4  }; // OK
```



```
1  interface State {  
2    name: string;  
3    capital: string;  
4  }  
5  const state = {} as State;  
6  state.name = 'New York'; // OK  
7  state.capital = 'Albany'; // OK
```

- \* 담장의 마이그레이션으로 타입 단언문을 사용했지만,  
타입 단언보다는 타입 선언을 사용하는 것이 낫다.

# .ts에서는 .js에서 작성한 JSDoc과 @ts-check가 '무효화'된다.

```
test > Js double.js ...
1 // @ts-check
2 /**
3  * @param {number} num
4  */
5 function double(num) {
6   return 2 * num;
7 }
8
9 double('trouble');
10 // ~~~~~ Argument of type '"trouble"' is not assignable to
11 //      parameter of type 'number'
```

```
test > ts double.ts ...
1 // @ts-check
2 /**
3  * @param {number} num
4  */
5 function double(num) {
6   return 2 * num;
7 }
8
9 double('trouble');
```

- .js 표시된 오류가 .ts에선 표시되지 않는다.

```
function double(num: any): number
@param num
JSDoc types may be moved to TypeScript types. ts(80004)
Quick Fix... (⌘.)
```

Quick Fix...

💡 Annotate with type from JSDoc

```
1 function double(num: number) {
2   return 2 * num;
3 }
4
5 double('trouble');
```

**모듈 단위로 타입스크립트로 전환했다면,  
마지막 단계는!...**

**3. 테스트 코드를 타입스크립트로 전환한다.**