💡 그렇다면 암묵적 타입 변환이 무조건적으로 좋지 않은 문화이자 기능인가?

암묵적 타입 변환이 무조건적으로 좋지 않은 기능이지는 않다. 암묵적 타입 변환에는 여러 가지의 장점들을 가지고 있는데, 코드를 간결하게 만들어주며, 사용에 편의성을 제공해준다. 암묵적 타입 변환은 동적 타입 언어에 유용하다.

💡 아래 설명에 따라 단축 평가를 이용하여 아래의 if문처럼 작동하는 true 값 여부를 판별하는 코드를 빈칸에 알맞게 작성해보시오. 결과도 내시길 바랍니다.

- 빈칸의 길이와 정답의 길이가 상이할 수 있습니다.

```
var isThereMessage = true;
var message ='';

if(isThereMessage) message = '멘토는 죽어있다.';

message = _______ _______ _______ ;
console.log(message);
```

```
> var isThereMessage = true; var message = ''; message = isThereMessage && '멘토는 죽어있다'; console.log(message)
멘토는 죽어있다
```

```
var person = {
  firstName : 'turtle',
  last-name : 'park'
};

console.log(person);
//-----------------------------------------------
```

```
Uncaught SyntaxError: Unexpected token '-'
> var person = {firstName : 'turtle', last-name : 'park'}; console.log(person);
var person = {firstName : 'turtle', last-name : 'park'}; console.log(person);
                                          ^

Uncaught SyntaxError: Unexpected token '-'
```

```
> var person = {firstName : 'turtle', lastName : 'park'}; console.log(person);
{ firstName: 'turtle', lastName: 'park' }
```

```
var word1 = {
  var: '',
  function: ''
};

console.log(word1);
```

```
> var word1 = { var: '', fuction : '' }; console.log(word1);
{ var: '', fuction: '' }
```

```
//프로퍼티 키 동적 생성
var objES5 = {}
var keyES5 = 'ES5'
objES5[keyES5] = 'world';

console.log(objES5);
```

```
> var objES5 = {};var keyES5 = 'ES5'; objES5[keyES5] = 'world'; console.log(objES5);
{ ES5: 'world' }
```

```
//계산된 프로퍼티 이름
var keyES6 = 'HELL';
var objES6 = {[keyES6]: 'o'};

console.log(objES6);
```

```
> var keyES6 = 'HELL'; var objES6 = {[keyES6]: 'o'}; console.log(objES6);
{ HELL: 'o' }
```

```
var emptyObj = {
'' : ''
};
console.log(emptyObj);
```

```
> var emptyObj = {'' : ''}; console.log(emptyObj);
{ '': '' }
```

```
var numObj = {
1 : 0,
2 : 1,
3 : 2
};
console.log(numObj);
```

```
> var numObj = { 1:0, 2:1, 3:2 }; console.log(numObj);
{ '1': 0, '2': 1, '3': 2 }
```

```
var duplicateObj = {
  name : 'park',
  name : 'kim'
};
console.log(duplicateObj);
```

```
> var duplicateObj = { name : 'park', name : 'kim'}; console.log(duplicateObj);
{ name: 'kim' }
```

브라우저 환경과 Nodejs 환경을 준비하고 아래의 코드를 돌려봅시다.

```
var wind = {
  'last-name' : 'park',
  1: 10
};

wind.'last-name';
wind.last-name;

wind[last-name];
wind['last-name'];

wind.1;
wind.'1';
wind[1];
wind['1']
```

Nord.js

```
JS test.js > ...
  1   var wind = {'last-name' : 'park', 1: 10};
  2   wind.'last-name';
  3   wind.last-name;
  4   wind[last-name];
  5   wind['last-name'];
  6   wind.1;
  7   wind.'1';
  8   wind[1];
  9   wind['1']
 10
```

```
SyntaxError: Unexpected string
    at internalCompileFunction (internal/vm:73:18)
    at wrapSafe (internal/modules/cjs/loader:1178:20)
    at Module._compile (internal/modules/cjs/loader:1220:27)
    at Module._extensions..js (internal/modules/cjs/loader:1310:10)
    at Module.load (internal/modules/cjs/loader:1119:32)
    at Module._load (internal/modules/cjs/loader:960:12)
    at executeUserEntryPoint (internal/modules/run_main:86:12)
    at <anonymous> (internal/main/run_main_module:23:47)
Process exited with code 1
```

브라우저

```
var wind = {'last-name' : 'park', 1: 10};
wind.'last-name';
wind.last-name;
wind[last-name];
wind['last-name'];
wind.1;
wind.'1';
wind[1];
wind['1']
```

Uncaught SyntaxError: Unexpected string