9장 단축 평가 (심화)

논리 연산자를 이용한 단축 평가

- || 또는 && 연산자 표현식의 평과 결과는 불리언 값이 아닐 수도 있음
- 해당 표현식은 언제나 2개의 피연산자 중 어느 한쪽으로 평가됨

```
'Cat' && 'Dog' // => Dog
```

- 위 예시 코드처럼 논리곱(&&) 연산일 때 두 연산자가 모두 true일 때 true를 반환하며 해당 연산자는 좌항에서 우항으로 평가가 진행됨
    - 이는 좌항인 Cat이 true일 때 우항인 Dog도 true 이면 해당 논리연산을 결정하는 우항인 Dog를 반환함

```
'Cat' || 'Dog' // => Cat
```

- 위 예시 코드처럼 논리합(||) 연산일 때 한 연산자가 true이면 나머지 연산자의 결과 상관없이 true를 반환함
    - 이는 좌항인 Cat이 true일 때 우항인 Dog의 진위여부 상관없이 논리연산이 결정되어 Cat이 반환됨
- 논리연산의 결과를 결정하는 피연산자를 타입 변환하지 않고 그대로 반환하며 이를 **단축 평가(short-circuit evaluation)**이라고 칭함
    - 표현식을 평가하는 도중에 평가 결과가 확정된 경우 나머지 평가 과정을 생략하는 것을 칭함

단축 평가 표현식	평가 결과
true || anything	true

단축 평가 표현식	평가 결과
false || anything	anything
true && anything	anything
false && anything	false

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);
```

어떨 때 사용하나요?

- 우린 포인터를 배우면서 '이건 이럴 때 사용합니다' 라는 것을 배우지 않음
  - 하지만 우린 불법이라 대표적인 패턴을 알려드립니다.

객체를 가리키기 기대하는 변수가 null or undefined인지 판별 후 프로퍼티를 참조할 때

- 객체는 key와 value로 이루어진 property의 집합임
  - 객체를 가르키기를 기대하는 변수의 값이 객체가 아니라 null 또는 undefined인 경우 객체의 프로퍼티를 참조하면 TypeError가 발생함 → 강제 종료 엔딩

```
var elem = null;
var value = elem.value; // -> Type Error: Cannot read property 'value' of null
```

- 위의 코드를 바꾸면 단축 평가로 바꾸면

```
var elem = null;
var value = elem && elem.value; // -> null 반환
```

```
// elem이 null 이나 undefined은 falsy로 elem이 평가됨
// elem이 truthy 값이면 elem.value로 평가됨
```

함수 매개변수에 기본값을 설정할 때

- 보통 함수를 호출할 때 인수를 전달하지 않으면 매개변수에는 undefined가 할당됨
    - 이 때 단축 평가를 사용하여 매개변수의 기본값을 설정하면 undefined로 인해 발생할 수 있는 에러가 방지됨

```
// 매개변수를 검사하는 단축평가
function getStringLength(str){
  str = str || '';
  return str.length;
}

getStringLength();  // -> 0
getStringLength('hello'); // -> 5

//ES6 매개변수의 기본값 설정
function getStringLength6(str = ''){
  return str.length;
}

getStringLength6(); // -> 0
getStringLength6('hello'); // -> 5
```

옵셔닝 체이닝 연산자

- ES11(ECMAScript2020)에서 도입된 옵셔널 체이닝 연산자 ?.는 좌항의 피연산자가 null 또는 undefined인 경우 undefined를 반환하고 그렇지 않으면 우항의 프로퍼티 참조를 이어감

```
var elem = null;
// elem이 null 또는 undefined이면 undefined를 반환하고, 그렇지 않으면 우항의 프로퍼티 참조를 이어감
var value = elem?.value;
console.log(value); // undefined
```

- &&는 좌항 피연산자가 false로 평가되는 값인 경우 좌항 피연산자를 그대로 반환하는데 좌항 피연산자가 0, '' 인 경우 객체로 판별될 수 있음

```
var str = '';
var length = str && str.length;

console.log(length); // ''
```

- ?.는 좌항 피연산자가 false로 평가되는 값이라도 null 또는 undefined가 아니면 우항 프로퍼티 참조를 이어감

```
var str = '';
var length = str?.length;

console.log(length); // 0
```

null 병합 연산자

- ES11(ECMAScript2020)에서 도입된 null 병합(nullish coalescing)연산자 ??는 좌항의 피연산자가 null 또는 undefined인 경우 우항의 피연산자를 반환하고, 그렇지 않으면 좌항의 피연산자를 반환한다.

```
var foo = null ?? 'hello'
console.log(foo) // -> hello (좌항이 null 이라서 우항을 반환)

var fooo = '' ?? 'hello'
console.log(fooo) // -> '' (좌항이 null 또는 undefined이 아니라서 좌항을 반환)
```