# Nullish Coalescing Operator (nullish coalescing operator)

## ☑ Nullish coalescing operator(nullish coalescing operator) '??'

스펙에 추가된 지 얼마 안 된 최근에 추가된 문법이다. Old Browser는 polyfill이 필요하다.

nullish coalescing operator ??를 사용하면 짧은 문법으로 여러 operand 중 그 값이 '확정되어있는' 변수를 찾을 수 있다.

a ?? b의 평가 결과는 다음과 같다.

☞ a가 null도 아니고 undefined도 아니면 a
☞ 그 외의 경우는 b

nullish coalescing operator ??없이 x = a ?? b와 동일한 동작을 하는 코드를 작성하면 다음과 같다.

```
x = (a !== null && a !== undefined) ? a : b;
```

comparison operator와 logical operator만으로 nullish coalescing operator와 같은 기능을 하는 코드를 작성하면 코드 길이가 길어진다.

또 다른 예시를 살펴보자. firstName, lastName, nickName이란 변수에 사용자 이름이나 별명을 저장하는데, 사용자가 아무런 정보도 입력하지 않는 케이스도 허용한다고 해보자.
화면엔 세 변수 중 실제 값이 있는 변수의 값을 출력하는데, 세 변수 모두 값이 없다면 'Smart'가 출력되도록 해보자.
이럴 때 nullish coalescing operator ??를 사용하면 값이 정해진 변수를 간편하게 찾아낼 수 있다.

```
let firstName = null;
let lastName = null;
let nickName = "Smart";

// null이나 undefined가 아닌 첫 번째 operand
alert(firstName ?? lastName ?? nickName ?? "Smart"); // Smart
```

## ❖ '??'와 '||'의 차이

nullish coalescing operator는 OR operator ||와 상당히 유사해 보인다. 실제로 위 예시에서 ??를 ||로 바꿔도 그 결과는 동일하기까지 하다.

그런데 두 operator 사이에는 중요한 차이점이 있다.

☞ ||는 첫 번째 truthy 값을 반환한다.
☞ ??는 첫 번째 정의된(defined) 값을 반환한다.

null과 undefined, 숫자 0을 구분지어 다뤄야 할 때 이 차이점은 매우 중요한 역할을 한다.

예시를 살펴보자.

```
height = height ?? 100;
```

height에 값이 정의되지 않은경우 height엔 100이 할당된다.

이제 ??와 ||을 비교해보자.

```
let height = 0;

alert(height || 100); // 100
alert(height ?? 100); // 0
```

height || 100은 height에 0을 할당했지만, 0을 falsy 한 값으로 취급했기 때문에 null이나 undefined를 할당한 것과 동일하게 처리한다. 따라서 height || 100의 평가 결과는 100이다.

반면 height ?? 100의 평가 결과는 height가 정확하게 null이나 undefined일 경우에만 100이 된다. 예시에선 height에 0이라는 값을 할당했기 때문에 alert 창엔 0이 출력된다.

이런 특징 때문에 height처럼 0이 할당될 수 있는 변수를 사용해 기능을 개발할 땐 ||보다 ??가 적합하다.

## ❖ operator 우선순위

??의 operator 우선순위는 5로 꽤 낮다.
따라서 ??는 =와 ? 보다는 먼저, 대부분의 operator보다는 나중에 평가된다.
그렇기 때문에 복잡한 표현식 안에서 ??를 사용해 값을 하나 선택할 땐 괄호를 추가하는 게 좋다.

```
let height = null;
let width = null;

// 괄호를 추가!
let area = (height ?? 100) * (width ?? 50);
```

```
alert(area); // 5000
```

그렇지 않으면 *가 ??보다 우선순위가 높기 때문에 *가 먼저 실행된다.

결국엔 아래 예시처럼 동작한다.

```
// 원치 않는 결과
let area = height ?? (100 * width) ?? 50;
```

??엔 JavaScript 언어에서 규정한 또 다른 제약사항이 있다.
안정성 관련 이슈 때문에 ??는 &&나 ||와 함께 사용하지 못한다.

아래 예시를 실행하면 문법 에러가 발생한다.

```
let x = 1 && 2 ?? 3; // SyntaxError: Unexpected token '??'
```

이 제약에 대해선 아직 논쟁이 많긴 하지만 사람들이 ||를 ??로 바꾸기 시작하면서 만드는 실수를 방지하고자 명세서에 제약이 추가된 상황이다.

제약을 피하려면 괄호를 사용해주자.

```
let x = (1 && 2) ?? 3; // 제대로 동작한다.

alert(x); // 2
```

☞ 요약하면, nullish coalescing operator ??를 사용하면 operand 중 '값이 할당된' 변수를 빠르게 찾을 수 있다. ??는 변수에 기본값을 할당하는 용도로 사용할 수 있다.

```
// height가 null이나 undefined인 경우, 100을 할당
height = height ?? 100;
```

☞ ??의 operator 우선순위는 대다수의 operator보다 낮고 ?와 = 보다는 높다.

☞ 괄호 없이 ??를 ||나 &&와 함께 사용하는 것은 금지되어있다.