空值合并运算符 '??'

空值合并运算符 `??` 提供了一种简短的语法，用来获取列表中第一个“已定义”的变量（译注：即值不是 `null` 或 `undefined` 的变量）。

`a ?? b` 的结果是：

- `a`，如果 `a` 不是 `null` 或 `undefined`，
- `b`，其他情况。

所以，`x = a ?? b` 是下面这个表达式的简写：

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

下面是一个更长一点的例子。

假设，我们有一个用户，变量 `firstName`、`lastName` 和 `nickName` 分别对应用户的名字、姓氏和昵称。如果用户决定不输入任何值，那么这些变量都可能是未定义的。

我们想要显示用户的名称：显示这三个变量中的一个，如果都没有设置值，则显示 “Anonymous”。

让我们使用 `??` 运算符选择第一个已定义的变量：

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

// 显示第一个不是 null/undefined 的值
alert(firstName ?? lastName ?? nickName ?? "Anonymous"); // Supercoder
```

与 || 比较

或运算符 `||` 可以与 `??` 运算符以同样的方式使用。正如 上一章 所讲的，我们可以用 `||` 替换上面示例中的 `??`，也可以获得相同的结果。

重要的区别是：

- `||` 返回第一个 **真** 值。
- `??` 返回第一个 **已定义的** 值。

当我们想将 `null/undefined` 与 `0` 区别对待时，这个区别至关重要。

例如，考虑下面这种情况：

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

如果 `height` 未定义，则将其赋值为 `100`。

让我们将其与 `||` 进行比较：

```
let height = 0;

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

在这个例子中，`height || 100` 将值为 `0` 的 `height` 视为未设置的（unset），与 `null`、`undefined` 以及任何其他假（falsy）值同等对待。因此得到的结果是 `100`。

`height ?? 100` 仅当 `height` 确实是 `null` 或 `undefined` 时才返回 `100`。因此，`alert` 按原样显示了 `height` 值 `0`。

哪种行为更好取决于特定的使用场景。当高度 `0` 为有效值时，`??` 运算符更适合。

优先级

优先级要在复杂表达式中使用 ?? 进行取值，需要考虑加括号：

因此，`??` 在大多数其他运算之后，但在 `=` 和 `?` 之前进行运算。

如果我们需要在复杂表达式中使用 `??` 进行取值，需要考虑加括号：

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

// 重要：使用括号
let area = (height ?? 100) * (width ?? 50);

alert(area); // 5000
```

否则，如果我们省略了括号，`*` 的优先级比 `??` 高，会优先执行。

运算过程将等同于下面这个表达式：

```
// 可能不正确的
let area = height ?? (100 * width) ?? 50;
```

这里还有一个相关的语言级别的限制。

出于安全原因，禁止将 `??` 运算符与 `&&` 和 `||` 运算符一起使用。

下面的代码会触发一个语法错误：

```
let x = 1 && 2 ?? 3; // Syntax error
```

这个限制无疑是值得商榷的，但是它被添加到语言规范中是为了避免编程错误，因为人们开始使用 `??` 替代 `||`。

可以明确地使用括号来解决这个问题：

```
let x = (1 && 2) ?? 3; // 起作用

alert(x); // 2
```

注意:

- 空值合并运算符 `??` 提供了一种简洁的方式获取列表中“已定义”的值。

  它被用于为变量分配默认值：

  ```
  // 当 height 的值为 null 或 undefined 时，将 height 的值设置为 100
  height = height ?? 100;
  ```

- `??` 运算符的优先级非常低，只略高于 `?` 和 `=`。
- 如果没有明确添加括号，不能将其与 `||` 或 `&&` 一起使用。