


 → [Язык JavaScript](#) → [Основы JavaScript](#)

 17 декабря 2022 г.

Оператор нулевого слияния (??)



Новая возможность

Эта возможность была добавлена в язык недавно. В старых браузерах может понадобиться [полифил](#).

Оператор нулевого слияния представляет собой два вопросительных знака `??`.

Так как он обрабатывает `null` и `undefined` одинаковым образом, то для этой статьи мы введём специальный термин. Для краткости будем говорить, что значение «определено», если оно не равняется ни `null`, ни `undefined`.

Результат выражения `a ?? b` будет следующим:

- если `a` определено, то `a`,
- если `a` не определено, то `b`.

Иначе говоря, оператор `??` возвращает первый аргумент, если он не `null/undefined`, иначе второй.

Оператор нулевого слияния не является чем-то принципиально новым. Это всего лишь удобный синтаксис, как из двух значений получить одно, которое «определено».

Вот как можно переписать выражение `result = a ?? b`, используя уже знакомые нам операторы:

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

Теперь должно быть абсолютно ясно, что делает `??`. Давайте посмотрим, где это может быть полезно.

Как правило, оператор `??` нужен для того, чтобы задать значение по умолчанию для потенциально неопределённой переменной.

Например, здесь мы отобразим `user`, если её значение не `null/undefined`, в противном случае **Аноним**:

```
1 let user;
2
3 alert(user ?? "Аноним"); // Аноним (user не существует)
```



А вот пример, когда `user` присвоено значение:

```
1 let user = "Иван";
2
3 alert(user ?? "Аноним"); // Иван (user существует)
```



Кроме этого, можно записать последовательность из операторов `??`, чтобы получить первое значение из списка, которое не является `null/undefined`.

Допустим, у нас есть данные пользователя в переменных `firstName`, `lastName` или `nickName`. Все они могут не существовать, если пользователь решил не вводить соответствующие значение.

Мы хотели бы отобразить имя пользователя, используя одну из этих переменных, или показать «Аноним», если все они `null/undefined`.

Для этого воспользуемся оператором `??`:

```
1 let firstName = null;
2 let lastName = null;
3 let nickName = "Суперкодер";
4
5 // показывает первое значение, которое определено:
6 alert(firstName ?? lastName ?? nickName ?? "Аноним"); // Суперкодер
```

Сравнение с ||

Оператор ИЛИ `||` можно использовать для того же, что и `??`, как это было показано в [предыдущей главе](#).

Например, если в приведённом выше коде заменить `??` на `||`, то будет тот же самый результат:

```
1 let firstName = null;
2 let lastName = null;
3 let nickName = "Суперкодер";
4
5 // показывает первое истинное значение:
6 alert(firstName || lastName || nickName || "Аноним"); // Суперкодер
```

Исторически сложилось так, что оператор ИЛИ `||` появился первым. Он существует с самого начала в JavaScript, поэтому разработчики долгое время использовали его для таких целей.

С другой стороны, сравнительно недавно в язык был добавлен оператор нулевого слияния `??` – как раз потому, что многие были недовольны оператором `||`.

Важное различие между ними заключается в том, что:

- `||` возвращает первое *истинное* значение.
- `??` возвращает первое *определённое* значение.

Проще говоря, оператор `||` не различает `false`, `0`, пустую строку `""` и `null/undefined`. Для него они все одинаковы, т.е. являются ложными значениями. Если первым аргументом для оператора `||` будет любое из перечисленных значений, то в качестве результата мы получим второй аргумент.

Однако на практике часто требуется использовать значение по умолчанию только тогда, когда переменная является `null/undefined`. Ведь именно тогда значение действительно неизвестно/не определено.

Рассмотрим следующий пример:

```
1 let height = 0;
2
3
```

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

- `height || 100` проверяет `height` на ложное значение, оно равно `0`, да, ложное.
 - поэтому результатом `||` является второй аргумент, т.е. `100`.
- `height ?? 100` проверяет, что переменная `height` содержит `null/undefined`, а поскольку это не так,
 - то результатом является сама переменная `height`, т.е. `0`.

На практике нулевая высота часто является вполне нормальным значением, которое не следует заменять значением по умолчанию. Таким образом, `??` здесь как раз работает так, как нужно.

Приоритет

Приоритет оператора `??` такой же, как и у `||`. Они оба равны **3** в [таблице на MDN](#).

Это означает, что, как и `||`, оператор нулевого слияния `??` вычисляется до `=` и `?`, но после большинства других операций, таких как `+`, `*`.

Так что, в выражениях такого вида понадобятся скобки:

```
1 let height = null;
2 let width = null;
3
4 // важно: используйте круглые скобки
5 let area = (height ?? 100) * (width ?? 50);
6
7 alert(area); // 5000
```



Иначе, если опустить скобки, оператор `*` выполнится первым, так как у него приоритет выше, чем у `??`, и это приведёт к неправильным результатам.

```
1 // без скобок
2 let area = height ?? 100 * width ?? 50;
3
4 // ...сработает вот так (совсем не как нам нужно):
5 let area = height ?? (100 * width) ?? 50;
```

Использование `??` вместе с `&&` или `||`

По соображениям безопасности JavaScript запрещает использование оператора `??` вместе с `&&` и `||`, если приоритет явно не указан при помощи круглых скобок.

Выполнение следующего кода приведёт к синтаксической ошибке:

```
1 let x = 1 && 2 ?? 3; // Синтаксическая ошибка
```



Это, безусловно, спорное ограничение было добавлено в спецификацию языка с целью избежать программные ошибки, когда люди начнут переходить с `||` на `??`.

Используйте скобки, чтобы обойти это ограничение:



```
1 let x = (1 && 2) ?? 3; // Работает без ошибок
2
3 alert(x); // 2
```

Итого

- Оператор нулевого слияния `??` — это быстрый способ выбрать первое «определённое» значение из списка. Используется для присвоения переменным значений по умолчанию:

```
1 // будет height=100, если переменная height равна null или undefined
2 height = height ?? 100;
```

- Оператор `??` имеет очень низкий приоритет, лишь немного выше, чем у `?` и `=`, поэтому при использовании его в выражении, скорее всего, потребуются скобки.
- Запрещено использовать вместе с `||` или `&&` без явно указанного приоритета, то есть без скобок.



Предыдущий урок

Следующий урок



Поделиться



Карта учебника

Проводим курсы по JavaScript и фреймворкам.



Комментарии

- Если вам кажется, что в статье что-то не так - вместо комментария напишите [на GitHub](#).
- Для одной строки кода используйте тег `<code>`, для нескольких строк кода — тег `<pre>`, если больше 10 строк — ссылку на песочницу ([plnkr](#), [JSBin](#), [codepen...](#))
- Если что-то непонятно в статье — пишите, что именно и с какого места.