28-го октября 2020

Оператор объединения с null '??'

Новая возможность

Эта возможность была добавлена в язык недавно. В старых браузерах может понадобиться полифил.

В этой статье мы будем говорить, что значение выражения «определено», если оно отличается от `null` или `undefined`.

Оператор объединения с null представляет собой два вопросительных знака `??`.

Результат выражения `a ?? b` будет следующим:

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

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

Оператор объединения с null не является чем-то принципиально новым. Это всего лишь удобный синтаксис, как из двух значений получить одно «определённое».

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

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

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

Например, в следующем примере, если переменная `user` не определена, покажем модальное окно с надписью `Аноним`:

```
let user;

alert(user ?? "Аноним"); // Аноним
```

Конечно, если бы переменная `user` содержала любое значение, кроме `null/undefined`, то мы бы увидели его:

```
let user = "Иван";

alert(user ?? "Аноним"); // Иван
```

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 быть неопределёнными, если отсутствует соответствующая информация.

Выведем имя пользователя, используя одну из этих переменных, а в случае если все они не определены, то покажем «Аноним».

Для этого воспользуемся оператором `??`:

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

// показывает первое определённое значение:
alert(firstName ?? lastName ?? nickName ?? "Аноним"); /
```

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:

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

// показывает первое истинное значение:
alert(firstName || lastName || nickName || "Аноним"); /
```

Оператор ИЛИ `||` существует с самого появления JavaScript, поэтому ранее для решения похожих задач разработчики использовали именно его.

С другой стороны, сравнительно недавно в язык был добавлен оператор объединения с null `??` 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 одинаковые, 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.

Например, рассмотрим следующий пример:

```
let height = 0;

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

- `height || 100` проверяет, имеет ли переменная `height` ложное значение, что так и есть,
  - 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`.

Если нулевая высота является «нормальным» значением, которое не должно заменяться значением по умолчанию, то оператор `??` делает как раз то, что нужно.

Приоритет

Оператор `??` имеет довольно низкий приоритет: `5`, согласно таблице на MDN. Таким образом, оператор `??` вычисляется до `=` и `?`, но после большинства других операций, таких как `+`, `*`.

Из этого следует, что если нужно выбрать значение при помощи оператора `??` вместе с другими операторами в выражении, следует добавить круглые скобки:

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

// важно: используйте круглые скобки
let area = (height ?? 100) * (width ?? 50);

alert(area); // 5000
```

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

```
// без круглых скобок
let area = height ?? 100 * width ?? 50;

// ...то же самое, что предыдущее выражение (вероятно,
let area = height ?? (100 * width) ?? 50;
```

Использование ?? вместе с && или ||

По соображениям безопасности JavaScript запрещает использование оператора `??` вместе с `&&` и `||`, если только приоритет явно не указан в круглых скобках.

Выполнение следующего кода приведёт к синтаксической ошибке:

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

Это довольно спорное ограничение, которое было описано в спецификации языка, чтобы избежать ошибок при замене оператора `||` на `??`.

Используйте круглые скобки, чтобы обойти это ограничение:

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

alert(x); // 2
```

Итого

- Оператор объединения с null `??` — это быстрый способ выбрать первое «определённое» значение из списка.

  Используется для присвоения переменным значений по умолчанию:

  ```
  // будет height=100, если переменная height равна nul
  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 указанных круглых скобок.

Комментарии

перед тем как писать...

Раздел

Основы JavaScript

Навигация по уроку

Сравнение с ||

Приоритет

Итого

Комментарии

Поделиться

Редактировать на GitHub