

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

 16 августа 2022 г.

# Преобразование типов

Чаще всего операторы и функции автоматически приводят переданные им значения к нужному типу.

Например, `alert` автоматически преобразует любое значение к строке. Математические операторы преобразуют значения к числам.

Есть также случаи, когда нам нужно явно преобразовать значение в ожидаемый тип.

## Пока что мы не говорим об объектах

В этой главе мы не касаемся объектов. Сначала мы разберём преобразование примитивных значений.

Мы разберём преобразование объектов позже, в главе [Преобразование объектов в примитивы](#).

## Строковое преобразование

Строковое преобразование происходит, когда требуется представление чего-либо в виде строки.

Например, `alert(value)` преобразует значение к строке.

Также мы можем использовать функцию `String(value)`, чтобы преобразовать значение к строке:

```

1 let value = true;
2 alert(typeof value); // boolean
3
4 value = String(value); // теперь value это строка "true"
5 alert(typeof value); // string
  
```



Преобразование происходит очевидным образом. `false` становится `"false"`, `null` становится `"null"` и т.п.

## Численное преобразование

Численное преобразование происходит в математических функциях и выражениях.

Например, когда операция деления `/` применяется не к числу:

```

1 alert( "6" / "2" ); // 3, строки преобразуются в числа
  
```



Мы можем использовать функцию `Number(value)`, чтобы явно преобразовать `value` к числу:



```
1 let str = "123";
2 alert(typeof str); // string
3
4 let num = Number(str); // становится числом 123
5
6 alert(typeof num); // number
```

Явное преобразование часто применяется, когда мы ожидаем получить число из строкового контекста, например из текстовых полей форм.

Если строка не может быть явно приведена к числу, то результатом преобразования будет `NaN`. Например:



```
1 let age = Number("Любая строка вместо числа");
2
3 alert(age); // NaN, преобразование не удалось
```

Правила численного преобразования:

Значение	Преобразуется в...
undefined	NaN
null	0
true / false	1 / 0
string	Пробельные символы (пробелы, знаки табуляции <code>\t</code> , знаки новой строки <code>\n</code> и т. п.) по краям обрезаются. Далее, если остаётся пустая строка, то получаем <code>0</code> , иначе из непустой строки «считывается» число. При ошибке результат <code>NaN</code> .

Примеры:



```
1 alert( Number(" 123 ") ); // 123
2 alert( Number("123z") ); // NaN (ошибка чтения числа на месте символа "z"
3 alert( Number(true) ); // 1
4 alert( Number(false) ); // 0
```

Учтите, что `null` и `undefined` ведут себя по-разному. Так, `null` становится нулём, тогда как `undefined` приводится к `NaN`.

Большинство математических операторов также производит данное преобразование, как мы увидим в следующей главе.

## Логическое преобразование

Логическое преобразование самое простое.

Происходит в логических операциях (позже мы познакомимся с условными проверками и подобными конструкциями), но также может быть выполнено явно с помощью функции `Boolean(value)`.

Правило преобразования:

- Значения, которые интуитивно «пустые», вроде `0`, пустой строки, `null`, `undefined` и `NaN`, становятся `false`.

- Все остальные значения становятся `true` .

Например:

```
1 alert( Boolean(1) ); // true
2 alert( Boolean(0) ); // false
3
4 alert( Boolean("Привет!") ); // true
5 alert( Boolean("") ); // false
```

### ⚠ Заметим, что строка с нулём `"0"` — это `true`

Некоторые языки (к примеру, PHP) воспринимают строку `"0"` как `false` . Но в JavaScript, если строка не пустая, то она всегда `true` .

```
1 alert( Boolean("0") ); // true
2 alert( Boolean(" ") ); // пробел это тоже true (любая непустая строка это true)
```

## Итого

Существует 3 наиболее широко используемых преобразования: строковое, численное и логическое.

**Строковое** – Происходит, когда нам нужно что-то вывести. Может быть вызвано с помощью `String(value)` . Для примитивных значений работает очевидным образом.

**Численное** – Происходит в математических операциях. Может быть вызвано с помощью `Number(value)` .

Преобразование подчиняется правилам:

Значение	Становится...
undefined	NaN
null	0
true / false	1 / 0
string	Пробельные символы по краям обрезаются. Далее, если остаётся пустая строка, то получаем <code>0</code> , иначе из непустой строки «считывается» число. При ошибке результат <code>NaN</code> .

**Логическое** – Происходит в логических операциях. Может быть вызвано с помощью `Boolean(value)` .

Подчиняется правилам:

Значение	Становится...
0 , null , undefined , NaN , ""	false
любое другое значение	true

Большую часть из этих правил легко понять и запомнить. Особые случаи, в которых часто допускаются ошибки:

- `undefined` при численном преобразовании становится `NaN` , не `0` .
- `"0"` и строки из одних пробелов типа `" "` при логическом преобразовании всегда `true` .

В этой главе мы не говорили об объектах. Мы вернёмся к ним позже, в главе [Преобразование объектов в примитивы](#), посвящённой только объектам, сразу после того, как узнаем больше про основы JavaScript.



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.