**ПРЕОБРАЗОВАНИЕ ТИПОВ**

В JavaScript нет строгой типизации, это язык с динамической типизацией данных, что приводит к тому, что в определенных случаях, данные одного типа могут быть неявно преобразованы в другой тип автоматически.  
Есть три типа преобразований:

* • строковое преобразование;
* • численное преобразование;
* • логическое преобразование.

Преобразования происходят не только автоматически, их можно производить и вручную.

Строковое преобразование

Строковое преобразование вручную.  
Функция String().

String(15); // "15"

String(true); // "true"

String(null); // "null"

String(undefined); // "undefined"

String([]); // "" (пустая строка)

String([1, 2, 3]); // "1, 2, 3"

String({}); // "[object Object]"

let typeCasting = function() { console.log('Test') }

String(typeCasting); // "function () { console.log('Test') }" (тело функции)

Метод toString().

(15).toString(); // "15"

(true).toString(); // "true"

([]).toString(); // "" (пустая строка)

([1, 2, 3]).toString(); // "1, 2, 3"

({}).toString(); // "[object Object]"

Автоматическое строковое преобразование.  
С помощью console, alert, prompt и innerHTML.

console.log(55); // "55"

alert(false); // "false"

prompt(); // любое значение, введенное в окно, будет преобразовано в строку

let status = true;

document.getElementById("container").innerHTML = status; // true

При помощи бинарного +.

"test" + 15; // "test15"

"test" + true; // "testtrue"

"test" + null; // "testnull"

"test" + undefined; // "testundefined"

"" + 15; // "15"

" " + true; // " true"

"" + null; // "null"

"" + undefined; // "undefined"

Численное преобразование

Численное преобразование вручную.  
Функция Number().

Number("5"); // 5

Number("a5"); // NaN

Number("5a"); // NaN

Number(true); // 1

Number(false); // 0

Number([]); // 0

Number(null); // 0

Number(undefined); // NaN

С помощью унарного +.  
Если хотя бы один операнд (элемент) выражения является строкой, то и второй будет преобразован к строке, происходит конкатенация – «склеивание строк», причем сначала число приводится к строке и лишь потом происходит конкатенация.

5 + +"5"; // 10

+"test"; // NaN

+"true"; // NaN

+true; // 1

+false; // 0

+null; // 0

+undefined; // NaN

При помощи операторов инкремент (++) или декремент (—).

var first = '1';

++first;

console.log(typeof first); // number

var second = '2';

second--;

console.log(typeof second); // number

var third = 'three';

++third;

console.log(typeof third); // NaN

Преобразование при помощи функции parseInt() или parseFloat().  
Функция parseInt() используется для преобразования строки в целое число, а функция parseFloat() для преобразования в дробное.

parseInt("test", 10); // NaN

parseInt(true, 10); // NaN

parseInt(null, 10); // NaN

parseInt(undefined, 10); // NaN

parseInt("12a34", 10); // 12

parseInt("1234", 10); // 1234

parseInt("a1234", 10); // NaN

parseFloat("3.14"); // 3.14

Автоматическое численное преобразование.

При выполнении математических операций — , \* , / , % (кроме +).

"4" - "2"; // 2

"6" % 4; // 2

"2" \* 5; // 10

true \* 5; // 5

false \* 5; // 0 (false преобразуется к 0)

20 / "10"; // 2

При выполнении операций сравнения >, >=, <, <= , ==, !=, (кроме ===, !==).

true > 0 // true

true == 1 // true

undefined > 0 // false

undefined < 0 // false

undefined == 0 // false

undefined <= 0 // false

undefined >= 0 // false

"2" == 2 // true

"2a" == 2 // false

"a2" == 2 // false

Null стоит рассмотреть отдельно. При сравнении больше/меньше кажется, что null равен нулю. Но прямое сравнение показывает, что null нулю НЕ равен. Этот случай является исключением, и его надо учитывать при сравнении различных величин с null.

null < 2 // true

null > 2 // false

null < 0 //false

null > 0 // false

null == 0 // false

null <= 0 //true

null >= 0 //true

Логическое преобразование

Ручное логическое преобразование.  
Функция Boolean().

Boolean("text"); // true

Boolean("false"); // true

Boolean("0"); // true

Boolean("1"); // true

Boolean(""); // false

Boolean(1); // true

Boolean(0); // false

Boolean(-5); // true

Boolean(5); // true

Boolean(null); // false

Boolean(undefined); // false

При помощи оператора !!

!!"test" // true

!!"false" // true

!!"0" // true

!!"" // false

!!0 // false

!!1 // true

!!-15 // true

!!null // false

!!undefined // false

Автоматическое логическое преобразование.  
Все выражения, находящиеся внутри оператора if или тернарного оператора, всегда приводятся к логическому типу.

if ("") {

console.log("True");

} else {

console.log("False");

} // False

if ("test") {

console.log("True");

} else {

console.log("False");

} // True

let status = "Done";

let result = (status) ? "True" : "False";

console.log(result); // True

Преобразование объектов

При логическом преобразовании объект, даже пустой, всегда преобразовывается в true.  
У каждого объекта всегда есть оба метода (toString() и valueOf()). Строковым представлением объекта является «[object Object]».

При преобразовании объекта в строку интерпретатор JavaScript выполняет следующие действия:

• Если объект имеет метод toString(), интерпретатор вызывает его. Если он возвращает простое значение, интерпретатор преобразует значение в строку (если оно не является строкой) и возвращает результат преобразования.  
• Если объект не имеет метода toString() или этот метод не возвращает простое значение, то интерпретатор проверяет наличие метода valueOf(). Если этот метод определён, интерпретатор вызывает его. Если он возвращает простое значение, интерпретатор преобразует это значение в строку (если оно не является строкой) и возвращает результат преобразования.  
• В противном случае интерпретатор делает вывод, что ни toString() ни valueOf() не позволяют получить простое значение и возбуждает ошибку TypeError.

При преобразовании объекта в число интерпретатор выполняет те же действия, но первым пытается применить метод valueOf():

• Если объект имеет метод valueOf(), возвращающий простое значение, интерпретатор преобразует (при необходимости) это значение в число и возвращает результат.  
• Если объект не имеет метода valueOf() или этот метод не возвращает простое значение, то интерпретатор проверяет наличие метода toString(). Если объект имеет метод toString(), возвращающий простое значение, интерпретатор выполняет преобразование и возвращает полученное значение.  
• В противном случае интерпретатор делает вывод, что ни toString() ни valueOf() не позволяют получить простое значение и возбуждает ошибку TypeError.

"" + new Date(); // Sun Sep 10 2017 20:57:14 GMT+0300 (Eastern Europe Daylight Time)

+new Date() // 1505066265350

Таким образом, в каждом объекте можно переопределить стандартный метод toString() и valueOf().

var customObject = {};

Number(customObject); // NaN

String(customObject); // "[object Object]"

customObject = {

toString: function() {

return "It's an Object";

},

valueOf: function() {

return 777;

}

}

Number(customObject); // 777

String(customObject); // "It's an Object"

У массивов так же есть свои методы для преобразования к примитивам:

var array = [];

Number(array); // 0

String(array); // ""

array = [1, 2, 3];

Number(array); // NaN

String(array); // "1, 2, 3"

[ПРЕДЫДУЩАЯ ЧАСТЬ](https://www.jsexpert.net/course/clear-javascript-beginner/?course_type=content&course_page=7&lecture=1&section-quiz)[СЛЕДУЮЩАЯ ЧАСТЬ](https://www.jsexpert.net/course/clear-javascript-beginner/?course_type=content&course_page=7&lecture=3&section-quiz=1)