**Type Conversions**

Most of the time, operators and functions automatically convert the values given to them to the right type.

For example, alert automatically converts any value to a string to show it. Mathematical operations convert values to numbers.

There are also cases when we need to explicitly convert a value to the expected type.

**Not talking about objects yet**

In this chapter, we won’t cover objects. For now, we’ll just be talking about primitives.

Later, after we learn about objects, in the chapter [Object to primitive conversion](https://javascript.info/object-toprimitive) we’ll see how objects fit in.

**[String Conversion](https://javascript.info/type-conversions" \l "string-conversion)**

String conversion happens when we need the string form of a value.

For example, alert(value) does it to show the value.

We can also call the String(value) function to convert a value to a string:

let value = true;

alert(typeof value); // boolean

value = String(value); // now value is a string "true"

alert(typeof value); // string

String conversion is mostly obvious. A false becomes "false", null becomes "null", etc.

**[Numeric Conversion](https://javascript.info/type-conversions" \l "numeric-conversion)**

Numeric conversion in mathematical functions and expressions happens automatically.

For example, when division / is applied to non-numbers:

alert( "6" / "2" ); // 3, strings are converted to numbers

We can use the Number(value) function to explicitly convert a value to a number:

let str = "123";

alert(typeof str); // string

let num = Number(str); // becomes a number 123

alert(typeof num); // number

Explicit conversion is usually required when we read a value from a string-based source like a text form but expect a number to be entered.

If the string is not a valid number, the result of such a conversion is NaN. For instance:

let age = Number("an arbitrary string instead of a number");

alert(age); // NaN, conversion failed

Numeric conversion rules:

| **Value** | **Becomes…** |
| --- | --- |
| undefined | NaN |
| Null | 0 |
| true and false | 1 and 0 |
| string | Whitespaces (includes spaces, tabs \t, newlines \n etc.) from the start and end are removed. If the remaining string is empty, the result is 0. Otherwise, the number is “read” from the string. An error gives NaN. |

Examples:

alert( Number(" 123 ") ); // 123

alert( Number("123z") ); // NaN (error reading a number at "z")

alert( Number(true) ); // 1

alert( Number(false) ); // 0

Please note that null and undefined behave differently here: null becomes zero while undefined becomes NaN.

Most mathematical operators also perform such conversion, we’ll see that in the next chapter.

**[Boolean Conversion](https://javascript.info/type-conversions" \l "boolean-conversion)**

Boolean conversion is the simplest one.

It happens in logical operations (later we’ll meet condition tests and other similar things) but can also be performed explicitly with a call to Boolean(value).

The conversion rule:

* Values that are intuitively “empty”, like 0, an empty string, null, undefined, and NaN, become false.
* Other values become true.

For instance:

alert( Boolean(1) ); // true

alert( Boolean(0) ); // false

alert( Boolean("hello") ); // true

alert( Boolean("") ); // false

**Please note: the string with zero "0" is true**

Some languages (namely PHP) treat "0" as false. But in JavaScript, a non-empty string is always true.

alert( Boolean("0") ); // true

alert( Boolean(" ") ); // spaces, also true (any non-empty string is true)

**[Summary](https://javascript.info/type-conversions" \l "summary)**

The three most widely used type conversions are to string, to number, and to boolean.

**String Conversion** – Occurs when we output something. Can be performed with String(value). The conversion to string is usually obvious for primitive values.

**Numeric Conversion** – Occurs in math operations. Can be performed with Number(value).

The conversion follows the rules:

| **Value** | **Becomes…** |
| --- | --- |
| undefined | NaN |
| null | 0 |
| true / false | 1 / 0 |
| string | The string is read “as is”, whitespaces (includes spaces, tabs \t, newlines \n etc.) from both sides are ignored. An empty string becomes 0. An error gives NaN. |

**Boolean Conversion** – Occurs in logical operations. Can be performed with Boolean(value).

Follows the rules:

| **Value** | **Becomes…** |
| --- | --- |
| 0, null, undefined, NaN, "" | false |
| any other value | true |

Most of these rules are easy to understand and memorize. The notable exceptions where people usually make mistakes are:

* undefined is NaN as a number, not 0.
* "0" and space-only strings like " " are true as a boolean.

Objects aren’t covered here. We’ll return to them later in the chapter [Object to primitive conversion](https://javascript.info/object-toprimitive) that is devoted exclusively to objects after we learn more basic things about JavaScript.

**[Comments](https://javascript.info/type-conversions" \l "comments)**

read this before commenting…