Converting to the Number Type

In this lesson we shall see how different JavaScript types can be converted to the number type.
Let's begin!

WE'LL COVER THE FOLLOWING

JavaScript provides **three functions** to convert non-numeric values into numbers:

1. the `Number()` casting function

2. the `parseInt()` function
3. the `parseFloat()` function

While Number() is used to convert any data to numbers, parseInt() and parseFloat() are used specifically to convert strings to numbers.

The first function that can be used to convert non-numeric values into numbers is the `Number()` casting function.

The `Number()` function

Rules of the `Number()` Function

The `Number()` function uses these rules:

♦ An undefined value returns `NaN`.

♦ null is converted to 0.

♦ When the argument is a number, it is simply passed through and returned.

♦ Boolean values, true and false, are converted to 1 and 0, respectively.

Number conversion and the case of the `String`

String values are converted to numbers according to the following cases:

> 1. An empty string is converted to 0.
> 2. If the string contains only numbers, optionally preceded by a plus or minus sign, it is always converted to a decimal number. Leading zeros are ignored.
> 3. If the string contains a valid floating-point format, it is converted into the appropriate floating-point numeric value. Leading zeros are ignored.
> 4. If the string contains a valid hexadecimal format (with a “0x” prefix followed by one or more hexadecimal digits), it is converted into an

integer that matches the hexadecimal value.

5. If the string contains anything other than these previous formats, it is converted into NaN.

You can apply `Number()` to objects. In this case, the `valueOf()` method of the object instance is called and the returned value is converted based on the previously described rules. If that conversion results in NaN, the `toString()` method is called and the rules for converting strings are applied.

Examples:

Here are a few examples:

index.js

```
var n1 = Number(null);      // 0
var n2 = Number(undefined); // NaN
var n3 = Number(true);      // 1
var n4 = Number(26.78);     // 26.78
var n5 = Number("014");     // 14
var n6 = Number("0x3e");    // 62
var n7 = Number("");        // 0
var n8 = Number("qwe");     // NaN
var strObj = new String("34.5");
var n9 = Number(strObj);    // 34.5
```

The second method that can be used to convert non-numeric values into numbers is the `parseInt()` Function.

The `parseInt()` function

If you work with integer numbers, parseInt() is a more flexible solution than Number().

When examining a string value, `parseInt()` skips leading whitespaces and checks the first useful (non-whitespace) character. If this character isn't a number, the minus sign, or the plus sign, `parseInt()` returns `NaN`. This behavior implies that the empty string returns `NaN`. If the first character is a number, plus, or minus, then the conversion goes on to the second character and continues on until either the end of the string is reached, or a non-numeric character is found.

Provided that the first useful character is a number, the `parseInt()` function also recognizes all hexadecimal numbers as they begin with “`0x`”.

Examples:

Let’s have a look at these examples:

index.js

```
var n1 = parseInt("");        // NaN
var n2 = parseInt("  -12.4"); // -12
var n3 = parseInt("238bla");  // 238
var n4 = parseInt("047");     // 47
var n5 = parseInt("0x41");    // 65
var n6 = parseInt("42");      // 42
```

> 📃**NOTE:** While ECMAScript 3 recognized integers starting with “0” as octal numbers, ECMAScript 5 interpreted them as decimal numbers.

To be more accurate, `parseInt()` provides a second optional argument, the **radix** where the base of conversion must be between two and 36 to use.

It makes the conversion more explicit:

index.js

```
var n1 = parseInt("1001001", 2); // 73
var n2 = parseInt("124", 5);     // 39
var n3 = parseInt("238", 10);    // 238
var n4 = parseInt("41", 16);     // 65
var n5 = parseInt("54", 8);      // 44
```

Even if you convert strings holding decimal numbers, use the radix argument (10) explicitly:

index.js

```
var number = parseInt("281", 10);
```

The third method that can be used to convert non-numeric values into numbers is the `parseFloat()` function.

The `parseFloat()` function

The `parseFloat()` function provides similar flexibility as `parseInt()` by means of skipping leading whitespaces and stopping at the first non-float-number character.

Evidently, parseFloat() does not support the hexadecimal format or the radix argument.

When you provide a string representing an integer, then an integer number will be returned.

Examples:

Here are a few examples:

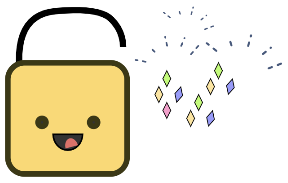
index.js

```
var n1 = parseFloat("12.34e4bla"); // 123400
var n2 = parseFloat("   -0012");    // -12
var n3 = parseFloat("0x4f");        // 0
var n4 = parseFloat("3.14");        // 3.14
```

Achievement unlocked!

Congratulations! You've understood how to deal with number type conversion in JavaScript.

Great work! Give yourself a round of applause! :)



In the *next lesson*, we'll study some number methods!