imarketinx.de

Webdesign // individuell • flexibel • robust

Rüdiger Alte ● Trögelsbyer Weg 38E ● 24943 Flensburg ● Tel.: 0461 / 67 999 667 ● Mobil: 0160 / 43 82 369 ● Mail: office@imarketinx.de ● Web: imarketinx.de

Automatically Number Headings, Figures and Tables in an HTML Text

Structure Texts with the CSS Function "`counter()`"

Even if short cracky texts with many pictures, videos and animations dominate today on internet pages, there are still occasions for longer texts on websites: If, for example, it is about the core of a product, the essence of a service, the unique selling proposition of a company or – yes, there is also – about scientific texts, a text on the Internet may be a bit more extensive than generally suggested – if the quality is right.

Well thought-out structuring

Longer texts should be well structured by

- headings
- numbering the headings (e. g. 1. Chapter, 1.1. Chapter, 1.2. Chapter etc.)
- typographic features like lists or text boxes
- figures and tables

In a previous article I have already described how texts should be structured on the Web.

Wouldn't it be nice if the numbering of headings, figures and tables would automatically adjust when new headings or figures are inserted ?

Exactly that can be done with a little HTML and CSS.

To do this I use the CSS function `counter()` and the pseudo element `::before`. Rather it doesn't need – except for a little code around it of course 😉. The CSS function `counter()` is supported by all modern browsers.

Complex Numbering of an HTML Text

In the following I will create a template for numbering a text with headings, figures and tables, which can be used for any text. First I would like to take a closer look at the CSS function `counter()`.

The CSS Function "`counter()`" under Magnifying Glass

A counter alone has no effect on the HTML code of a website. Only with the help of the counter function can the values of a counter be output. Let's have a closer look at this:

The CSS function `counter()` is mainly used in the pseudo elements `::after` and `::before`. In order to initialize the counter function, a name for the counter must first be defined. This is always done in a comprehensive HTML element; this can be the body element, the main element or a div container.

For example, if we have a numbered list with the HTML elements `ol` and `li` in front of us, the name of the function is defined in the CSS selector `ol`:

```
ol { counter-reset: count; }
```

The property *counter-reset* has the value *count*. This value is the name of the counter, in our example it is *count*; the name can be assigned arbitrarily.
counter-reset also causes the numbering to be set to zero.

Hint
For lists, remove the default `list-style-type` either by
`ol { list-style-type: none; }` or through
`li { display: block; }`.
Now you can freely edit the numbering of the list elements.

So that numbering can now be initiated, the list element `li` is addressed in the CSS code and the counter value is incremented, i.e. increased step by step:

```
li { counter-increment: count; }
```

That means that the counter with the name *count* is now incremented by 1 in the list elements, i.e. from 0, which was determined by the counter reset, to 1 and then to 2 and so on.

If you want to increase in steps of two instead, the code is:

`li { counter-increment: count 2; }` (Negative values are also allowed.)

Now you can design and enhance the numbering. In this example this is done in the pseudo element `::before`, in which the CSS function `counter()` is inserted in the property `content` (and only here this function will work in all browsers, see here).

The CSS function `counter()` is structured as follows:

```
li::before { content: counter(count); }
```

The structure of the CSS statement in general terms:
Selector (= li::before) { Property (= content): Value (= counter(counter-name); }

To change the numbering – e. g. in capitals –, you can use the CSS function in that way:

```
li::before { content: counter(count, upper-latin); }
```

It is also possible to insert words, signs, pictures and symbols:

```
li::before { content: "Chapter " counter(count, upper-latin) " \
2606"; }
```

The word *Chapter* is inserted before the Counter (A, B, C ...) and after it the star symbol:

Chapter A ☆

In the next step I use the counter for headings, figures and tables.

Numbering with the CSS Counter Function and the Pseudo Element ::before

The Requirements

Let us imagine that we want to create a text with six levels of headings, which should also be nested, and with figures and tables. The numbering should be assigned automatically on all levels according to the scheme:

A. 1. 2. 3. 4. 5. in the headings, where *A.* is the first level heading and *5.* is the sixth level heading; the figures and tables should be numbered consecutively.

Further options for numbering you will find here.

Die Numbering of the Headings

Hint
I recommend in practice the use of CSS classes instead of element names, so instead of `h1` the class `.heading-1` and so on. For a better overview I remain here with *h1, h2* ...

First of all I start the incrementing for each heading with the following code:

```
h1 { counter-increment: h1; } ... h6 { counter-increment: h6; }
```

Then I use the following code to set all the heading levels to zero so that the numbering can start at 1 (i.e. if a level is not present, a zero is displayed):

In the body as a comprehensive element the counter is initiated:

```
body { counter-reset: h1 h2 h3 h4 h5 h6; }
```

And is then continued for each level of headings, so that the numbering for each level always begins anew:

```
h1 { counter-reset: h2 h3 h4 h5 h6; }
h2 { counter-reset: h3 h4 h5 h6; }
```

up to

```
h5 { counter-reset: h6; }
```

It is important that only one space but no comma is placed between the values `h1` to `h6`.

For the highest level `h1` I have got now that code:

```
h1 { counter-increment: h1; counter-reset: h2 h3 h4 h5 h6; }
```

To make the numbering visible, I add the CSS counterfunction `counter()` just before the heading:

```
h1::before { content: counter(h1,upper-alpha) ". "; }
```

The result for the first heading `h1` in the text is *A.* and for the second heading of the level `h1` *B.* .

For each level the counter of the superior level(s) is inserted and additionally the counter of this level:

```
h2::before { content: counter(h1,upper-alpha) ". "
counter(h2,decimal) ". "; }
```

The result is: *A. 1.*

The lowest level `h6` accordingly has the following code:

```
h6::before { content: counter(h1,upper-alpha) ". "
counter(h2,decimal) ". " counter(h3,decimal) ". "
counter(h4,decimal) ". " counter(h5,decimal) ". "
counter(h6,decimal) ". "; }
```

And that's it with the headings!

Numbering of Figures and Tables

Analogous to the procedure with the headings, I initiate the counter in a comprehensive element. Since the body element is already reserved for the headings, the element `main` is suitable for the tables. For the figures I create a container with the class `.wrapper`.

The numbering should appear in the captions of the figures and tables. For this I need the following code:

Tables

For the table description I use the class `.table-subline` in a paragraph `p` to avoid colliding with the table elements `tfoot` or `caption`. I choose "table-counter" as name for the counter function:

```
main { counter-reset: table-counter; }
p.table-subline { counter-increment: table-counter; }
p.table-subline::before { content: "Tab. " counter(table-counter)
": "; }
```

The output now looks like this: *Tab. 1:* .

Figures

For the figures i use the HTML element `figcaption` and choose "fig-counter" as name for the counter function:

```
.wrapper { counter-reset: fig-counter; }
figcaption { counter-increment: fig-counter; }
figcaption::before { content: "Fig. " counter(fig-counter) ": "; }
```

The output now looks like this: *Fig. 1:* .

Headings, Figures and Tables Put Together

You can see the result on this demo site. You can also edit the template yourself. For this I have set the template to codepen.io for you.

Hint

In the next article I will develop a practical example for the use of the CSS counter function: **A template for a menu list!**

Links

You can find more informations and a lot of examples for the usage of the counter function on the websites of the World Wide Web Consortium (W3C):

here and here

and on the website of Mozilla Developer: here and here.

From Rachel Andrew there is a worth reading article on the subject in the Smashing Magazine.

For browser support of the counter function see caniuse.com.