CREATING WEB-PAGES
Using HTML5 and CSS3

STEP
IT ACADEMY

Lesson 3

Lists. CSS Margins and Padding

Contents

Lists. CSS Margins and Padding 4
- Lists 4
- Ordered (numbered) list 4
- Bulleted list. 11
- Description List 13
- Multi-level or nested lists 16

CSS styles for lists. 21
- Using the ::before pseudo element for list styles 28
- What are ::before and ::after pseudo elements? 28
- Style of multi-level lists. Nested selectors 31

CSS properties for indents and margins 40
- Margins and Padding. General Information 40
- Margins. 41

Padding 45
Default indent values 47
Home Task 52
Appearance of Tasks 53

› **There are references to files with examples and home task in the text. You can find them in the archive attached to the PDF of this lesson.**

Lists. CSS Margins and Padding

Lists

Multiple Internet pages use a list of any options for services or goods. Lists perfectly suits for such an enumeration. In HTML, there are 3 types: description, ordered (numbered) and unordered (bulleted) lists. Using lists, you can organize and arrange various types of data and present them in a user-friendly form.

Ordered (numbered) list

An ordered (numbered) list is a block with items, each of which by default has an Arabic number:

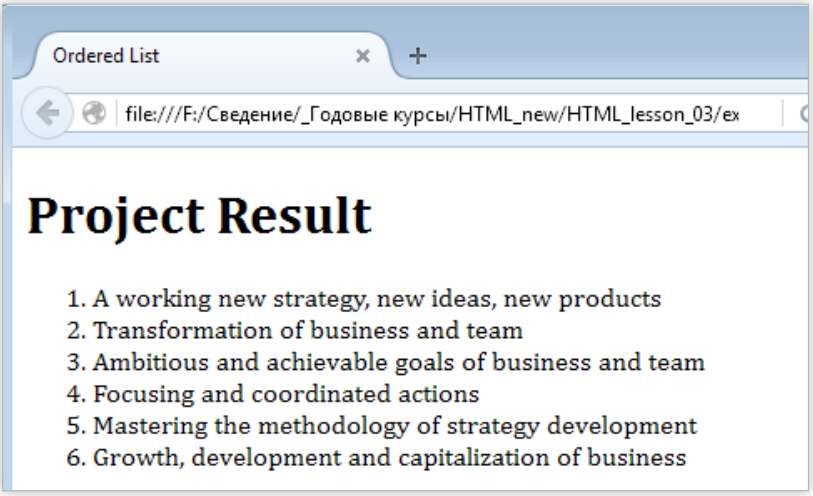


Figure 1

In html-markup, a numbered list is formed using the <ol> tag (*ordered list*) and the <li> tags (*list item*) nested in it.

```
<ol>
    <li> Effective new strategy, new ideas, new
         products</li>
    <li> Business and team transformation</li>
    <li> Ambitious and achievable business and team
         goals</ li>
    <li> Focusing and coordination of actions</li>
    <li> Mastering the methodology of strategy
         development</li>
    <li> Growth, development and capitalization of
         business</li>
</ol>
```

The sample file is *ordered_list.html*.

Instead of the Arabic numerals, we can indicate Roman numbers or letters of the Latin alphabet, both lowercase and uppercase, as the numbering, and also indicate the numbering on the basis of the Greek, Armenian or Georgian languages. For this, in HTML4 there was a type attribute, which is currently deprecated. In HTML5, it is customary to specify the type of the list using styles, in particular, using the list-style-type css property:

```
list-style-type: decimal (by default, you can even
    not specify) | decimal-leading-zero | armenian |
    georgian | lower-alpha | lower-greek |
    lower-latin | lower-roman | upper-alpha |
    upper-latin | upper-roman;
```

Visually, these styles will represent such a numbering (fig. 2, 3)

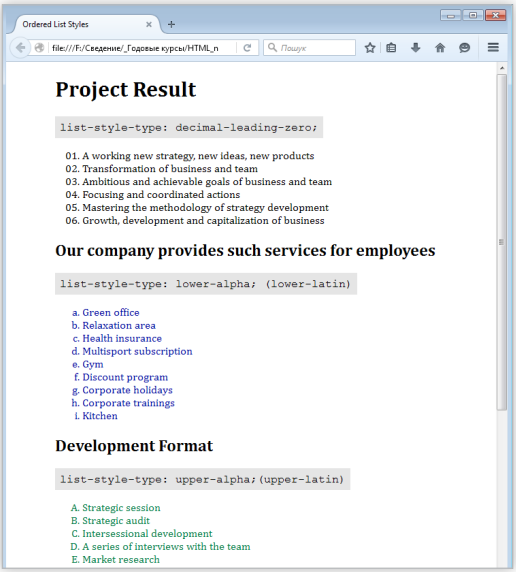


Figure 2

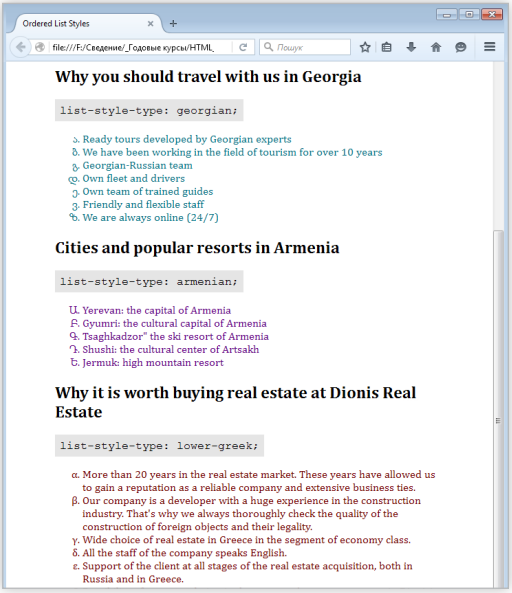


Figure 3

- Arabic numerals (1, 2, 3, ...) — decimal;
- Arabic numerals with leading zero for the figures of the first ten (01, 02, 03, ..., 09) — decimal-leading-zero;
- Armenian numbering — armenian;
- Georgian numbering — georgian;
- Uppercase Latin letters (A, B, C, ...) — upper-alpha or upper-latin;
- Lowercase Latin letters (a, b, c, ...) — lower-alpha or lower-latin;
- Uppercase Roman numerals (I, II, III, ...) — upper-roman;
- Lowercase Roman numerals (i, ii, iii, ...) — lower-roman;
- Lowercase Greek letters — lower-greek.

You can look at the appearance of lists with different styles in the file *ordered-list-style.html*.

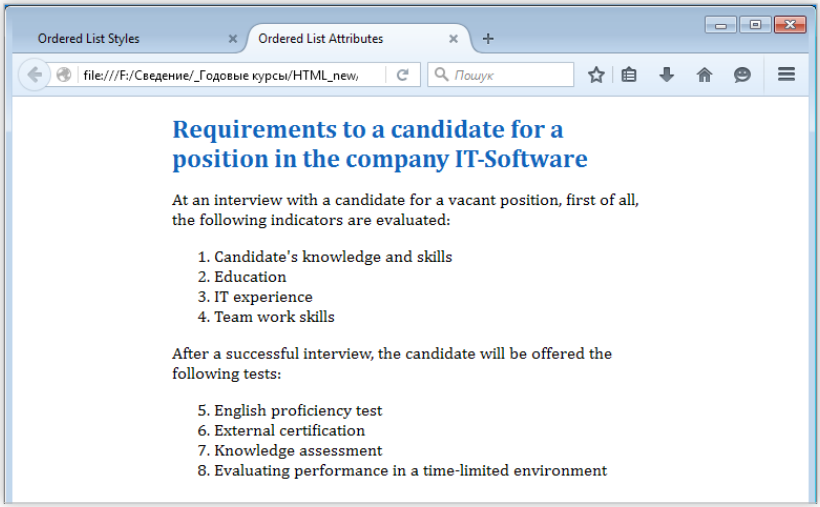


Figure 4

Numbered lists also have additional attributes: start for the <ol> tag and value for the <li> tag. In this case, the list does not start with the number 1, but with the one that is specified in the attribute start. As a rule, this attribute is necessary if the list needs to be split by any explanation or picture. For example (fig. 4).

```
<p> At an interview with a candidate for a vacant
    position, first of all, the following indicators
    are evaluated:</p>
     <ol>
         <li>Candidate's knowledge and skills</li>
         <li>Education</li>
         <li>IT experience</li>
         <li>Team work skills</li>
     </ol>

     <p>After a successful interview, the candidate
        will be offered the following tests:</p>
     <ol start="5">
         <li>English proficiency test</li>
         <li>External certification</li>
         <li>Knowledge assessment </li>
         <li>Evaluating performance in a time-limited
             environment</li>
     </ol>
```

It should be noted that it is used extremely rarely. The value attribute for the li element is also of rare use, and it changes the order of the numbering starting with the number that is specified as the value of the attribute. This attribute takes precedence over the start attribute if they are listed in the same list. In addition, these 2 attributes are applied to any numbering (in Latin letters, Roman numerals, etc.). In

the example, the li elements that have the value attribute are highlighted in red (fig. 5).

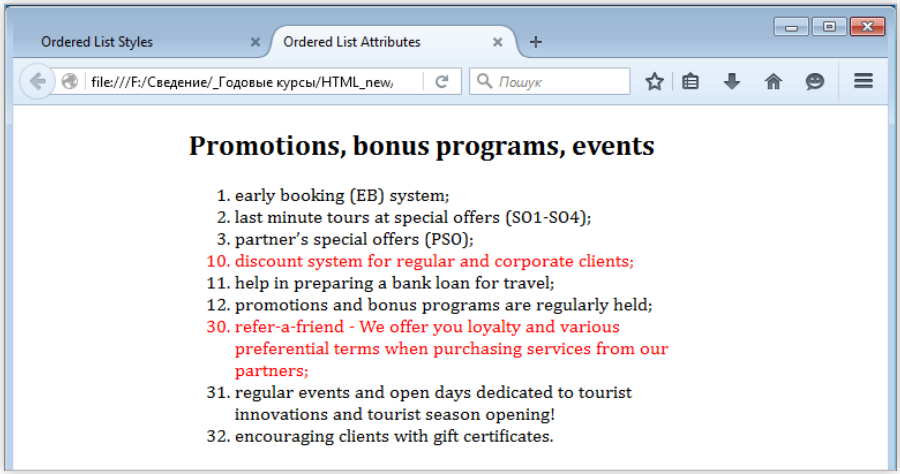


Figure 5

```
<ol>
    <li>Early booking (EB) system;</li>
    <li>last minute tours at special offers (SO1-SO4);
    </li>
    <li>partner's special offers(PSO);</li>
    <li value="10">discount system for regular and
        corporate clients;</li>
    <li>help in preparing a bank loan for travel;</li>
    <li>promotions and bonus programs are regularly
        held;</li>
    <li value="30">refer-a-friend - we offer loyalty…;
    </li>
    <li>regular events and open days dedicated to
        tourist innovations and tourist season
        opening! </li>
    <li>encouraging clients with gift certificates.</li>
</ol>
```

Another little used attribute for the <ol> tag is reversed, which changes the order of list numbering to the reverse one. Thus, instead of 1, 2, 3, 4 we will have 4, 3, 2, 1. For example, as follows:

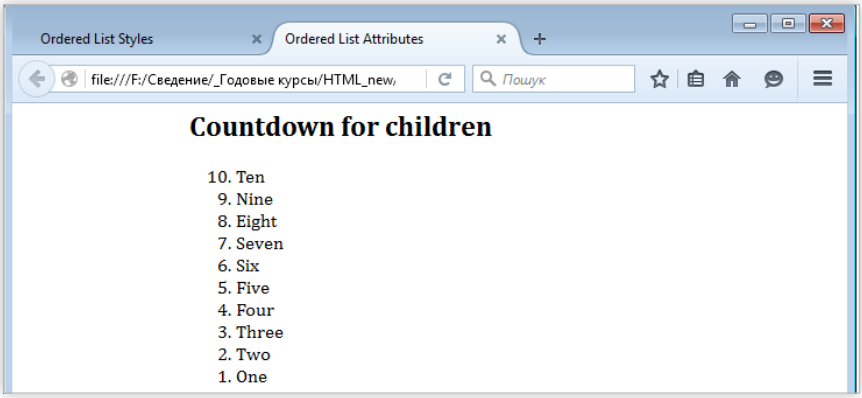


Figure 6

```
<ol reversed>
      <li>Ten</li>
      <li>Nine</li>
      <li>Eight</li>
      <li>Seven</li>
      <li>Six</li>
      <li>Five</li>
      <li>Four</li>
      <li>Three</li>
      <li>Two</li>
      <li>One</li>
  </ol>
```

- **Note:** *Internet Explorer does not support this attribute.*

You can see the sample code in the file *ordered-list-attributes.html*.

Bulleted list

The bulleted list differs from the numbered one in that it has markers instead of digits (by default, in the form of bullets). Also the bulleted list is called an unordered list, because numbering is not needed.

A bulleted list is formed similarly to the numbered one using two tags: <ul> (*unordered list*) and the already known <li> (*list item*).

```
<ul>
      <li>Biographies (255)</li>
      <li>Action (1581)</li>
      <li>Westerns (76)</li>
      <li>War (237)</li>
      <li>Detective (627)</li>
      <li>Documental (64)</li>
      <li>Dramas (2814)</li>
      <li>Historical (237)</li>
      <li>Comedy (1983)</li>
</ul>
```

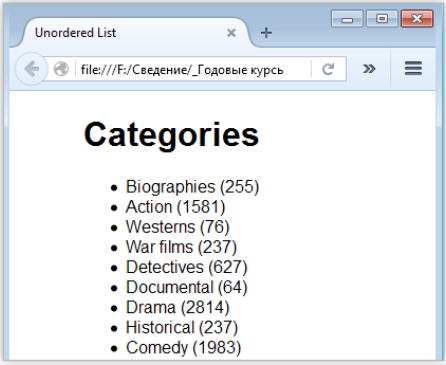


Figure 7

The sample file is *unordered_list.html*.

The marker type was previously defined using the deprecated attribute type, which is currently replaced with the css-property list-style-type:

```
list-style-type: disc | circle | square | none
```

- **disc** — a filled circle (the default value, you can even not specify);
- **circle** — an empty circle;
- **square** — a dark square;
- **none** — no markers (numbering).

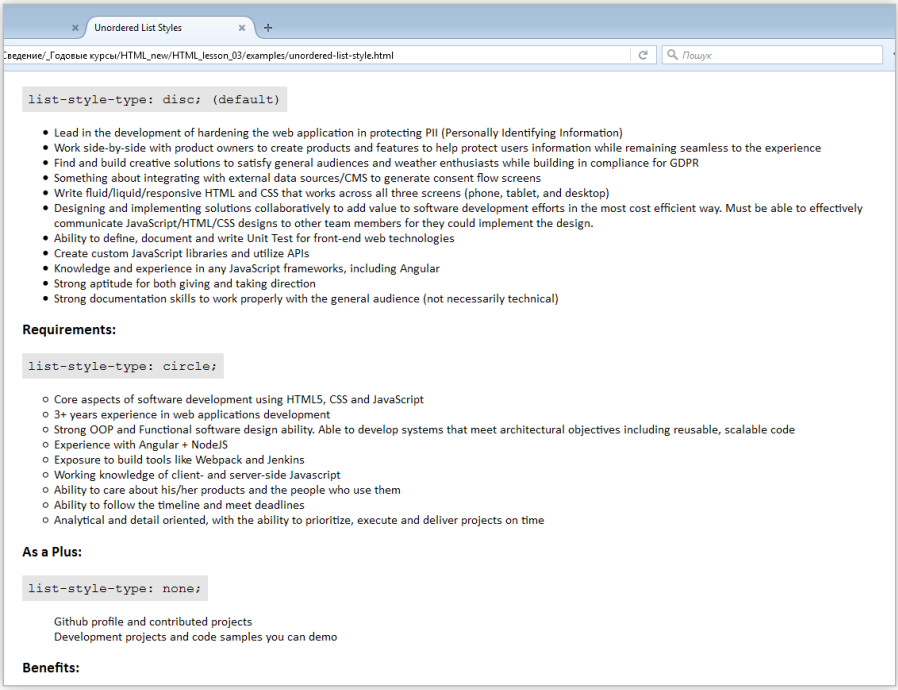


Figure 8

The sample file is *unordered-list-style.html*.

Description List

Description lists are needed in those sites that are devoted, as a rule, to learning. For example, an excellent website webref.ru uses description lists to specify tag attributes or property options.

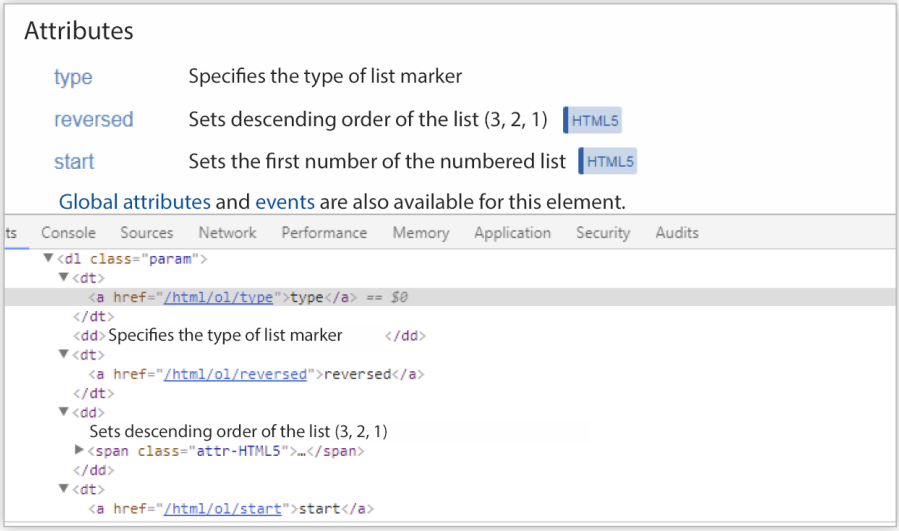


Figure 9

Any description list is created using 3 tags:

- dl — description list — primary or parent tag;
- dt — definition term — nested tag;
- dd — definition description — nested tag.

List code is as follows:

```
<dl>
  <dt>Term 1</dt>
  <dd>Description 1</dd>
  <dt>Term 2</dt>
```

```
  <dd>Description 2</dd>
  <dt>Term 3</dt>
  <dd>Description 3</dd>
</dl>
```

As an example, we use definitions of several terms:

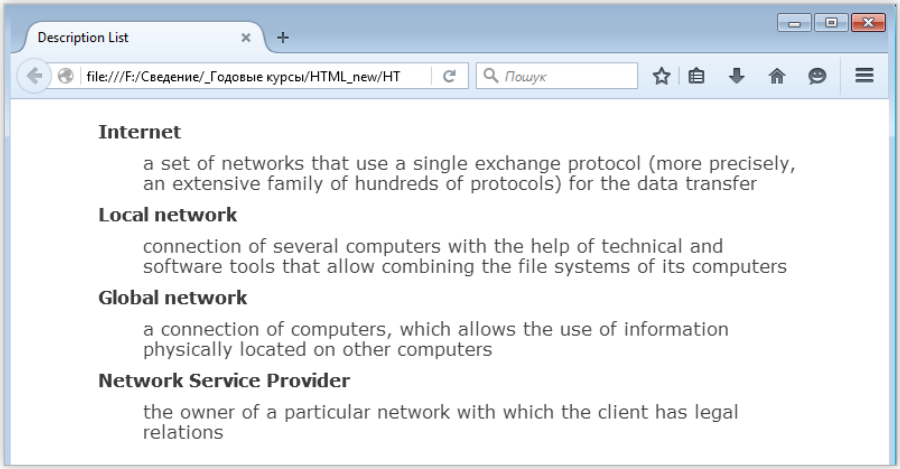


Figure 10

The screenshot shows that, in comparison with the term in the dt element, the dd element has a margin of 40px to the right from the left edge of the browser window. This margin is determined by the margin-left property or in the Chrome browser — webkit-margin-start. Due to this, the description visually differs from the term and attracts attention.

```
dd {                                    user agent stylesheet
   display: block;
   -webkit-margin-start: 40px;
}
```

Figure 11

The description list is well formatted with css. For a website that uses many variants of such lists, it makes sense to assign css properties to the tag selector, without resorting to the use of class selectors or id selectors. For example, the list in the screenshot above can be formatted like this:

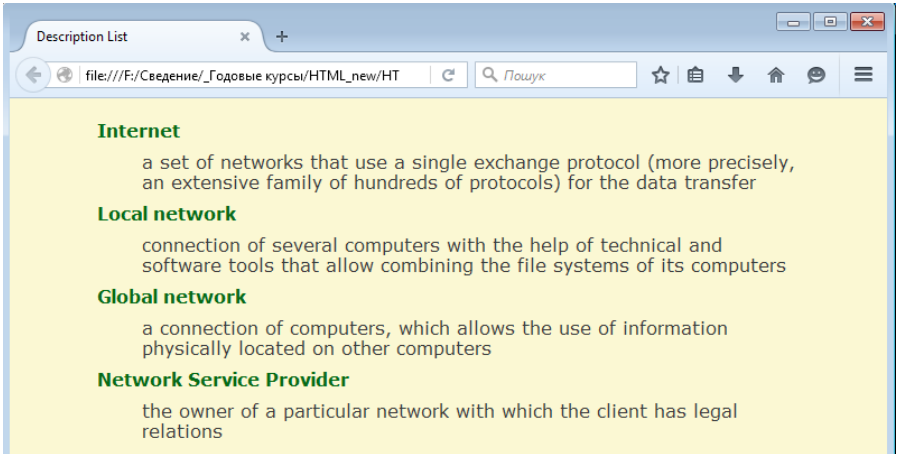


Figure 12

Code of css-styles:

```
<style>
  dt {
       font-weight: bold;
       color: #036011;
  }
  dd {
       color: #333;
       margin-top: 10px;
       margin-bottom: 10px;
  }
</style>
```

You can see this example in the file *description-list.html*

Related Links:

- https://webref.ru/layout/html5-css3/list
- https://html5book.ru/html-lists/
- http://htmlbook.ru/faq/theme/list
- http://html-plus.in.ua/spiski-v-html/

Multi-level or nested lists

A multi-level list is a list, which contain a second-level list within the <li> elements. Thus, the second list is put inside the first, therefore such lists are called nested (*nested_list.html* file).

In a multilevel list, you can use both ordered and unordered lists, as well as description lists, alternating them in any sequence. It is important that the nested list is in the <li> element of the parent list. The number of levels in the lists can be anything, although more than 3 nesting levels are extremely rare. Correct code example:

```
<ol>
     <li>Squares
          <ul>
               <li>Palace Square</li>
               <li>Market Square</li>
          </ul>
     </li>
     <li>Castles
          <ul>
               <li>Royal Castle</li>
               <li>Wilanów Palace</li>
          </ul>
```

```
    </li>
    <li>Stare Miasto</li>
    <li>Museums
        <ul>
            <li>Historical Museum</li>
            <li>Polish Army Museum</li>
            <li>Cartoon Museum</li>
            <li>Marie Curie Museum</li>
            <li>Chopin Museum</li>
            <li>National Museum</li>
        </ul>
    </li>
    <li>Krakowskie Przedmieście</li>
</ol>
```

Invalid code example (*invalid-multilevel-list.html* file)

```
<ol>
    <li>Squares</li>
    <ul>
        <li>Palace Square</li>
        <li>Market Square</li>
    </ul>
    <li>Castles</li>
    <ul>
        <li>Royal Castle</li>
        <li>Wilanów Palace</li>
    </ul>
    <li>Stare Miasto</li>
    <li>Museums</li>
    <ul>
        <li>Historical Museum</li>
```

```
            <li>Polish Army Museum</li>
            <li>Cartoon Museum</li>
            <li>Marie Curie Museum</li>
            <li>Chopin Museum</li>
            <li>National Museum</li>
        </ul>
        <li>Krakowskie Przedmieście</li>

</ol>
```

The mistake is in that the nested list is inserted not inside the <li> element, but between two adjacent elements, and this is inadmissible from the point of view of the validator:

Figure 13

Visually the list looks as follows (fig. 14).

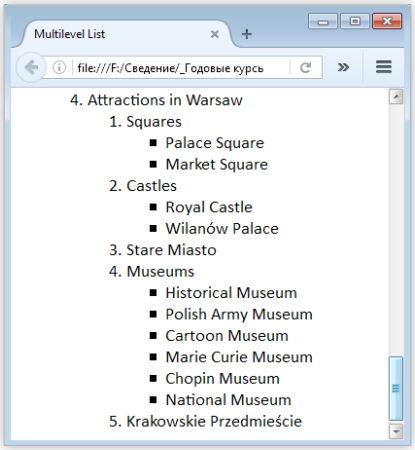


Figure 14

This list can be supplemented and it will be much larger:

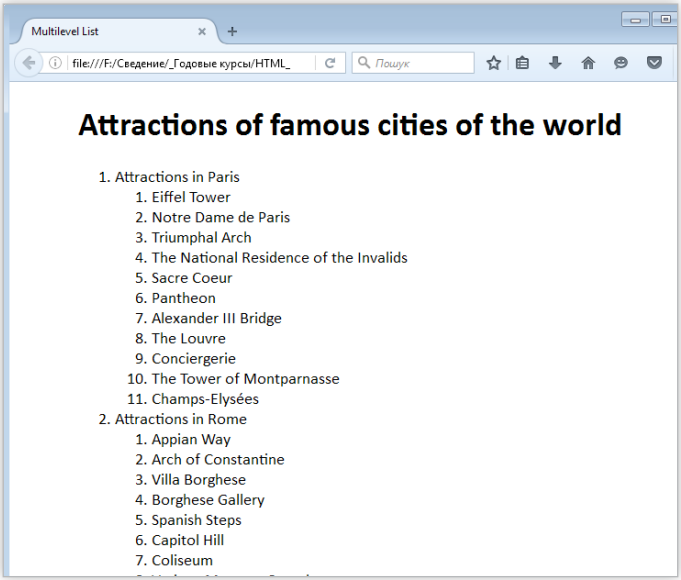


Figure 15

The sample file is *multilevel-list.html*.

Note that each nested list is shifted relative to its parent to the right, and when bulleted lists are nested in each other, the marker style changes: disk for the main list, circle for the first child list, and square for the second child list.

Related links:

- https://webref.ru/layout/html5-css3/list/nested
- https://puzzleweb.ru/html/11_lists2.php
- http://html-plus.in.ua/vlozhennye-spiski/

CSS styles for lists

CSS-properties, which will be discussed below, are usually assigned for bulleted or numbered lists, and can be specified for both parent (ul, ol) and nested (li) elements.

We have already considered one of the style properties of the lists. This is a list-style-type property that specifies **the type of marker or number**:

```
list-style-type: decimal | decimal-leading-zero |
   armenian |  georgian | lower-alpha | lower-greek |
   lower-latin | lower-roman | upper-alpha |
   upper-latin | upper-roman | disc | circle |
   square | none;
```

In the previous examples, we separated it by values that were different for numbered and bulleted lists. You can assign any of the values for any kind of list, but the html-markup logic will be lost in this case.

Quite often markers or numbers in lists are "hidden" using the value list-style-type: none. For example, this is done when formatting the menu.

```
Emmet abbreviations: lst, lstdlz, lstlr, lstur, lstd,
lstc, lsts, lstn
```

The second property is the position of the numbers or markers. It is defined by the list-style-position property with the following values:

```
list-style-position: outside | inside | inherit
```

By default, all lists have list-style-position: outside, so you will rarely use this value. It implies that the markers (numbers) are placed to the left of the list item text. The value list-style-position: inside places markers (numbers) inside the list item text.

The value inherit implies that the nested list inherits this value from the parent element. Because nested list items usually have the same properties as parent ones, so they do not use this value very often.

```
Emmet abbreviations: lsp, lspi, lspo
```

See the example in the file *list-style-position.html*.

The third, quite popular property is list-style-image, which allows you to set an image as a list marker:

```
list-style-image: none | url(path to the file image.
extn);
list-style-image: url(markers/arrow.gif);
```

or

```
list-style-image: url(http://mysite.com/images/
arrow.gif);
```

In the first example, the path to the file is specified relative to the location of the html file and the folder with image for the markers (they must be in the shared folder). In the second — the path is indicated on the basis of the domain name of your website. Since at this stage you hardly have your website, the only option for classroom and home work will be using the first path.

As for the text editor Brackets, there are very convenient hints that allow you to easily specify the path to the image-marker file, as well as see the file appearance and its size by hovering over the *string of the path to the image*. An important condition for these hints is to open the folder as a project in Brackets, which is accessible from the explorer shortcut menu.

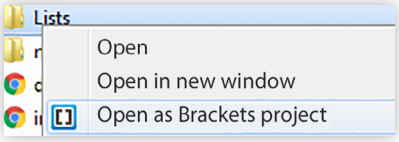


Figure 16

Once you have written the property list-style-image: url() and left the cursor in parentheses, Brackets starts prompting you, showing the folders that are in your project and the files in those folders:

Figure 17

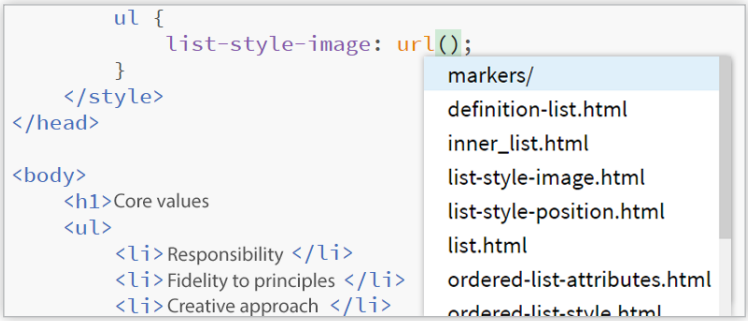


Figure 18

The list will also contain all the files that are in the current folder, so be careful. You can confirm your choice by pressing the Enter key or by clicking with the left mouse button.

Figure 19

When you hover over the text in parentheses, you will see how the image looks like and what its size is (in the screenshot — 25×25px):

Figure 20

In order to change the image to any other, you just put the cursor after the slash and press Ctrl + space:

Figure 21

To replace one image with another, you just need to select it and press Enter — and the file name will be automatically replaced, even if it was not selected.

When you select different files, you can use not only the mouse, but also the up and down arrow keys to move through the list.

As a result, we get the following list view:

The sample file is *list-style-image.html*

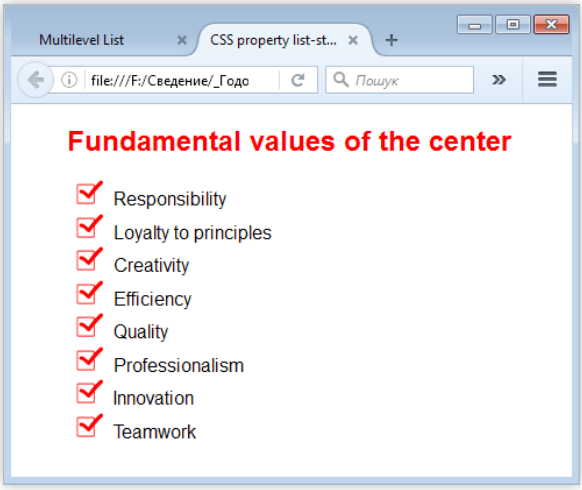


Figure 22

- **Note:** *the list-style-image property takes precedence over the list-style-type, i.e. the image will always overlap the marker or the number of the list item.*

In case you need to remove the assigned image marker in one of the list items, use the value none:

```
list-style-image: none;
Emmet abbreviation: lsi, lisi, lisin
```

The universal property list-style allows you to specify all the properties from the above separating them with a space, i.e. to specify both the style of the marker, and its position, and image. The enumeration goes in the following order:

```
list-style: list-style-type list-style-position
list-style-image;
```

For example:

```
list-style: upper-latin inside url(markers/somepic.png);
```

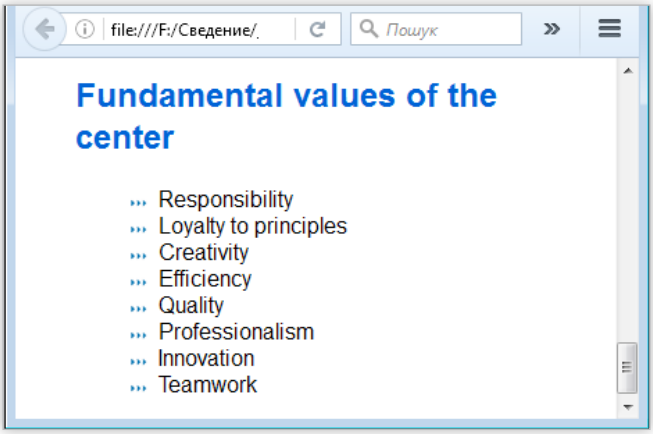


Figure 23

Since the image overlaps the marker, it makes sense to use only two properties (fig. 24):

```
list-style: upper-latin inside;
```

or

```
list-style: inside url(markers/somepic.png);
```

Figure 24

You can specify only one value in this css property:

```
list-style: url(markers/actor-icon.png);
```

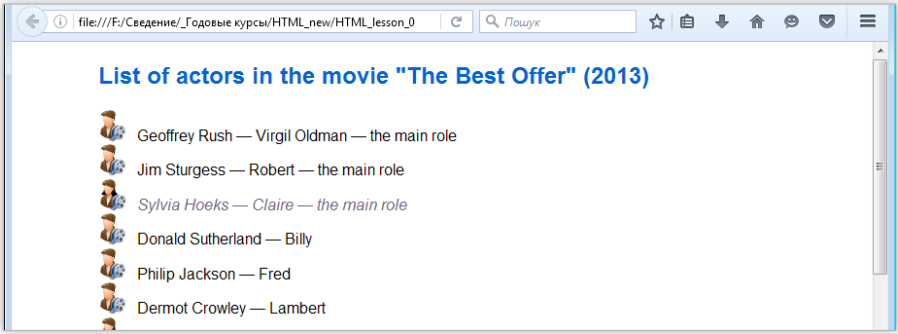


Figure 25

The default value is: disc outside none;
You can also specify an inherit value.

```
Emmet abbreviations: lis, lisn
```

The sample file is *list-style.html*.

Using the ::before pseudo element for list styles

Often one wants to change the numbering by specifying parentheses after the number in the ordered list, and a dash instead of a marker in the unordered list. To do this, disable the standard numbering or markers using the property list-style-type: none. And then use the ::before pseudo element for the <li> element.

What are ::before and ::after pseudo elements?

Pseudo elements have a word 'pseudo' before the word 'element' because initially they do not exist in the html-code. By the way, they will not be in the code at all, because they appear exclusively with the help of css. There are several pseudo elements, namely ::first-line, ::first-letter, ::selection, ::before and ::after. And they are allowed to use the syntax both with two colons before the name (CSS3), and with one (CSS2), because in the CSS3 version it was decided to separate the pseudo elements from the pseudo classes by adding a second colon to the pseudo elements. So do not be surprised if you see their different writing.

As for :before and :after, one can say that these pseudo elements allow you to add some content (text or visual content) before the text of the element or at the end of the element, i.e. after the contents of the element. And the text of the pseudo element will not be highlighted, and the same css-properties can be set both for it and for ordinary elements.

By default :before and :after are inline elements, and have the same font, text color, and font-style as the main element.

For example, the pseudo elements :before and :after for a paragraph that has italic style will also be in italic:

```
p {  font-style: italic;  }
p:before,  p:after {   background-color: #ccc; }
p:before {  content: 'Start'; }
p:after  {  content: "End";  }
```

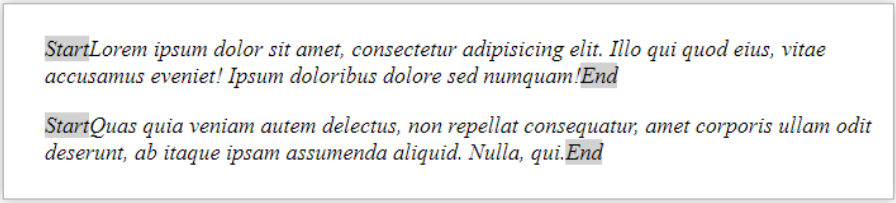


Figure 26

In the example (file *list-before.html*), the pseudo elements are highlighted with a gray background. In the code, in the content property, the text that will be displayed before and after the paragraph text is specified in quotes (double or single).

- **Note:** *the content property is mandatgory for the ::before and ::after pseudo elements. If this property is not specified, even with empty quotes («» or «'), then these pseudo elements will not be displayed on the page, whatever else properties you write for them.*

Related links:

- https://habrahabr.ru/post/154319/
- https://webref.ru/css/before
- https://webref.ru/css/after
- http://html-plus.in.ua/psevdo-elements-what-is-before-and-after/
- http://html-plus.in.ua/stili-dlya-spiskov/

As for the ordered lists, they usually use only the pseudo-element :before to change the numbering, specifying some counter name (for example, number) in several properties:

```
ol { list-style-type: none;
     counter-reset: number; }
ol li::before {
     content: counter(number) ') ';
     counter-increment: number;  }
```

Result:

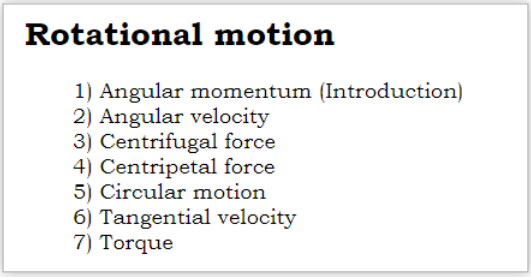


Figure 27

For the unordered lists, the content property specifies, for example, a text with a space instead of a marker:

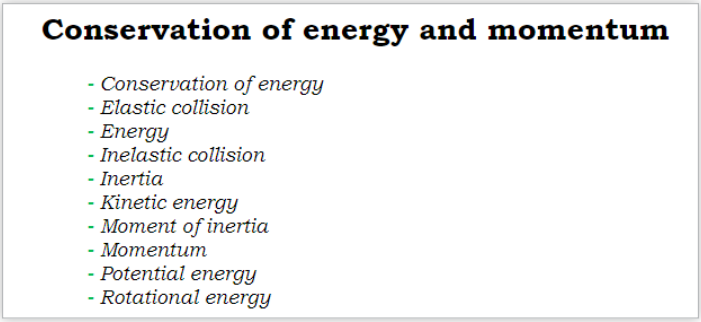


Figure 28

```
ul {  list-style-type: none;
      font-style: italic;    }

ul li::before {  content: '- ';
                 color: #04b548; }
```

Style of multi-level lists. Nested selectors

To style a multilevel list, it is not enough to write css-rules only for the element selector (ol, ul, li). More often it is necessary to apply context (or nested) selectors for them (and not only). They imply that css-properties are assigned to a tag nested in another tag, for example, the selector

```
ol ul {color: red}
```

implies that only the bulleted list (ul) that is nested in the numbered list (ol) will be red.

The number of selectors can be any, but it should be understood that css-rules are assigned to the lattermost selector, but taking into account its nesting in all previous ones. Thus, this selector is read from right to left. For example, the code

```
.season ul li { font-style: italic; }
```

says that only the li elements of the bulleted list ul that is in the element with the season class will have the italic style.

For example, you need to make a list of seasons as shown in the image (fig. 29).

Figure 29

If you analyze its appearance, it becomes clear that this is a numbered list with 2 levels of nesting, and the nested list can be both numbered and bulleted, because numbers or markers will still be covered by images. Since the numbering will not be visible, it makes sense to make the nested list bulleted. Therefore, the **initial html-markup** will be as follows:

```
<h2>Seasons</h2>
<ol>
    <li>Winter
        <ul>
            <li>December</li>
            <li>January</li>
            <li>February</li>
        </ul>
    </li>
    <li>Весна
        <ul>
            <li>March</li>
            <li>April</li>
            <li>May</li>
        </ul>
```

```
    </li>
    <li>Лето
        <ul>
            <li>June</li>
            <li>July</li>
            <li>August</li>
        </ul>
    </li>
    <li>Осень
        <ul>
            <li>September</li>
            <li>October</li>
            <li>November</li>
        </ul>
    </li>
</ol>
```

The appearance of the list is presented in the screenshot (fig. 30).

Figure 30

Now we need to make the font of the parent list <ol> bold, and that of the child list <ul> normal. Since the <li> elements

inherit the properties of the parent element, we assign the following property to the selector ol:

```
ol { font-weight: 600;  }
```

And as a result we get a list in which **all** the elements have become bold (fig. 31).

Figure 31

In order for the nested list to get the normal style back, you must specify the following css rule:

```
ul { font-weight: 200; }
```

Now the list has the following appearance: the elements of the parent list are bold, the elements of the nested list are normal (fig. 32).

The next step is to assign a *color* to different blocks of text: blue for the winter months, green for the spring months, etc.

Figure 32

The easiest way to do this is to set a class or id to each of the li elements in the parent list. Since the color values will be used individually for each month block, we use the id attribute:

```
<ol>
    <li id="winter">Winter
        <ul>
            <li>December</li>
            <li>January</li>
            <li>February</li>
        </ul>
    </li>

    <li id="spring">Spring
        <ul>
            <li>March</li>
            <li>April</li>
            <li>May</li>
        </ul>
    </li>
```

```
    <li id="summer">Summer
        <ul>
            <li>June</li>
            <li>July</li>
            <li>August</li>
        </ul>
    </li>

    <li id="autumn">Autumn
        <ul>
            <li>September</li>
            <li>October</li>
            <li>November</li>
        </ul>
    </li>

</ol>
```

Now you can assign colors for each of the id selectors:

```
#winter {  color: rgb(0, 0, 200);   }
#spring {  color: rgb(0, 190, 0);   }
#summer {  color: #ffd24c;   }
#autumn {  color: #d96c00;  }
```

The appearance of the blocks has changed in accordance with the assigned colors (fig. 33).

The last task is to define markers in the form of different images. It would be possible to assign to each of the 3 elements <li> the class attribute and set the list-style-image property for it, but it's simpler to use a context selector of this type:

Figure 33

```
#winter li { list-style-image: url(markers/snow.png); }
#spring li { list-style-image: url(markers/leaf.png); }
#summer ul { list-style-image: url(markers/sun.png);  }
#autumn ul { list-style-image: url(markers/rain.png); }
```

As a result, we get the list appearance we need (fig. 34).

Figure 34

Note that in the first two selectors, we assigned an image marker for the li element nested in a specific id selector, and in the third and fourth selectors — for the ul element. In this case, the difference is not visible. This happened for the reason that the properties assigned to the entire list (ul in the example) are inherited by its li elements, and the properties for the li elements are applied to them directly.

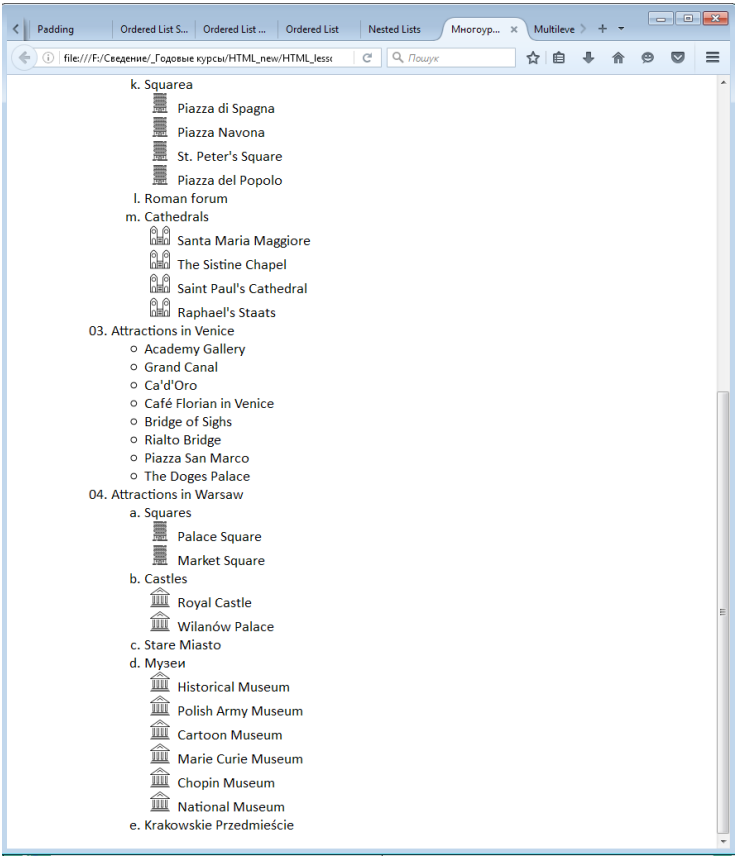


Figure 35

The sample file is *seasons.html*.

In the file *multilevel-list-style.html* you will find css-formatting for the multilevel list considered earlier.

Related links:

- https://webref.ru/css/selector/descendant
- http://html-plus.in.ua/vlozhennye-spiski/

CSS properties for indents and margins

Indents between blocks of text are very important for layout. And it does not matter whether you mean the layout for a book or magazine or a web page. It is impossible to read text that has no indents, because in this case, the text is a solid block, which simply scares with its appearance. Therefore, CSS provides 2 types of indents: external ones, called margin, and internal ones with the name padding.

It should be noted that we will consider any kind of indents for block elements, because строчные элементы display them somewhat differently. This difference will be discussed in the lesson 'Box Model of Elements' in detail.

Margins and Padding. General Information

The padding property defines the distance from the content of the block element to its boundary, and margin is the distance between the boundaries of the elements that are located side by side.

Margins are transparent, and padding can be filled with background color. Margins have the effect of collapsing, that is, combining values of neighboring elements. Margins can have negative values, and padding — only positive ones. Margins and padding can be written either in abbreviated form or for each side separately.

Indents can be assigned from four sides of the element: top, right , bottom and left:

- padding-top/margin-top — top indent;
- padding-right/margin-right — right indent;
- padding-bottom/margin-bottom — bottom indent;
- padding-left / margin-left — left indent;
- padding/margin — abbreaviated form.

Margins

Margins can be specified for all sides of an element, for example:

```
margin: 10px;
```

You can specify margins for the top and bottom edges of the element (first value) and for the left and right side of the element (second value) separating by a space:

```
margin: 10px 2em;
```

Three values are used for:

- Top side,
- Left and right sides
- Bottom side of the element

```
margin: 10px 1.5% 30px;
```

Four values define the values of the margins in the following sequence: from top right to bottom left

```
margin: 15px 10% 30pt 0;
```

The order of margin values starts from the upper left corner and goes clockwise:

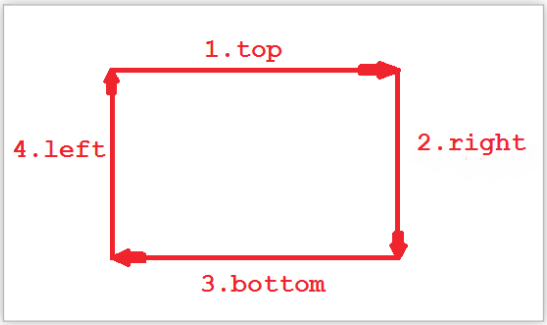


Figure 36

Margins can be specified in any units of measurement (px, pt, em, rem, % etc.). You can specify negative values, auto (the browser calculates the value automatically) or 0 (no margin). In the latter case, the units of measurement are not specified, because 0 either in %, or in px, or in em is all the same 0.

You can also assign only one of the margins:

```
margin-right: 20px;
margin-left: 2%;
margin-top: 10pt;
margin-bottom: 2em;
```

You can also assign a common margin for all sides, and then undo or override it for one of them:

```
margin: 20px;
margin-right: 0;
```

You can find the options for setting the margin property in the *margin.html* file. In it margins are displayed in beige color, and the block which they are applied to is white.

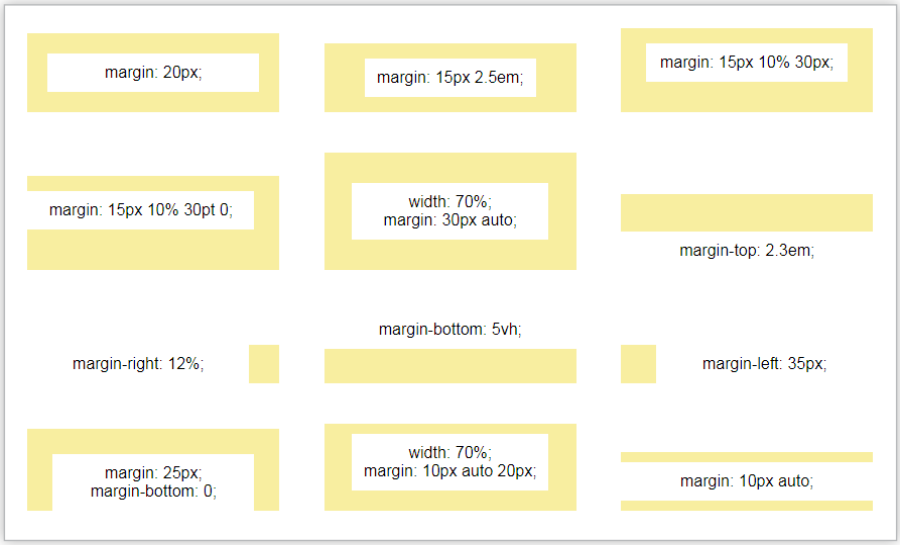


Figure 37

The auto value is interesting. Using it together with the width property, you allow the browser to calculate the value of margin by itself, based on the size specified for the width of the selector.

For example, if you specify the following properties for body:

```
body {  width: 1000px;
        margin: auto;}
```

with a browser width of 1340px, the browser will reserve 1000px to display the content, and will split the remaining 340px in half and will form margins of 170px to the right and left of the content, centering it. The same will happen if the width is specified in %. The difference is only that the browser first calculates the width of the element in px, based on the width

of the browser or the width of the parent element, and then calculates the value of margins.

This is usually used to indicate margins not for body, but for some wrapper element, which often has a class name or id wrap, wrapper, container, content, and so on. And margin is set to 0 for body, because by default the body has margins of 8px.

```
body { margin: 0}
.wrapper {width: 80%; margin: auto;}
```

Another feature of the margin property is the collapse of vertical margins in the elements that follow one another. For example, if you specify such css rules for elements with the block class:

```
.block {  margin: 40px 0;   }
```

you can expect that between the blocks that follow each other, the margins will be 80px. However, actually, the margins are superimposed, and the general margin will be of 40px (file *margin-collapse.html*).

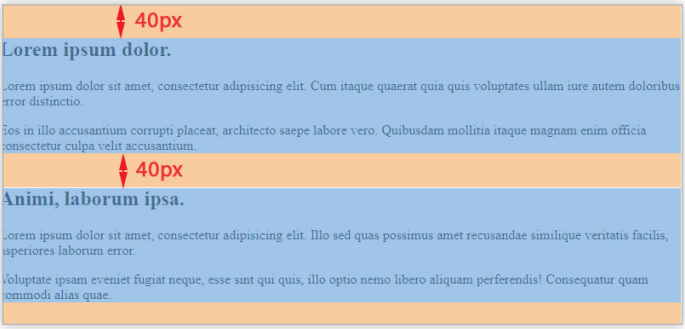


Figure 38

The collapse only works for vertical margins, but does not apply to horizontal ones. In addition, the collapse does not work:

- for the <html> tag
- for inline elements;
- for elements that have the property float: left or right
- for absolutely positioned elements
- for elements that have the padding property set on the collapse side
- for elements that have the border property set on the collapse side

```
Emmet abbreviation: m20, m2em,mb2%,mt,ml,mr:7,m:a,m20px10
```

Padding

Padding extends from the content of the block element to its border (css-property border), if it is specified. When you fill an element with a background color, padding usually also has this color.

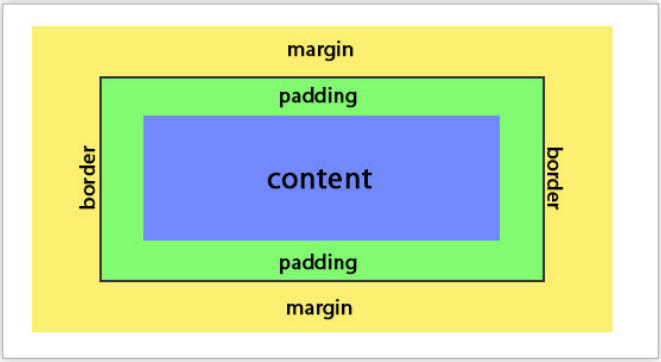


Figure 39

Paddings cannot be negative and they do not collapse like margins. The similarity of these two indents is that padding can be specified in px, pt, em, rem, %, etc., and you also can do this on the 4 sides of the block in the same way as for margin. You can also use the value inherit, which indicates that it is inherited from the parent.

One value for all sides:

```
padding: 20px;
```

Padding on the horizontal (*top and bottom*) and vertical (*right and left*) sides:

```
padding: 2em 10px;
```

Padding on top, padding for left and right side and padding from below:

```
padding: 2vw 3% 10px;
```

Padding from above, the right, below, the left:

```
padding: 1rem 5% 0 20px;
```

A separate padding for each side:

```
padding-top: 15px;
padding-bottom: 1rem;
padding-left: 5%
padding-right: 14pt;
```

The general value of paddings with a different one from one (left) side:

```
padding: 20px; padding-left: 3px;
```

Visually blocks with all these values will look as follows: You can find all the values in the *padding.html* file.

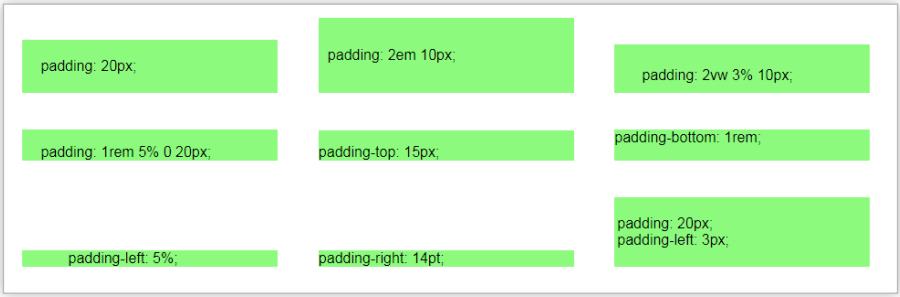


Figure 40

Another difference between margins and paddings is that the property padding: auto is not applied, because there is no auto value for padding.

```
Emmet abbreviation: p20, p2em,pb2%,pt,pl,pr:7, p20px6
```

- **Note:** *when assigning padding values in %, it should be understood that percentages are calculated based on the width of the parent element, not the current one, so the 5% padding for an element with a width of 200px that is placed inside a block with a width of 1000px will be 50px (1000px*5%/100% = 50px), and not 10 (200px*5%/100% = 10px).*

Default indent values

To ensure that different html-elements look good even without css-formatting on the page, they have a number of css-properties assigned by default. They are pulled from

the *user agent stylesheet*, which is created by developers based on W3C recommendations. The properties such as margin and padding also have default values.

The elements of lists considered today have such indents by default:

```
ul, ol {margin-top: 1em;
    margin-bottom: 1em;
    padding-left: 40px; }
```

It is important to note that the nested ol or ul elements in the multilevel lists do **NOT have indents** at the top and bottom.

In the description lists, the default margins are:

```
dl { margin-top: 1em;
    margin-bottom: 1em;
}
dd { margin-left: 40px;}
```

Margins are also used for headers:

```
h1 { margin-top:    0.67em;
    margin-bottom: 0.67em;
}
h2 { margin-top:    0.83em;
    margin-bottom: 0.83em;
}
h3 { margin-top:    1em;
    margin-bottom: 1em;
}
h4 { margin-top:    1.33em;
    margin-bottom: 1.33em;
}
```

```
h5 { margin-top:    1.67em;
     margin-bottom: 1.67em;
}
h6 { margin-top:    2.33em;
     margin-bottom: 2.33em;
}
```

Here there is such a tendency: the larger the number of the header is, the smaller the font size is, but the larger the top and bottom margins are.

Margins of headers are often necessary to be changed, because in psd-mockups, designers either increase or decrease them. The default paragraphs also have horizontal margins of the same size as h3:

```
p { margin-top:    1em;
    margin-bottom: 1em;
}
```

For a block quote, margins are assigned by default for all sides, but of different values:

```
blockquote {
margin-top:    1em;
margin-bottom: 1em;
margin-left: 40px;
margin-right: 40px;
}
```

For the body element, as already mentioned, the default margins are 8px:

```
body {margin: 8px;}
```

Elements such as html, div, section, article, aside, main by default do NOT have any margins or paddings, so if necessary, you can assign them yourself.

To remove indents, you must assign 0 as the value:

```
body { margin: 0;}
```

If you want to remove the default indents of all elements, use the 'simplest reset' setting rules for the universal selector:

```
* { margin: 0; padding:0;}
```

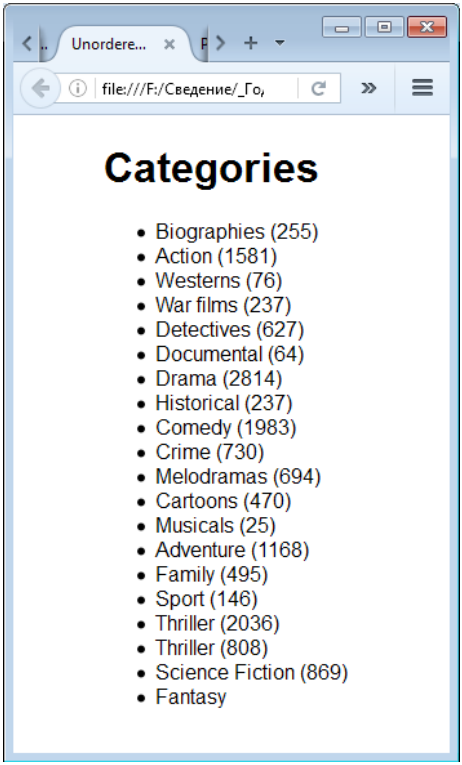


Figure 41

Note that with this approach, the list markers will 'go outside' the browser, if the body or other parent element does **not have** margin-left added.

Also, to bring the styles of all page elements to a uniform appearance there is the reset.css file proposed by CSS-guru Eric Meyer. This is done by removing indents and assigning one font size. Now a different approach is used more often: normalization of the element appearance due to the use of the file normalize.css, which introduced uniform styles for all elements whose css-formatting differs depending on the browser and its version.

To find out what are the css-properties of any page element, you can use the *development tools* provided in any browser, but we'll talk about this in detail in the next lesson.

Related Links:

- https://meyerweb.com/eric/tools/css/reset/
- https://habrahabr.ru/post/45296/
- https://webref.ru/course/css-basics/css-reset
- https://necolas.github.io/normalize.css/
- https://htmlacademy.ru/blog/64-about-normalize-css

Home Task

At home, you will have to create and format several multi-level lists using css-rules. In the folder with the home task you will find images with the appearance of lists, texts to them in txt-files and image-markers in the folder markers.

To center the content, use a combination of width and margin: auto.

In order to make it easier to put text from a txt file into html-tags, it makes sense to use the *Emmet-wrapper*, namely:

- Select the desired text;
- Press Ctrl+Shift+A;
- Enter the desired abbreviation;
- Press the Enter key.

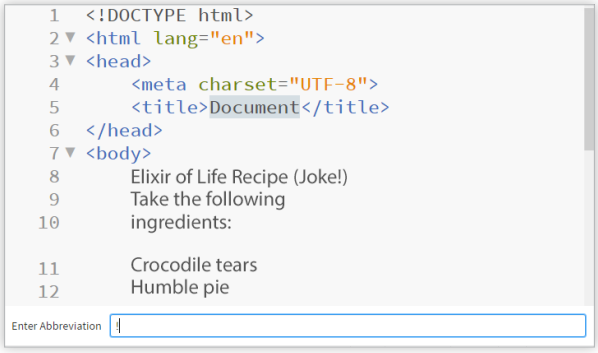


Figure 42

For example, to add structure tags using the Emmet wrapper, select all the text (Ctrl+A), press Ctrl+Shift+A, and then enter an exclamation mark (!):

To put the list text inside the list tags, select it and use the abbreviation:

```
ol>li*
```

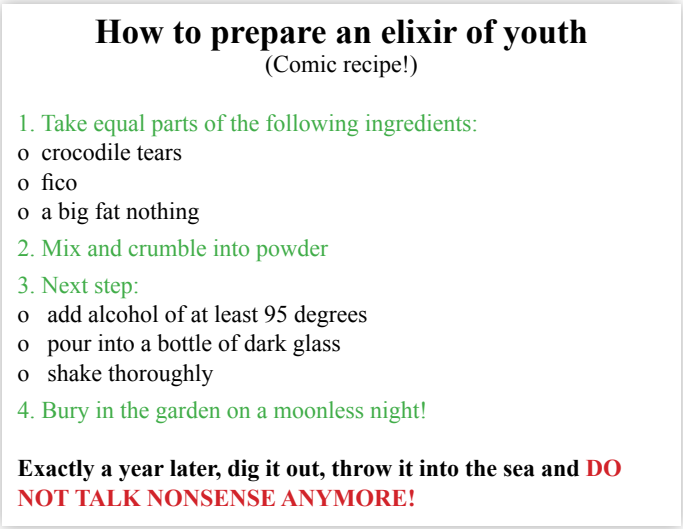
In this case, the ol tags appear at the beginning and at the end of the selected text, and the <li> tags wrap all the lines separated by a Enter key in the txt file.

Appearance of Tasks

Please, pay attention to the font family and its face.

For style design, use context selectors, class selectors, and id.

Figure 43



How to prepare an elixir of youth

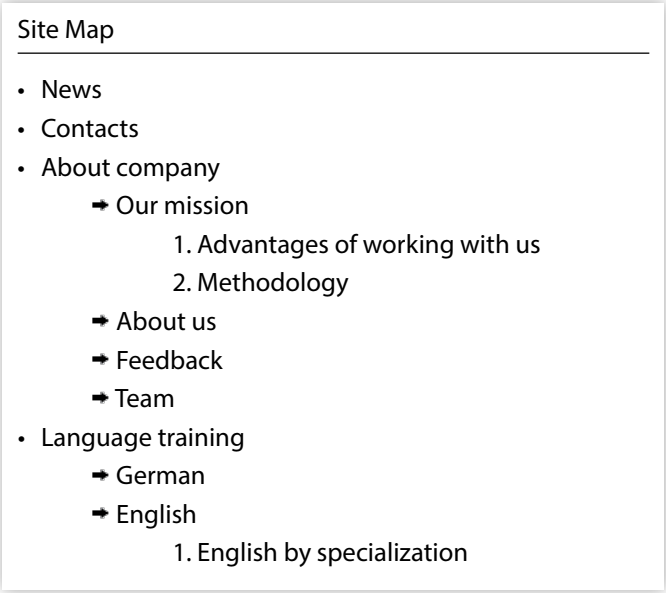
(Comic recipe!)

1. Take equal parts of the following ingredients:
   - o crocodile tears
   - o fico
   - o a big fat nothing
2. Mix and crumble into powder
3. Next step:
   - o add alcohol of at least 95 degrees
   - o pour into a bottle of dark glass
   - o shake thoroughly
4. Bury in the garden on a moonless night!

Exactly a year later, dig it out, throw it into the sea and DO NOT TALK NONSENSE ANYMORE!

Figure 44

Note that the heading 'Site map' is shifted to the right and has a normal face. Use margins and padding.



Site Map

- News
- Contacts
- About company
  - ➡ Our mission
    1. Advantages of working with us
    2. Methodology
  - ➡ About us
  - ➡ Feedback
  - ➡ Team
- Language training
  - ➡ German
  - ➡ English
    1. English by specialization

2. Preparation for exams
3. Business English
4. General English
5. Conversation Club
6. Special intensive courses
7. New competition "Master Mind"

- Services
  - ➡ English for children
  - ➡ Trainings
  - ➡ Translation Agency
  - ➡ Study abroad
- Loyalty programs
- Our newsletter
- Practical tips
- FAQ
- Site Map

Figure 45

Lesson 3
Introduction. HTML Structure