HyperText Markup Language

HTML&CSS

Unit 1

Introduction. HTML Structure

Contents

What is HTML. History of creation

HTML (*HyperText Markup Language*) is currently the standard for documents transmitted on the Internet. HTML markup is interpreted by browsers in the form of formatted text and can be displayed on the screen of a computer monitor, tablet or smartphone.

HTML is based on **SGML** (*Standard Generalized Markup Language*), which appeared in 1986, and corresponds to the international standard ISO 8879. This language was originally intended for text structural markup, but did not contain a description of document appearance that could be created using it.

SGML meant a description of the syntax for writing the main elements of markup, and even then they were called tags, it in itself was not a system for marking text and did not have a list of structural elements of the language in order to use them when creating a document.

Nevertheless, there was a need to create a hypertext language, and it should have:

- a description of the elements and their application;
- a list of elements for a document that can be displayed by special programs.

That is why in 1991 the *European Organization for Nuclear Research* (**CERN**) in Geneva, Switzerland, announced the need to develop a mechanism that allows the transfer of hypertext information through the Global Network. The em-

ployee of this organization, Tim Berners-Lee started the development of this language. And it is the SGML standard that formed the basis of the future **HTML language** — *Hyper Text Markup Language*.

In accordance with the requirements and needs of CERN, Tim Berners-Lee developed HTML primarily for the exchange of scientific and technical documentation. And this language should have been used without any special problems by people who are not experts in the field of markup. A small *set of structural and semantic descriptor elements* was defined, *which then began to be called* ***tags*** that allowed to create fairly simple and at the same time beautifully designed documents. Thus, HTML successfully solved the problems left by SGML.

Working in CERN, Tim Berners-Lee dealt not only with the development of HTML. His task was to build an internal network of the organization. The concepts implemented in it were further improved and developed into a project called the *World Wide Web*.

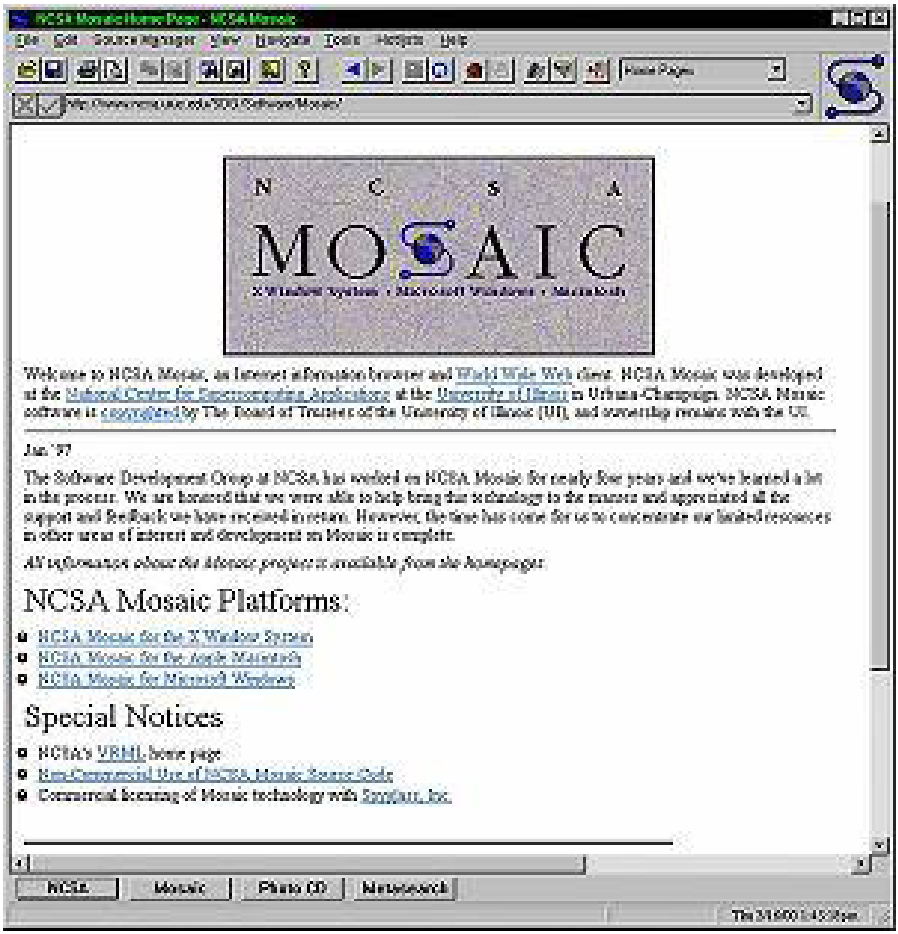
The project implied that it would be possible to publish documents marked using HTML on open access. All documents should have hyperlinks to each other, which allows them to be linked together, turning them into a web-like formation.

To implement his idea, Berners-Lee created special programs: HTTP-server and WEB-browser. The world's first web-

site was posted on August 6, 1991 at http://info.cern.ch/ (you can see its archive version here). Its content describes how the network works, how to install the web server and create a simple page.

HTML Standards

In 1993, HTML 1.2 appeared, which contained only 40 tags. These tags, unfortunately, did not give a full page design.



To display HTML-pages, a special program was developed, which was called "browser". And the first browser was "Mosaic", which was developed at the *National Center for Supercomputer Applications* (**NCSA**, USA). For the first year, about two million copies of this program were installed. It supported the display of pictures, was distributed free of charge and placed on one floppy disk.

The next step in the development of the HTML standard was the creation of the **W3C** (*World Wide Web Consortium*) in April 1994.

Since the official HTML 1.0 specification did not exist, it was the W3C that began to prepare the HTML specification for the next version. It was immediately assigned a number 2.0 to emphasize the difference from other versions of HTML. The first result was obtained after a year of intensive work — in 1995. Out of the large additions, we should note the creation of a form mechanism for sending information from the user's computer to the server.

In parallel with the development of HTML 2.0, the 3rd version of the standard was developed, which appeared in March 1995 and contained tags to create:

- mathematical formulas;
- pages;
- notes;
- insert pictures wrapped by text;
- support for gif format, etc.

While this standard was compatible with the second version, its implementation was difficult for browsers of that time.

In addition, in 1995, there was a need for a greater visual variety of pages. The tools offered by HTML, especially within the SGML standard, were not enough. Then the W3C corporation started to create an additional system that does not contradict the basics of HTML, but at the same time allows to describe the visual design of documents. As a result, *CSS* appeared — Cascading Style Sheets, which were hierarchical style specifications. They had their own syntax, structure and tasks, which allowed adding HTML tags with visual formatting. Creating CSS was a big step forward, because the need for a visual representation of the pages greatly increased, and HTML did not offer any means for this.

Browser War

By the way, Mozaic was no longer the only browser. Since 1994, **Netscape Navigator** and **Opera** have appeared on the network, and since 1995 — **Internet Explorer**. To grab the largest piece of potential audience, to attract the maximum number of new users, **Netscape** introduced new and new improvements in HTML, which were supported, of course, only by the same browser. Almost all new tags were aimed at improving the appearance of the document and expanding its formatting capabilities.

In the summer of **1996,** version 3.0 of Internet Explorer was released. It supported almost all the extensions of Netscape and had a pretty and user-friendly interface. Therefore, it very quickly established itself as a «second main browser».

The fourth versions of both browsers were released almost simultaneously and did not differ from each other with

a special speed of work or other parameters. While Netscape needed to be bought, Internet Explorer began to be delivered free of charge in the Windows operating system and became virtually the industry standard. The 90s of the 20th century were marked by the notion of "Browser War», which, in fact, denotes confrontation between Netscape Navigator and Internet Explorer on the Internet market. This war was a problem for the coder, because each browser tried to contribute to the development of the HTML language and did not follow the W3C standards. Therefore, on the pages of the websites it was often possible to find the entry «Correctly displayed in the browser ...» with the link to the downloading of the corresponding program.

It should be noted that in the late 1990's and early 2000's due to the fact that IE was a default Windows browser, it became the most popular one, especially for not very experienced users. Thanks to this approach, at one time Microsoft Corporation captured the lion's share of website views on the Internet and drove Netscape Navigator out of a market, although it could not retain its championship.

The Mozilla FireFox was born based on Netscape Navigator, and supported the W3C standards, had tabs, not windows and a number of tricks that enticed advanced users and developers. The Opera browser also gained a lot of fans. The next new browser was Google Chrome, which was released on December 11, 2008, and its source code became available under the name Chromium. From that moment, any company could make its browser based on this code. For example, Yandex-browser or Amigo appeared so. In ad-

dition, Opera also switched to the WebKit engine on which Chrome was created.

Chrome slowly but surely took a place in the browser market, and by now is the winner, because it is used by about 50% of users to surf the Internet. In addition, Google Chrome provides convenient tools for developers of source code — html-developers, JavaScript programmers, so we will use it first.

It should be noted that the *developer tools* are now present in any browser, including the Internet Explorer of the latest versions. And not the least role here was played by the add-on for FireFox with the name FireBug, which provided information about html-elements, css-rules for them and allowed debugging JavaScript code. In Opera, for the same purposes, there was the Dragonfly extension.

Let's dwell on the concept of "**cross-browser**». It implies the same display of the site in all browsers. At the moment, this problem is not as acute as in 2005-2012. All modern browsers are trying to support the standards of the W3C organization. Nevertheless, when you finished a site, you should check how it is displayed in the most popular browsers at the moment. To do this, look at the statistics for today, for example, on the site https://www.w3schools.com/Browsers/default.asp:

2017	Chrome	IE/Edge	Firefox	Safari	Opera
August	76.9 %	4.3 %	13.1 %	3.0 %	1.2 %
July	76.7 %	4.2 %	13.3 %	3.0 %	1.2 %
June	76.3 %	4.6 %	13.3 %	3.3 %	1.2 %
May	75.8 %	4.6 %	13.6 %	3.4 %	1.1 %
April	75.7 %	4.6 %	13.6 %	3.7 %	1.1 %
March	75.1 %	4.8 %	14.1 %	3.6 %	1.0 %
February	74.1 %	4.8 %	15.0 %	3.6 %	1.0 %
January	73.7 %	4.9 %	15.4 %	3.6 %	1.0 %

2016	Chrome	IE/Edge	Firefox	Safari	Opera
December	73.7 %	4.8 %	15.5 %	3.5 %	1.1 %
November	73.8 %	5.2 %	15.3 %	3.5 %	1.1 %
October	73.0 %	5.2 %	15.7 %	3.6 %	1.1 %
September	72.5 %	5.3 %	16.3 %	3.5 %	1.0 %
August	72.4 %	5.2 %	16.8 %	3.2 %	1.1 %
July	71.9 %	5.2 %	17.1 %	3.2 %	1.1 %
June	71.7 %	5.6 %	17.0 %	3.3 %	1.1 %
May	71.4 %	5.7 %	16.9 %	3.6 %	1.2 %
April	70.4 %	5.8 %	17.5 %	3.7 %	1.3 %
March	69.9 %	6.1 %	17.8 %	3.6 %	1.3 %
February	69.0 %	6.2 %	18.6 %	3.7 %	1.3 %
January	68.4 %	6.2 %	18.8 %	3.7 %	1.4 %

And at the moment it's worthwhile to pay attention to the use of mobile browsers.

HTML Standards (continued)

But let's be back to the history of HTML standards.

The HTML 3.1 version was never officially offered, so the next version of the HTML standard was 3.2, released on January 14, 1997. Many new features of the version 3.0 were omitted in it. However, non-standard elements were added, supported by the Netscape Navigator and Mosaic browsers.

On December 18, 1997, the fourth version of HTML was adopted. The **HTML 4.0** standard contained, as the third version, many elements specific to individual browsers. It should be noted that in the 4th version many items were marked as obsolete and not recommended for use. It was recommended to use CSS style sheets instead.

Next comes the standard HTML 4.01, which was approved on December 24, 1999. It made significant changes,

and this standard has been popular on the web for quite some time.

In parallel with the development of the HTML standard, an alternative standard **XHTML** (*Extensible Hypertext Markup Language*) was also being development, which is an extensible hypertext markup language that uses the XML approach. It has stricter syntax requirements than HTML. XHTML 1.0 was approved on January 26, 2000 as a recommendation of the W3C. The XHTML version 1.1 was approved as a consortium recommendation on May 31, 2001.

In fact, the development of new HTML standards has never ceased, because the requirements for displaying pages are constantly increasing. Since 2007, the HTML5 standard was developed, which was finally approved on October 28, 2014. And since December 17, 2012, the development of the HTML 5.1 standard is already underway.

We will study the HTML5 version. This standard, in comparison with the previous ones, added many new tags, such as <canvas>, <nav>, <section>, <header>, <footer>, <main>, etc., new form elements that expanded the possibilities for structuring the html-document.

In addition, we will use the **CSS3** standard to visualize the pages, which also added a lot of new properties that make it easy to create beautiful web pages without using pictures or js-code.

It should be noted that many css-properties that we will use refer to the previous version — **CSS 2.1**. But this is understandable — CSS3 became the heir to the previous standard, adding and expanding its capabilities.

As in the standardization of HTML, the W3C organization does not sit still, but continues to introduce new properties that will already fall into the next standard — CSS 4.0. The beauty of this process is that the new properties are already supported by the latest versions of browsers. And the disadvantage of such a standard development is that the syntax of a property or of a whole group of properties can change 2 or even 3 times until it is finally approved, as it was with flexbox.

Basic HTML definitions

Creating an HTML document involves marking all of its contents in the form of tags (the first name — descriptors), which are written in angle brackets. The text should be placed between the opening (initial) and closing (final) tags. For example, paragraph tags with text look like this:

```
<p>Paragraph text</p>
```

There is one more concept — this is the concept of an **element**. *All text between the start and end tag, including the tags themselves, is called an element*. The very text between the tags is the **element content**. Note that the element content can include any text, including other elements.

For any tag, you can specify additional properties, which are called **attributes**. They are placed only in the opening tag and are pairs of the form 'property="value"', separated by spaces.

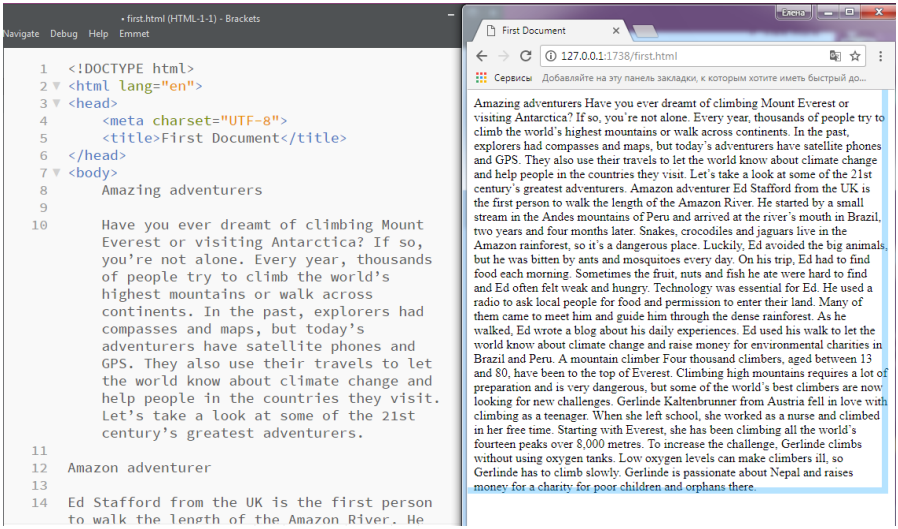
```
<p id="test" class="par">Paragraph text</p>
```

HTML is a case-insensitive markup language, so it is acceptable to write tags in both uppercase and lowercase.

Nevertheless, the de facto rule has long been to write tags in lowercase.

HTML documents, in fact, are text documents. They have the extension .html or .htm, and you need a browser to view them, and a code editor for editing; we'll talk about it selection below.

Another feature of HTML is the ignoring of spaces and line breaks. No matter how the text was formatted in your html document BEFORE adding tags — with indentations made with the TAB key, with the break of each line (ENTER key) — in the browser it will be just plain text.



Therefore, all the text inside the html-document must be divided into tags according to the semantics, or the internal logic of this document. And for different levels of nesting elements that are likely to be present on your pages, you should make indents to visually separate the parent and child elements:

```
<div class="modal fade" tabindex="-1" role="dialog" id="cartDetails">
    <div class="modal-dialog modal-lg">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal" aria-
                label="close"><span aria-hidden="true">&times;</span>
                </button>
                <h2 class="text-center">Cart</h2>
            </div>
            <div class="modal-body">
                <p>No items in cart yet</p>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-order" data-dismiss="modal" data-
                text="Continue shopping"></button>
                <button type="button" id="checkoutBtn" class="btn btn-success" data-
                text="Place an order">Place an order</button>
            </div>
```

Now this code seems completely incomprehensible, but will eventually become familiar.

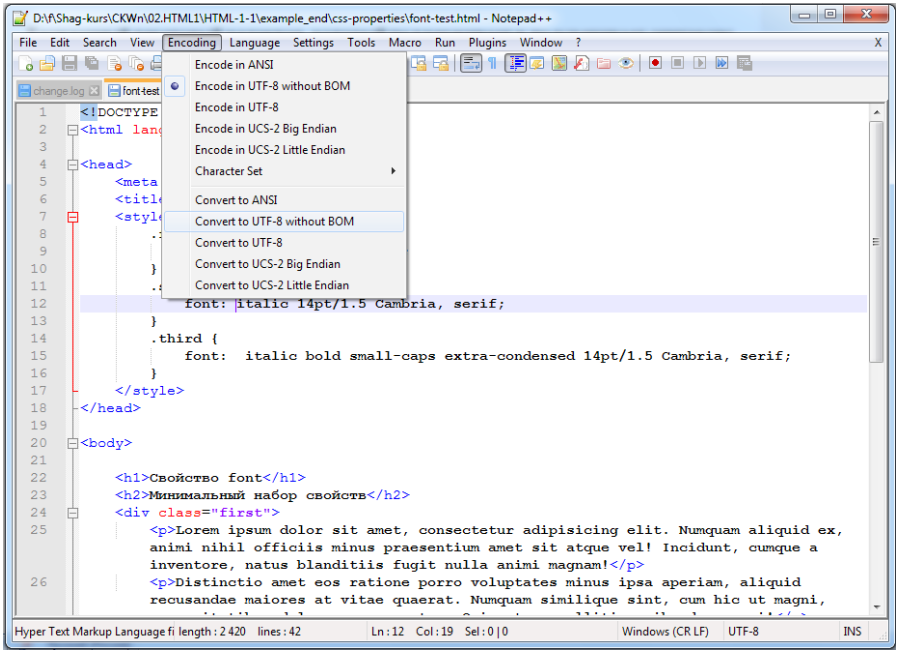
And now let's talk about the main tool for the coder — the code editor.

Selecting the code editor

At the moment there are a lot of code editors for HTML/CSS. Consider the most popular of them.

Notepad++

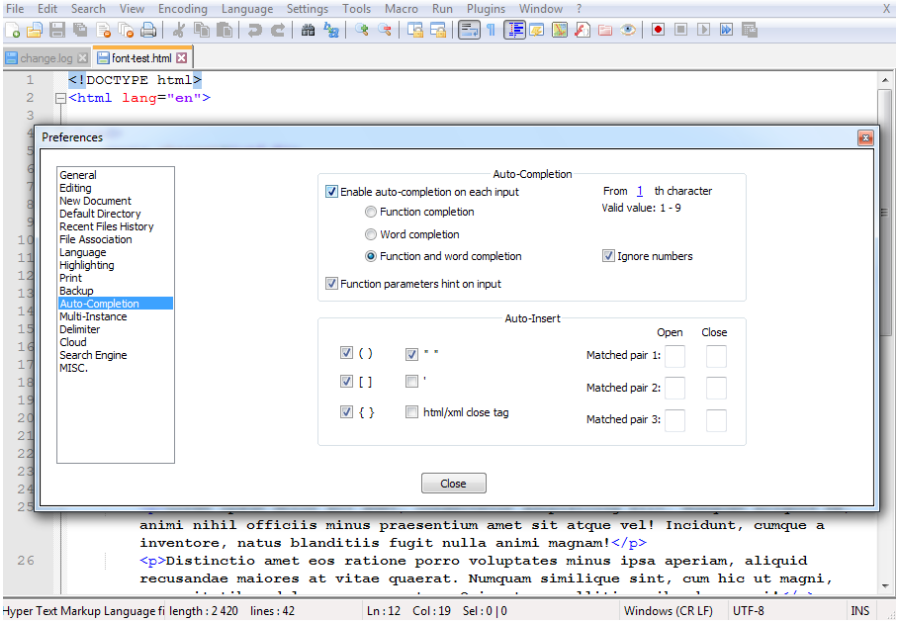
A free text editor that supports and highlights the syntax of over 100 languages. Can open files in different encodings. On the official site https://notepad-plus-plus.org/ you can download the latest version of the program.



This code editor is convenient, quickly loaded. In it, you can customize some functions through the Options -> Set-

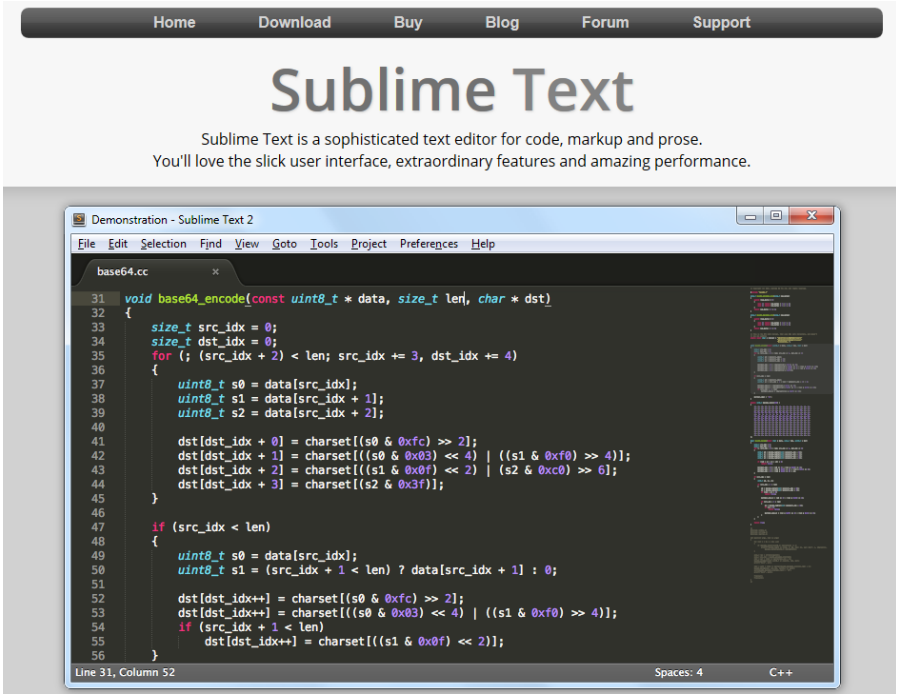
tings menu, for example, autocomplete, i.e. closing brackets, tags, and so on.

Notepad++ is designed only for Windows 32 and 64bit systems.



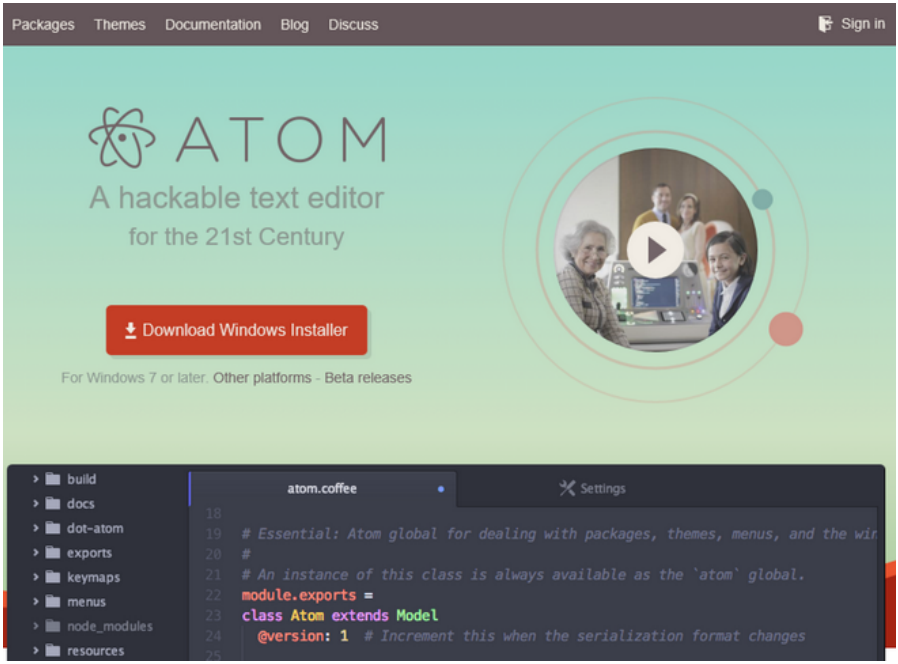
Sublime Text

Perhaps the most popular text editor. Conditionally free. Thus, you can download it without payment, but after a certain time it will offer you to buy a license. Official website https://www.sublimetext.com/ currently offers the 3rd version of the editor for download.



Sublime Text allows you to work with many formats. It easily copes with large amounts of textual information. The editor is expanded by installing additional packages. The downside is that for setting up Sublime Text you need to spend about 2-3 hours, because the OOB functionality will not be enough for you.

Atom



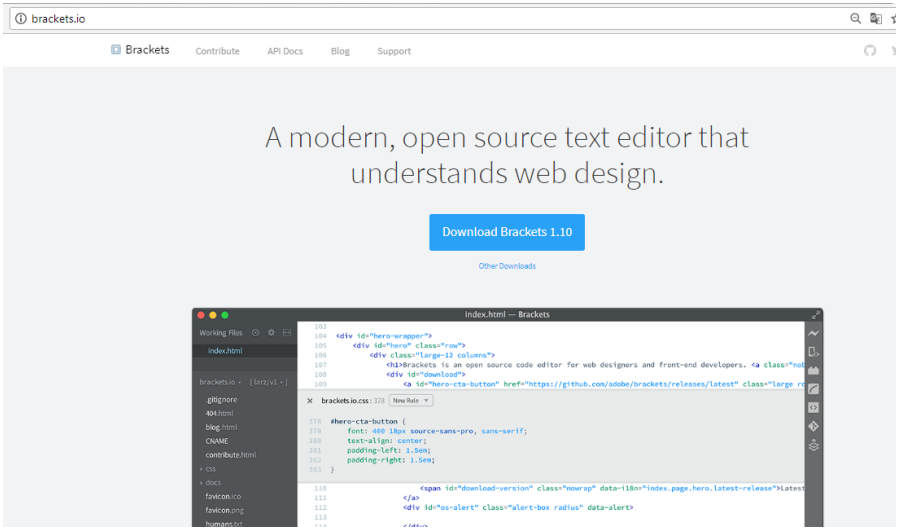
Quote from the official site https://atom.io/:

'Atom is a text editor that's modern, approachable, yet hackable to the core — a tool you can customize to do anything but also use productively without ever touching a config file.'

This editor was created by the Github team. It is free, open source, suitable for macOS, Linux, Windows. It is expanded with the help of plugins written in Node.js, which are embedded under Git Control. Also you can install and change themes on it, choosing color combinations for code highlighting that will be convenient for your eyes.

Brackets

A free text editor that supports opening files only in UTF-8 format — the most common at the moment in the web. It is designed by Adobe System to work primarily with HTML, CSS, JavaScript. But it also allows you to work with php-files and has a number of extensions to create themes for Wordpress, for example. The official site http://brackets.io/ allows you to download the latest version of the editor.



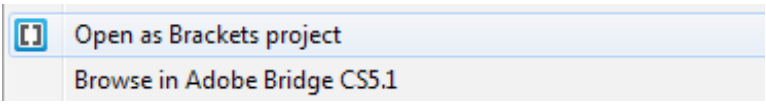
Brackets is designed for Windows, MacOS and Linux systems. Expanded with the help of plugins, also free, open source, hosted on Github.

We will focus on this editor, since it has a lot of «goodies» right «out of box» and allows you to simplify and speed up the writing of the code. In the conditions of a short time for training or order fulfillment, this can be very convenient.

Using Brackets to create and edit files

The first moment is that in Brackets, as well as in other editors, it is better to work with projects — in fact, it's just a folder on your computer or on a flash drive that will contain html and css files, as well as folders with images and further with js-scripts.

Therefore, first of all, create a folder on your computer where you will work (I call it HTML-1-1) and select the "Open as Brackets Project» item in the context menu for this folder:



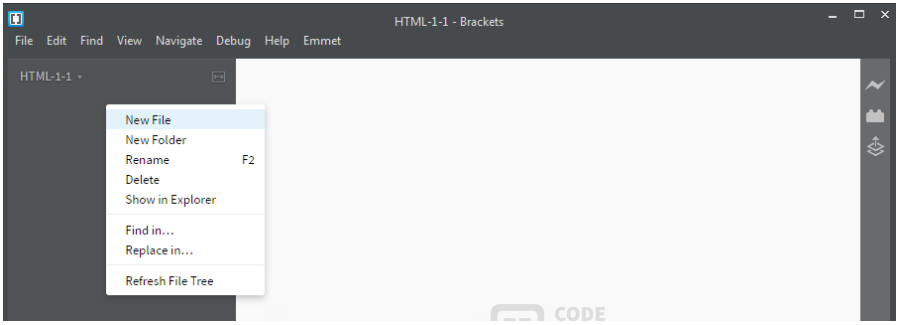
In this case, all files and folders within the project will be accessible to you. Especially you appreciate it when you consider the topic of links or images.

- **Note:** *By the way, if the Brackets window is already open, you can simply drag the project folder to the left side of the editor or open the desired directory by clicking on the arrow button.*

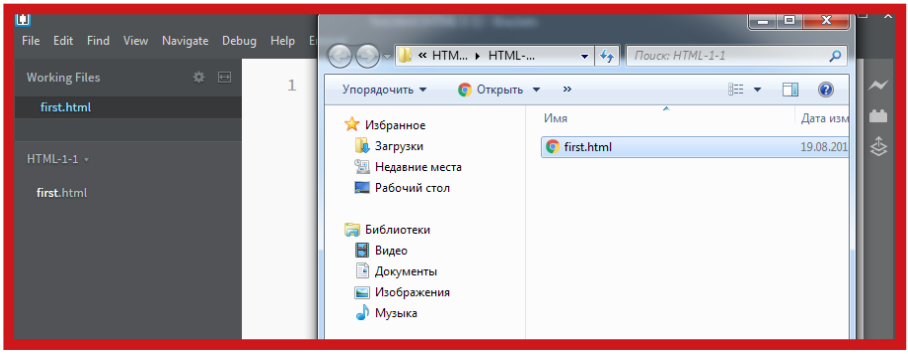
Creating files in Brackets

You can create files in the .html or .htm format (namely this format is used on the web) via File -> New (or Ctrl+N) and then save the file to the desired folder.

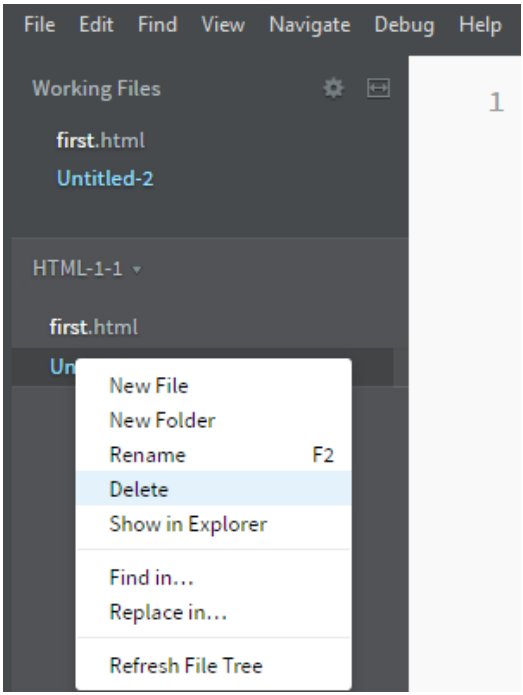
But it is much more convenient to create a file in the left dark-gray area of the Brackets by right-clicking on it. In the context menu select the first option New File:



By default, the new file will have the name «Untitled-1» and (attention !!!) no extension. Therefore, instead of this, enter first.html — voila — the file is simultaneously created in the desired folder and is available for editing in Brackets.

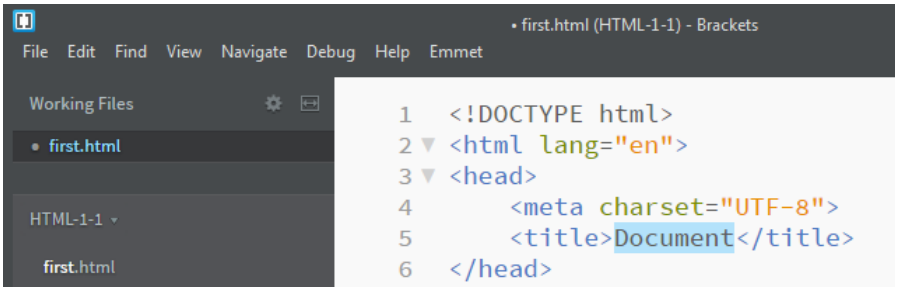


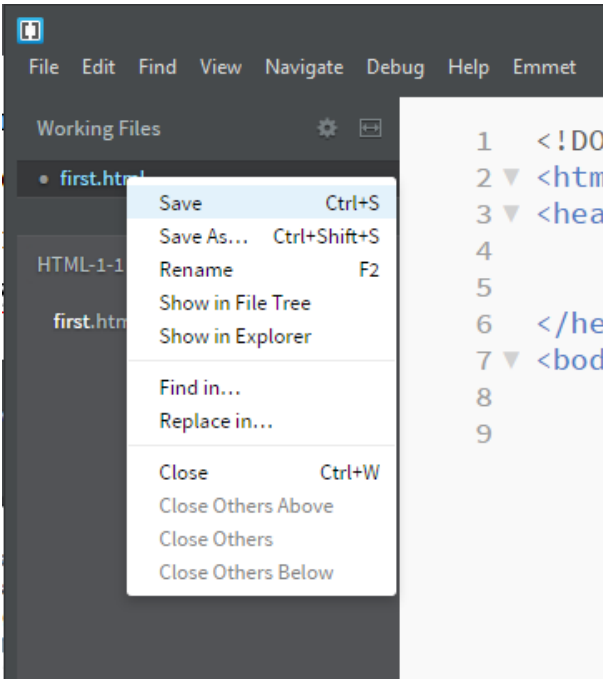
If you created an extra file, you can delete it or rename it with the right mouse button (the F2 key also helps). A good



option is 'Show in Explorer' — in this case the file will open in the folder in which it was created. At times this is a very useful option, because it happens that you create a file not in the directory you were going to, and then you cannot find it.

By the way, on the top line of the editor you will see the names of the file you are working with, and in parentheses — the folder in which it is located.





Another note — when you type the code, your file moves to the top of the left dark gray panel in the block named Working Files. And if you do not save the changes, a dot will appear next to it — it's a signal that you need to press Ctrl+ S (or File -> Save).

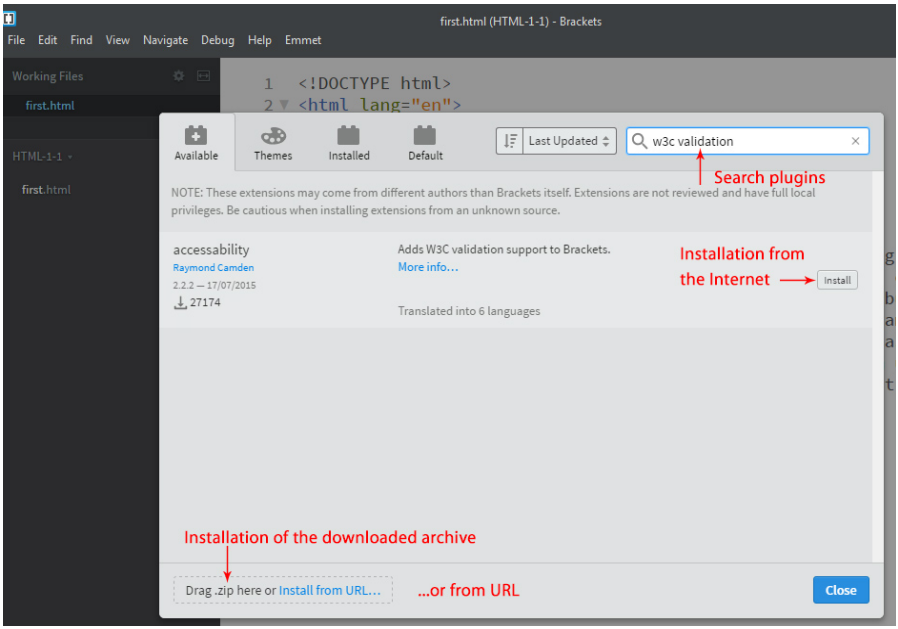
For those who are not friendly with the keys, and this is not good, the right click opens the Save option in the context menu, as well as others. Look, there is written Ctrl+S on the right — remember this key combination and always use it!

Brackets, like other programs, can sometimes «hang up», and when it closes, for example, through the Task Manager, all unsaved changes fall into oblivion. And you will have to start anew. And this is so sad!

Brackets Extensions

Believe me, it's very convenient to install several important extensions (add-ins) within a few minutes and use them in further practice.

Plugins, or Brackets extensions, are installed by clicking on the button in the form of the lego piece on the Brackets dark gray panel or through the menu File -> Extension Manager.



Plugins are downloaded from the Internet on the Available tab, and they can be installed immediately by clicking on the corresponding button. Alternatively, you can find a plugin in the repository on Github and by clicking on the 'Install from URL' button, enter the link address.

There are a lot of plugins, so you should use the search field. In this case, only those that are needed will remain from the large list.

If there is no Internet connection, and the plugins have been downloaded to a folder beforehand, then you can install them by simply dragging them to the button at the bottom left with the inscription 'Drag .zip here'.

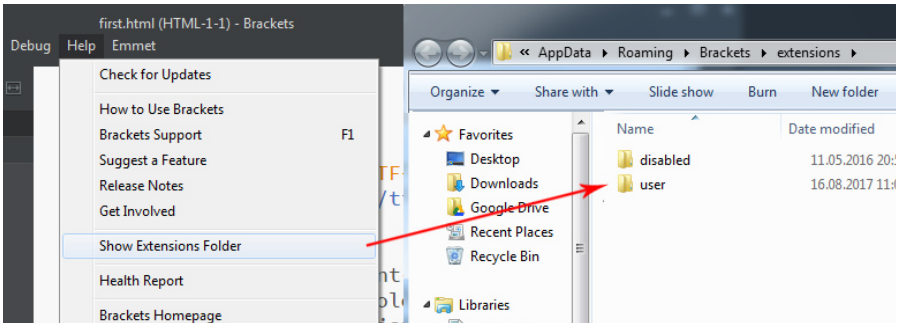
As for the recommended plug-ins — install the following set:

1. Emmet — https://github.com/emmetio/brackets-emmet. This plug-in will be needed to speed up the coding and writing css-styles.
2. Jsbeautifier — https://github.com/taichi/brackets-jsbeautifier or Beautify — https://github.com/brackets-beautify/brackets-beautify. The very name of the plugins says that any of them will nicely format the code.

All other plug-ins we will install as needed.

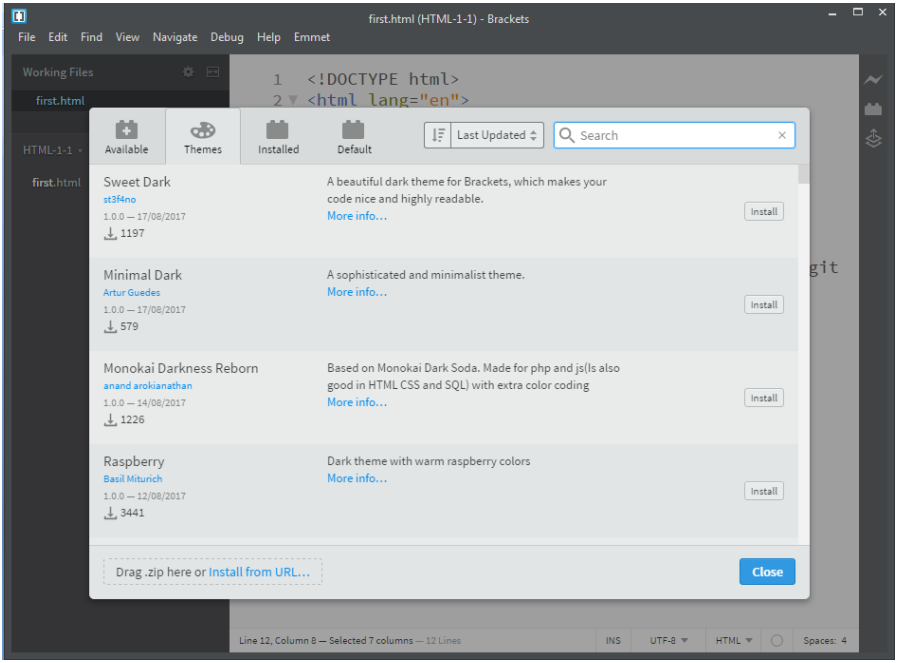
After a successful installation, you will receive a message that the installation was successful.

If for some reason you could not install the extension, then you can unzip the plug-in from the .zip-file into the folder with extensions. You can display it through Help -> Show Extensions Folder. The folder Roaming -> Brackets opens. It will contain the user folder, into which you need to unpack the archive. After that, **the editor must be reloaded**.

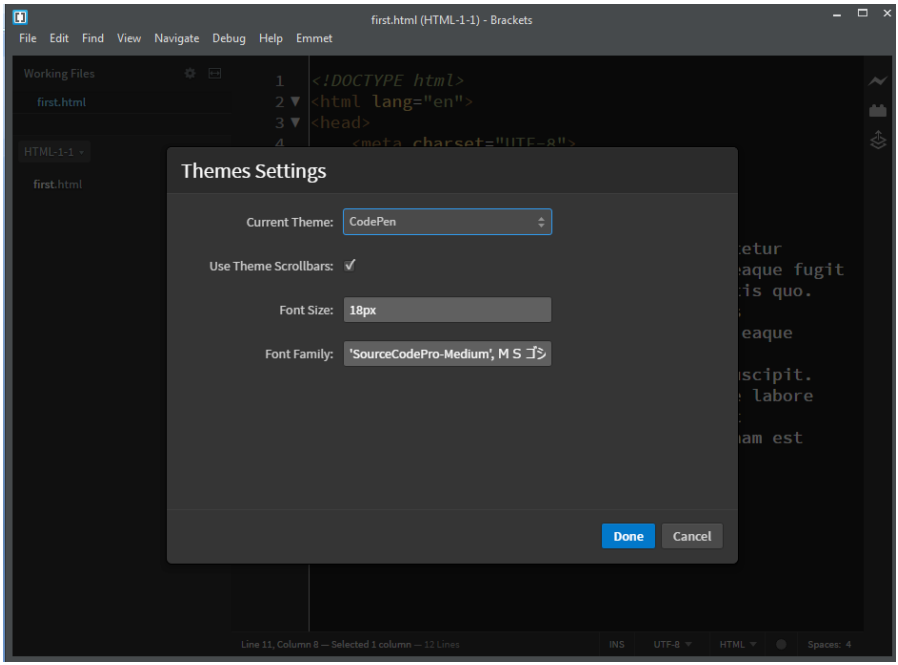


Themes for Brackets

They are loaded in the same way as extensions, but on the Themes tab. Choose by name or click on the «More info» button to see the appearance of the theme on Github.



To change the theme, go to the View -> Themes... menu and select the one you want. You can immediately see the changes. Stay on the one that is most acceptable to you.


first.html (HTML-1-1) - Brackets
File Edit Find View Navigate Debug Help Emmet
Working Files
first.html
HTML-1-1
first.html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
Themes Settings
Current Theme:
CodePen
Use Theme Scrollbars:
Font Size:
18px
Font Family:
'SourceCodePro-Medium', M S ゴシ
Done
Cancel
Line 11, Column 8 — Selected 1 column — 12 Lines
INS
UTF-8
HTML
Spaces: 4

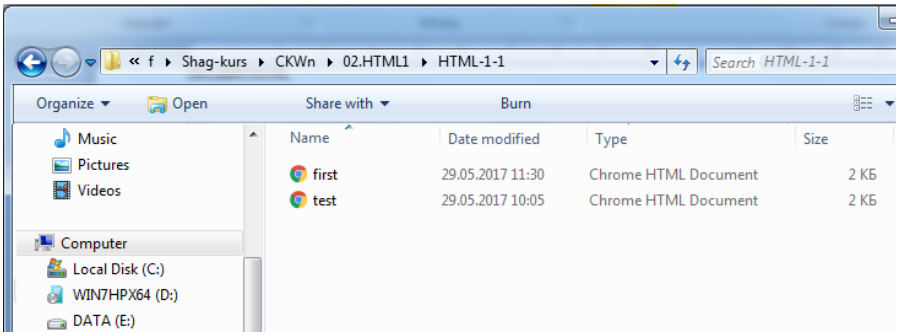
A few words about creating HTML files

As mentioned above, you need to create files with the extension .html or .htm to work with HTML.

In Brackets, you can create this file through File -> New or by pressing CTRL+N and saving it in the desired folder with the desired name and extension. I'll also remind you about the 2nd method — it's right click on the dark gray area on the left and selection of the New File option.

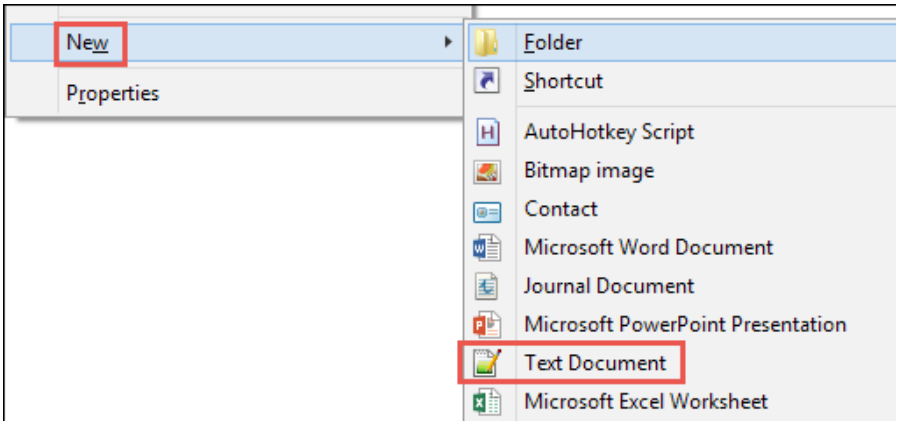
But for sure, you will want to create a file directly in the folder that you will work with using Windows Explorer. Here you can encounter one dirty trick.

The fact is that on Windows systems, since the 7th version it is usually accepted to hide file extensions. Therefore, the appearance of your folder will be something like this:

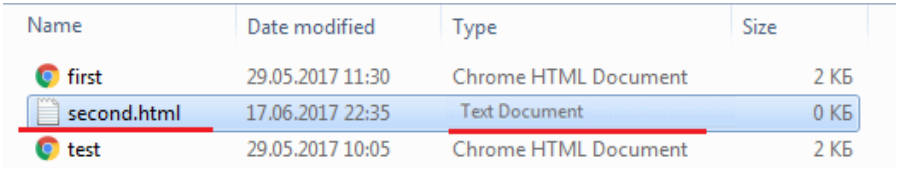


In the folder you can see the icons of the Google Chrome browser, the names first and test and the fact that they are Chrome HTML Document. But, notice that there are NO file extensions (.html)!

If you try to create a new document in this folder, you must create a plain text document, because in fact, html-files are just text documents but with their own characteristics — markup in the form of tags.

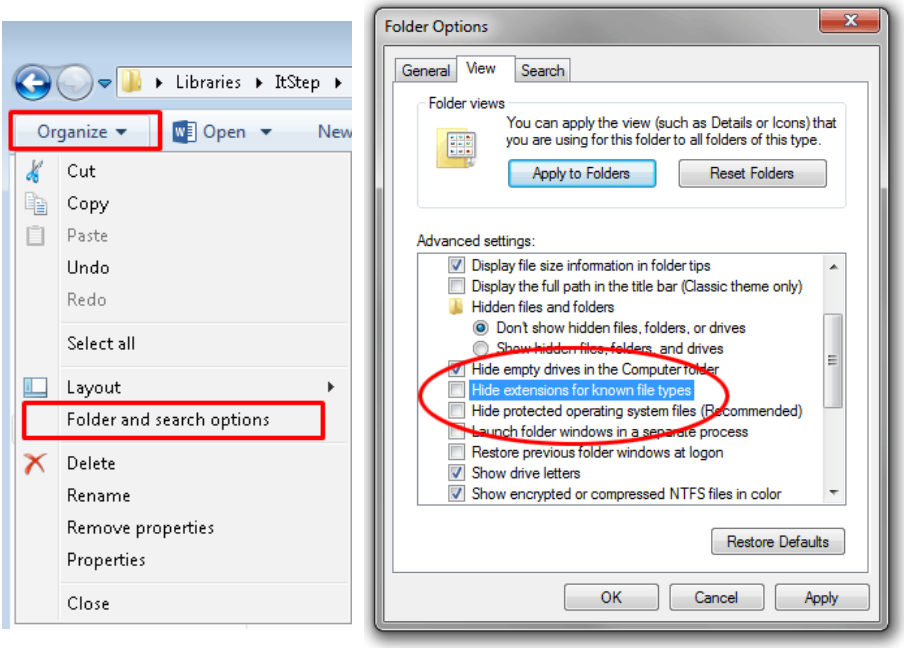


This document must be renamed. For example, call it second.html:

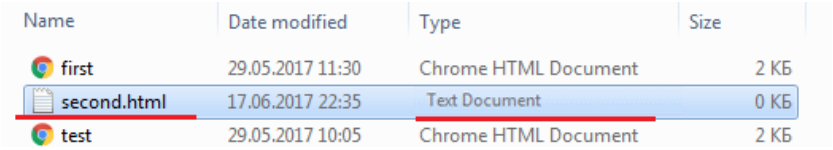


Note that visually the icon of the text document did NOT change and in the Type column there is a Text Document record, although the extension .html is present in the name of our file. If you double-click on second.html, you will most likely open the standard Windows Notepad.

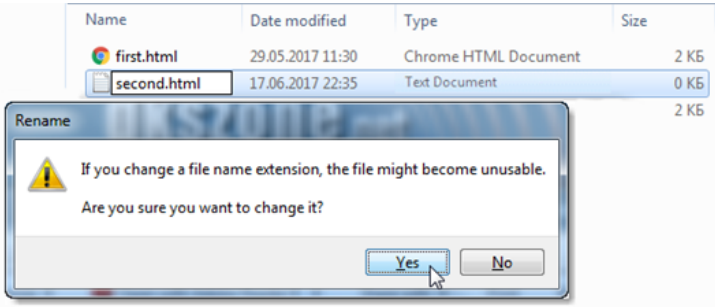
To change this, you need to select 'Folder and search options' in the 'Organize' menu in the Windows Explorer and deselect the 'Hide extensions for known file types' check box in the 'View' tab:



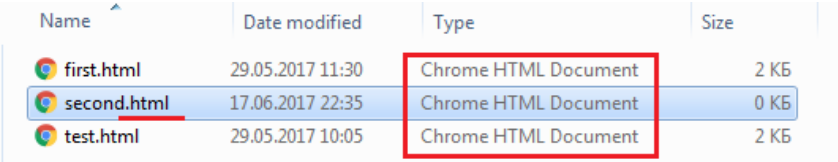
In this case, it becomes immediately apparent that the extension of our file is .txt, and .html is a part of the file name.



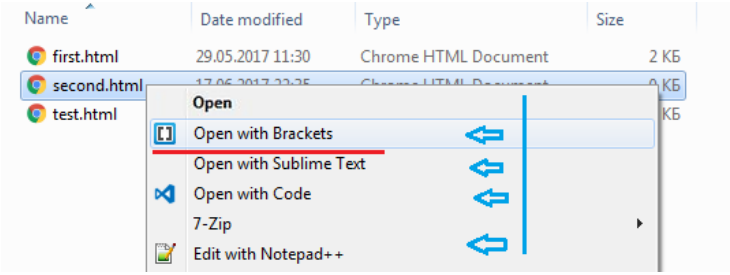
Now select the file, press F2 and remove the extension .txt.



Feel free to click the Yes button in the warning window and get the long-awaited html file.



Now make a right click and select the code editor. We will open the file using Brackets — 'Open file with Brackets' in the context menu.



The structure of the HTML file. DOCTYPE

Any html-file has a basic structure, which consists of html, head and body tags. But it always begins with a document type declaration — DOCTYPE.

We are considering the syntax of the latest HTML standard at the moment — this is HTML5. For it, the type of document is very simple:

```
<!DOCTYPE html>
```

The previous standards are HTML4.01 and XHTML 1.0, which existed in different versions: strict, transitional and framesets, so the DOCTYPE needed to be declared differently depending on the type of markup that was used in the html document.

Strict document type

Strict syntax of the language of the corresponding standard is used, and it is also possible to include all tags and attributes, except deprecated ones.

For HTML 4.01:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//
EN" "http://www.w3.org/TR/html4/strict.dtd">
```

For XHTML 1.0:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//
EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

The syntax of the language depends on the version used, but in both versions the following rules should be observed:

- It is highly undesirable to use tags such as: <applet>, <basefont>, , <dir>, <font>, <isindex>, <noframes>, <plaintext>, <s>, <strike>, <u >, <xmp>, since they belong to the deprecated ones. That is for this type of document, the use of the listed tags is unacceptable in terms of the W3C specification. Instead, you must specify the formatting using the css styles.
- In addition, you cannot add any text, images, and form elements directly to <body>. All these elements must be inside other block elements, for example, <p> or .
- The application of such attributes as target for links (<a> tag), as well as start (<ol> tag), type (<li>, <ol>, <ul>), etc. is deprecated. Now we will not go into detail on what are these tags since this will be the subject of a separate lesson. We'll talk about the attributes later in this lesson.
- You are also not allowed to use frames.

Transitional document type

In this case, the «soft» language syntax is used, and all tags and attributes, including deprecated ones, can also be used.

For HTML 4.01:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/
loose.dtd">
```

For XHTML 1.0:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
```

The purpose of such transitional doctypes is to gradually familiarize with the syntax of the language. It can help at the initial stage of learning the language, especially if you do it from old textbooks or online guides. In documents with transitional doctype, you can use the target attribute, which allows you to open a link in a new window. You can also use tags such as or <font>, which are so sweet to the heart of most beginning html coders. But in this DOCTYPE, frames are also not allowed.

Just want to mention about the transitional DOCTYPE — if you have formatting elements, which, frankly, are already outdated at the moment, such a doctype will create a valid document. But, if you seriously decide to make layout — forget about the or <font> tags — such css-properties as text-align: center or font-family, font-size and color will give you much more options for controlling the appearance of the document .

Frameset document type

In due time (the beginning of 2000th years) frames were very popular tool for loading 2, 3 or more html documents in one html-file. And at the moment there are many ways to create a beautiful html file, without resorting to such complexities. For a frame structure — frameset — a special DOCTYPE is used, which is similar in syntax to the transitional one.

For HTML 4.01:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">
```

For XHTML 1.0:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-frameset.dtd">
```

You may have a question: why do you need to know about all these types of documents, if now it is enough to use a simple <!DOCTYPE html>?

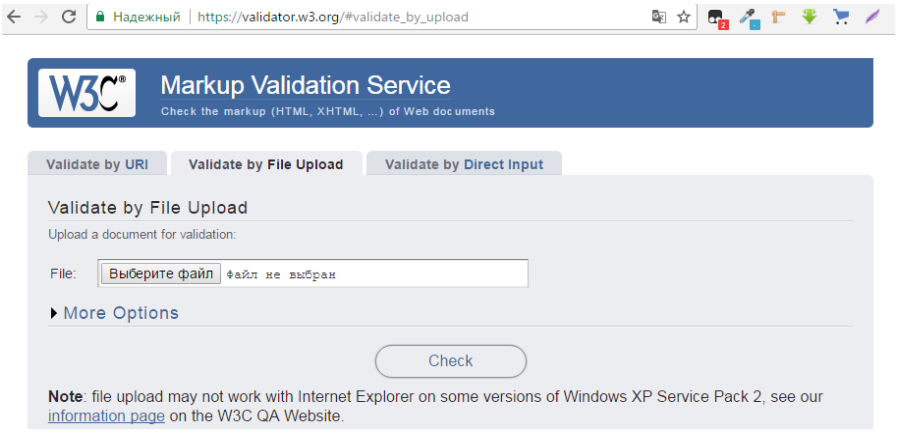
The fact is that the web still has very many sites created 5-10-15 years ago, foe which, perhaps, you will have to redesign (i.e. change the appearance) or just edit part of the code. And it is important to understand what can, and what cannot be left in the document, so that the code is considered valid, i.e. compliant to W3C standards.

Validation of html-documents

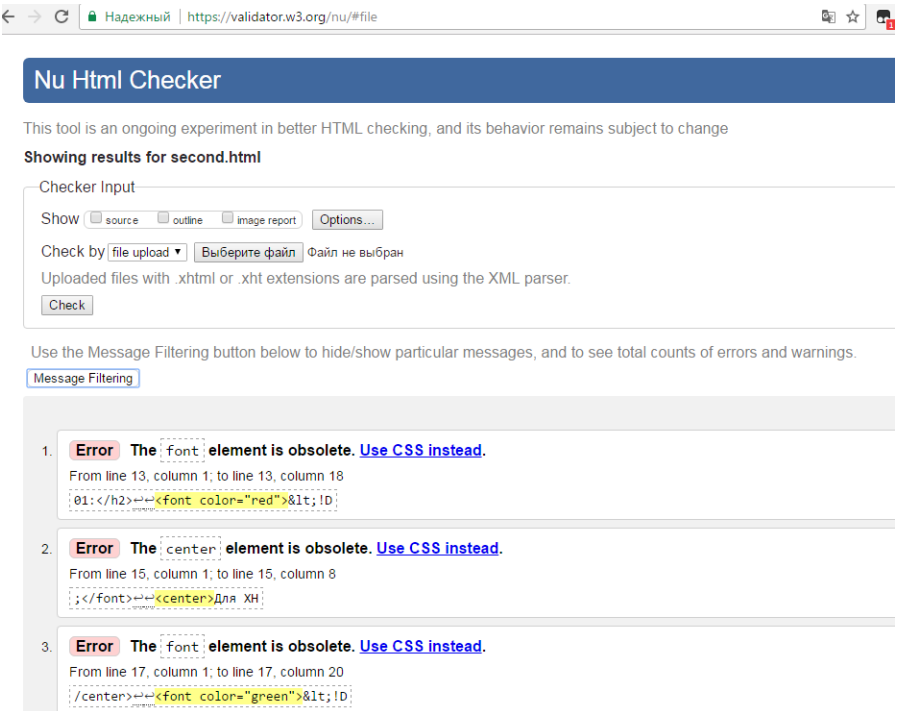
The validation of html-documents is the verification of individual html-documents and sites for the correctness of the code. Available via the links:

- https://validator.w3.org/
- https://html5.validator.nu/

At first you will need a validator that allows you to upload a file, and not to specify a link to the site on the Internet. Therefore, the link will be: https://validator.w3.org/#validate_by_upload.



Here's what the validator displayed when loading a document with <!DOCTYPE html> and <font> and tags inside:



As can be seen, both elements are classified as obsolete. Do you think it's worth using them in the modern standard?

Basic structure of html-document

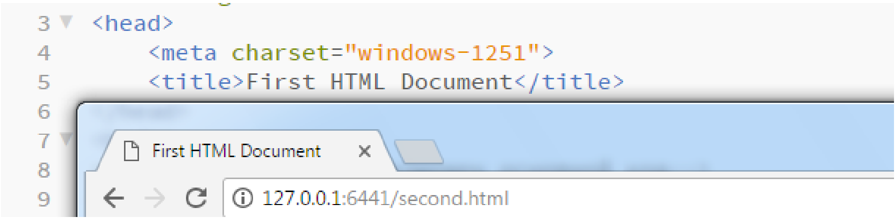
So, we dealt with DOCTYPE. Now let's consider which tags form the basis of the html-file.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>First HTML Document</title>
</head>
<body>
    <!--    The main code will be placed here-->
</body>
</html>
```

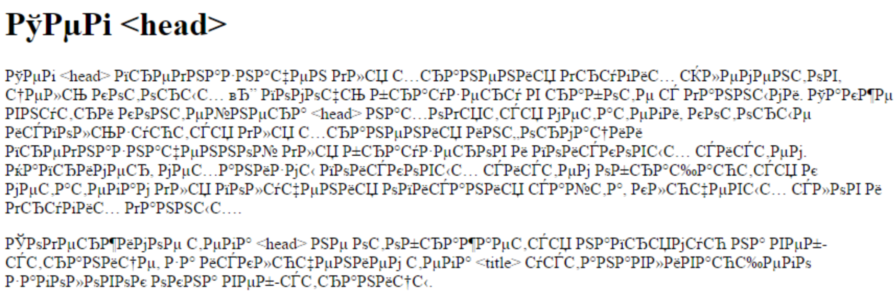
The root tag is <html>. All other tags are embedded in it, like nesting dolls. This tag is used to specify the lang attribute, which is responsible for the language on which this html page was created. In the example, lang=»en», because English is the most popular language in the world, and the text between the <title> tags is also in English.

If you are creating a page with Russian text, the value of this attribute must be «ru»: <htmllang=»ru»>

Further in the document structure there is a <head> tag, the contents of which are not displayed on the page. However, this is an important tag that performs «service» functions. First, it indicates the <title> of the document, which is displayed on the tab in the browser.



Secondly, it contains meta tags, which give information to search engines and not only. The <metacharset="UTF-8"> tag specifies the encoding of the document, which we'll talk about separately. Now I will only note that the wrong indication of the encoding will result in the text of your page being displayed as «mojibake», which is undesirable:



This appearance of the page resulted from the encoding specified in the form <meta charset="windows-1251"> whereas the document was saved in the utf-8 encoding.

- **Note:** *HTML5 simplified the writing of some tags, including <meta>. Earlier (HTML 4.01) it was necessary to specify the encoding in a somewhat longer way:*

```
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
```

And in XHTML, also add a closing slash at the end:

```
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8" />
```

So do not be surprised if you encounter such a syntax of the tag.

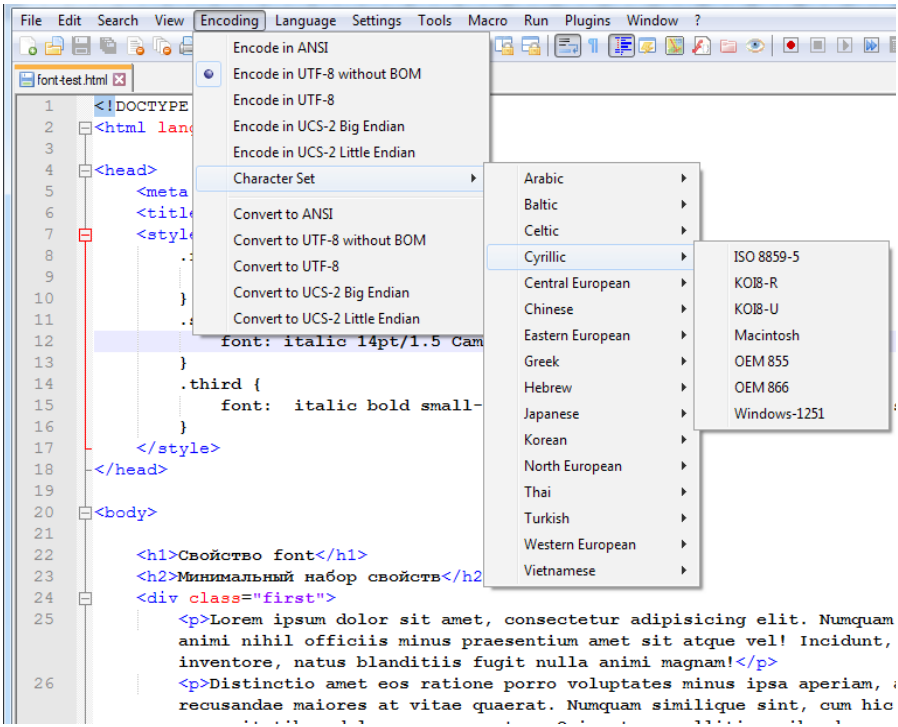
Document encoding

The fact is that Windows, by default, has the character encoding for files in the **ASCII system** (*American Standard Code for Information Interchange*), which includes Latin and Russian characters, numbers, punctuation marks, etc. In HTML it corresponds to the encoding windows-1251.

ASCII is a single-byte encoding that allows you to get a smaller file, but also inferior to the two-byte UTF-8 encoding. For example, it will be difficult for you to create a multilingual site using it, which should be displayed in Russian, English and, for example, Czech. Its capabilities will be not enough to display the symbols of all languages.

In this sense, the two-byte encoding UTF-8 is more universal, because contains character codes in a large number of languages. That is why it has become the most popular now and is now the de facto standard in the world of web development.

In addition to these encodings, there are also ISO-8859-5 and KOI-8-R for the Russian language, but they are so little used that it makes no sense to consider them in this course. If you installed Notepad++ on your computer, open it and in the Encoding -> Character Set -> Cyrillic menu, see the available options.

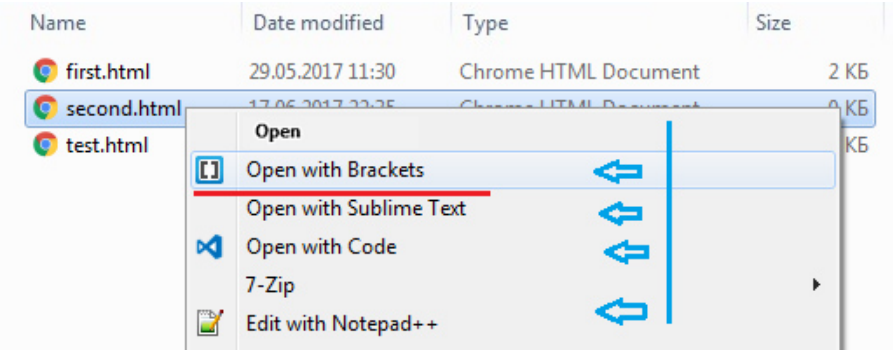


To view the encoding of the .txt document, which we considered as the initial one for creating html-files, you need to select ‘Save’ or ‘Save as’ in the ‘File’ menu of the Notepad program if the document has not been changed or has already been saved, and at the bottom of the settings window select the **UTF-8** encoding instead of **ANSI**.

This is the easiest way that is available to you in the Windows system. The Notepad program is installed there by default. But it has its negative sides, which are not noticeable in HTML, but are visible when creating and editing php files, for example for a CMS like Wordpress. The matter is that Notepad adds a **BOM** label (Byte Order Mark). Using it in site templates causes blank lines in the document to appear, so it should be removed from files destined for the web (.html, .css, .php, .js).

The program Notepad++ is best for this purpose, where the 'Encode in UTF-8 without BOM' option is provided in

the Encoding menu (see screenshot above). To do this, you need to edit the desired file with Notepad++ (Edit with Notepad++) — the option, available from the context menu of any file for the web.



Document body: <body> tag

The <body> tag is the place where you will put the main html code. It is its content that is the main content of the page and is displayed in the browser. All that will be between the opening <body> and the closing </ body> tag, must also be formatted as tags.

Comments in HTML

Comments in HTML do not appear on the page. All text written between tags

```
<!-- -->
```

will be hidden from the visitor of your site.

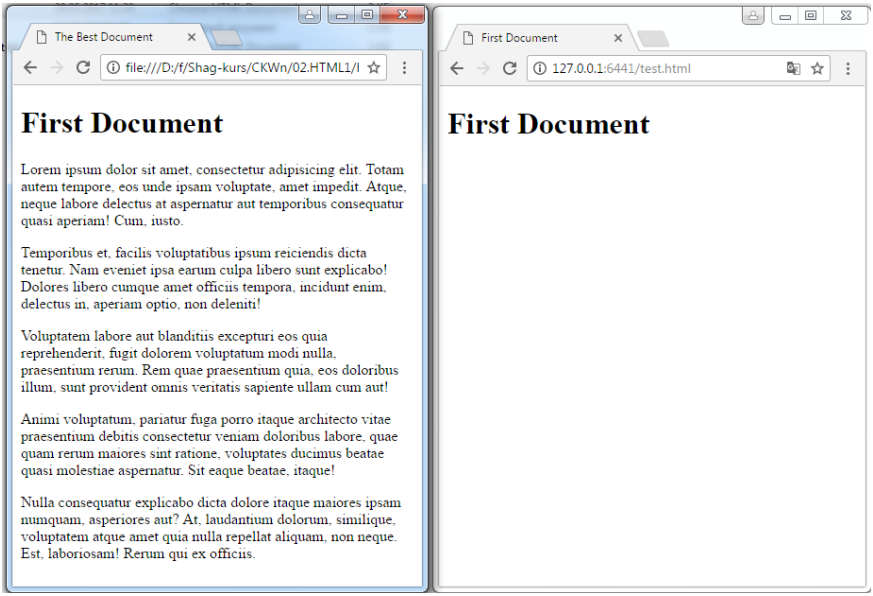
Comments are necessary to indicate, for example, the beginning and end of the formatting of a block:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <title>First Document</title>
</head>

<body>
    <h1>First Document</h1>
    <!--    start #wrap   -->
    <div class="wrap">
        <p>Lorem ipsum dolor sit amet, consectetur
        adipisicing elit. Totam autem tempore, eos unde ipsam
        voluptate, amet impedit. Atque, neque labore delectus
        at aspernatur aut temporibus consequatur quasi
        aperiam! Cum, iusto.</p>
        <p>Temporibus et, facilis voluptatibus ipsum
        reiciendis dicta tenetur. Nam eveniet ipsa earum
        culpa libero sunt explicabo! Dolores libero cumque
        amet officiis tempora, incidunt enim, delectus in,
        aperiam optio, non deleniti!</p>
    </div>
    <!--    end #wrap   -->
</body>

</html>
```

Or in order to hide this block:

```
<body>
    <h1>First Document</h1>
    <!--
    <div class="wrap">
        <p>Lorem ipsum dolor sit amet, consectetur
        adipisicing elit. Totam autem tempore, eos unde ipsam
        voluptate, amet impedit. Atque, neque labore delectus
        at aspernatur aut temporibus consequatur quasi
        aperiam! Cum, iusto.</p>
        <p>Temporibus et, facilis voluptatibus ipsum
        reiciendis dicta tenetur. Nam eveniet ipsa earum
        culpa libero sunt explicabo! Dolores libero cumque
        amet officiis tempora, incidunt enim, delectus in,
        aperiam optio, non deleniti!</p>
    </div>
    -->
</body>
```

The screenshots show that all the text placed in the <!-- --> tags becomes light gray in Brackets — as if it is inactive. If you compare the 2 options for displaying this page with a different arrangement of comments, then we see the following in the browser:

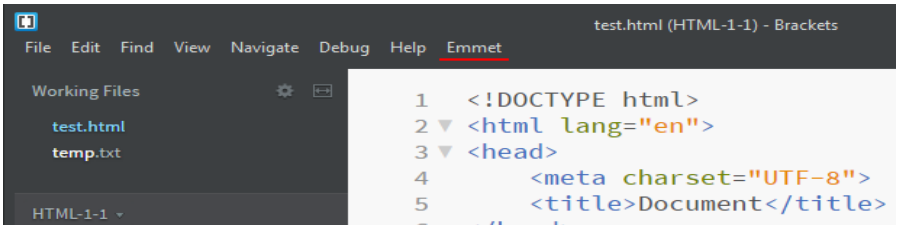


Where the comments get into the comments of the beginning and the end of the block, we do not see only these words. And in the example with the commenting of the block itself, all the text after the header disappeared.

It should be noted that in Brackets (as well as in other code editors, however) there are shortcuts for adding comments — this is CTRL+/. If you want to comment out one line, just place the cursor in any place and press CTRL+/. Comments are automatically added at the beginning and end of this line. To comment on several paragraphs, you must first select them, and then press CTRL+/.

Using the Emmet plug-in to create a document structure

If you installed the set of Brackets plugins recommended at the beginning of this lesson, then now is the time to take advantage of the Emmet plug-in. It is very simply to check that it is installed in your editor: look at the Brackets menu bar, if at the end you see the Emmet item then it means everything is in order.



If you do not see it, I highly recommend to fix this situation, because the code of the basic structure is typed using Emmet super-simple: ! and the Tab key.

Heading and paragraph tags

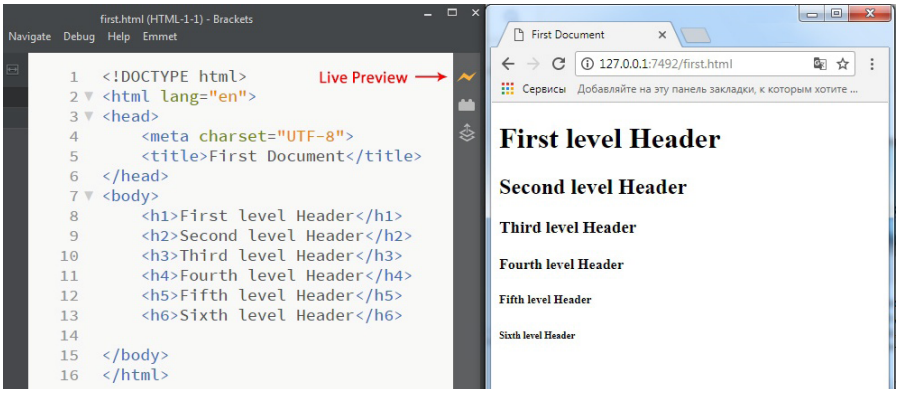
Headings are a mandatory part of the html document. There are even 2 new HTML5 tags that will not be validated without headings — <article> and <section>. But we will consider them later.

HTML provides 6 levels of headings, which differ from each other in the font size. The headings are as follows:

```
<h1>First level heading</h1>
<h2>Second level heading</h2>
<h3>Third level heading</h3>
<h4>Fourth level heading</h4>
<h5>Fifth level heading</h5>
<h6>Sixth level heading</h6>
```

The letter **h** in the tag indicates that this is the heading, and the figure indicates a level of the heading. The larger the figure is, the less important the heading is and the smaller font size it has.

Let's see how it looks in the code editor and in the browser:



The difference between the headings of different levels, in my opinion, is obvious. <h1> is the largest and most noticeable, and <h6> is almost unreadable. Actually, the importance of these headings for the search engines is also different. Typically, there should be one or two headings of type h1 on the page — this is the name of the company and the name of the page. It is also desirable that the text in h1, which defines the name of the page, coincides with the contents of the <title> tag in the <head> block.

You can use all other headings at your discretion, not necessarily in ascending order of their level, but it is still desirable that h1 be followed by the heading h2, because it is also important for ranking your page.

- **Note** *1: in the screenshot, the arrow shows a button in the form of lightning in the upper right corner of*

the Brackets. This is a Live Preview in the Google Chrome browser, «embedded» into the code editor. Pressing this button when editing the html-document allows you to all the changes that occur in the editor see in real time. To see this, it makes sense to place the editor window next to the browser window, as in the screenshot above.

When the Live Preview is enabled, in the browser the blue frame highlights the element on which the cursor is currently located (in the screenshot it is h2). If you move the cursor higher or lower, for example with the ↑ or ↓ keys, you will see how the frame moves from one element to another.

- **Note** *2: If you have to create blocks with similar or identical content in Brackets, it makes sense to use the CTRL+D keyboard shortcut. If the cursor is anywhere in the line, the whole line will be duplicated. If you select a part of words or several lines, then all the selected content will be duplicated. Believe me, it can be very convenient.*

As for headings, you should be aware that they are all block elements that are bolded by default in browsers and have indentations before and after the text that composes them.

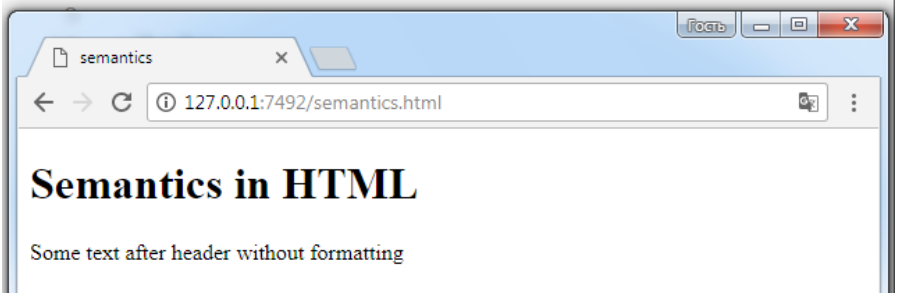
You still need to understand that you cannot fill the entire page with headings only — this will be incorrect from the point of view of semantics of the html-document. And the search engines are also unlikely to like it. Think about it — can a book or an article in a magazine or in a newspaper consist of only headlines? Naturally, it cannot. The heading must be followed by the main text, which in the print press is usually broken into paragraphs. Let's consider how to create them in html.

What are block elements?

Block elements are such HTML elements that by default occupy all available space within the browser or parent element, even if their content is very small. Thus, even with 2-3 words inside the heading, it will occupy the entire space to the right border of the browser, and the text placed next to the closing tag will be moved to the next line.

The screenshot in the example shows that the h1 heading, highlighted in the Brackets in Live Preview mode, is circled in a blue frame and takes up the whole place from left to right in the browser. And the text after it is moved to the next line.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>semantics</title>
6  </head>
7  <body>
8      <h1>Semantics in HTML</h1>Some text after
       header without formatting
```

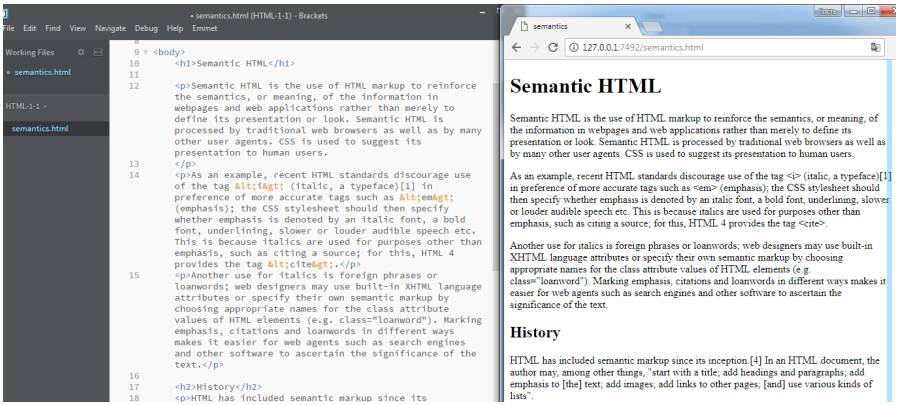


Paragraphs in HTML

As in printed publications, paragraphs are the basic elements for formatting the text that makes up the article. The   tag is intended for the creation of paragraphs.

You will find a variant of dividing the text into paragraphs in the file semantic.html *(example files are attached to the PDF file of this lesson)*, in which the text is a fragment of the article from Wikipedia.

The screenshot shows that the text, placed in paragraphs, is displayed in the browser in the usual form (not bold, as in the headings), but each paragraph has indents above and below. This visually separates one semantic block from the other and helps to more easily perceive the essence of the text.



You can watch videos on basic structure of an html document at https://www.youtube.com/watch?v=TAt1e7e90hE and heading and paragraph tags at https://www.youtube.com/watch?v=Iuf2uJtfd2U.

A few words about the attributes

For any tag, you can specify attributes — additional parameters in the form of pairs attribute=”value”, which to some extent distinguish it among others similar. Attributes are written only in the opening tag and are separated from the tag name and from each other by spaces. In general, it looks like this:

```
< element attribute1='value' attribute2='value'
attribute3='value'>Element text</element>
```

Example:

```
<h2 title='Article about the benefits of vitamins for
children' id='article1' class='articleheader'>
Vitamins for Children</h2>
```

In this example, the universal attributes title, id and class are used for the header of the 2nd level, i.e. such attributes that can be added for any element. Also universal attributes include tabindex, data attributes and some others rarely used, for example, contenteditable, hidden or contextmenu.

But there are also attributes that are specific only for certain tags. For example, to embed an image, you need a tag with the attributes src and alt, which are typical only for it (although src is needed for tags such as <script> and <iframe>):

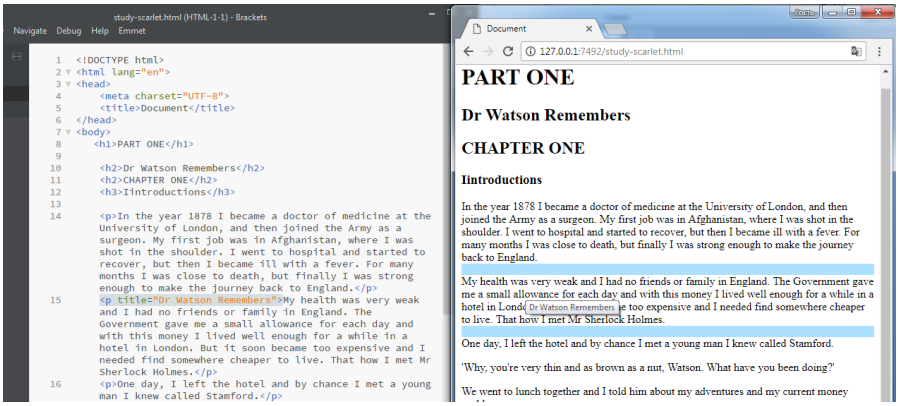
```
<img src='images/photo.jpg' alt='Image'>
```

When adding attributes, you must follow a few simple rules:

1. Attributes are separated from each other by spaces.
2. Attribute values are written in double or single quotes.
3. If you need to use quotes or an apostrophe inside the value of an attribute, you must combine both types of quotes taking into account their nesting:

```
<p title='Paragraph from Arthur Conan Doyle's book
"A Study In Scarlet"'>
```

Here, the value of the title attribute is in single quotation marks, and double ones are used inside for book name. This attribute adds a tooltip to the item that appears when you hover over an item in the browser.

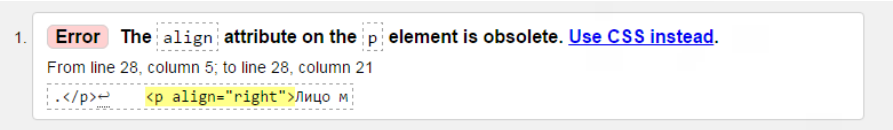


4. If the value of the attribute can be represented as a logical value true or false, then it can be written in several variants that are equivalent:

- **Note:** *All browsers are configured for the most correct display of tags and their attributes, so they will include the attribute even if there are values in it like true or 1, which are in fact unacceptable. It is better to avoid such variants since they contradict the HTML5 specification.*

Now let's talk about obsolete attributes. For example, the align attribute allows you to align the text in a paragraph, heading, or other block element along the left or right edge, to justify or to center it (align="left", or align="right", or align="justify", or align="center"). You will certainly find a bunch of tutorials on the Internet that recommend using this attribute for text alignment.

Only here with the HTML5 standard, this attribute is incompatible. And the validator will return the following message: 'The align attribute on the p element is obsolete. Use CSS instead.'



So it's not worth using obsolete attributes. HTML is a very popular markup language. It is constantly improving and developing. With each new standard, approaches to markup are revised. Some of the tags or attributes go to the obsolete ones and are not recommended for use. We will replace them with css-styles, as recommended by the validator. And if you did, check your document for validity.

What is Lorem Ipsum?

The fact is that in practice you will not always have to format the finished text provided by the customer. Unfortunately, customers quite often send it too late. In addition, you can create site templates, the real content of which at can only be guessed the design stage. Therefore, you will need a template text that begins with the words Lorem ipsum. It is also called 'dummy text'.

- **Quotation from Wikipedia:** *In publishing and graphic design, lorem ipsum is a filler text or greeking commonly used to demonstrate the textual elements of a graphic document or visual presentation. Replacing meaningful content with placeholder text allows designers to design the form of the content before the content itself has been produced.[1]*

Thus, this is a text that helps fill the site with content, but it has no sense. It is very convenient, among other things, at the training stage. You can fill with Lorem ipsum any test items (we already know about headings and paragraphs), without thinking about the amount and content of the text.

With the appropriate query, Google provides access to the bulk of Lorem ipsum generators:

- http://generator.lorem-ipsum.info/
- http://lorem-ipsum.perbang.dk/
- http://www.blindtextgenerator.com/lorem-ipsum.

But we have a much more convenient tool for creating such a text, and immediately "wrapped" in tags. And this ... again the Emmet plugin.

In order to get a paragraph with a dummy text, immediately after the opening tag <body> or after another tag already existing in it, type the abbreviation

```
p>lorem
```

and press the Tab key.

And voila, you have a paragraph with the text filler:

```
<p>
    Lorem ipsum dolor sit amet, consectetur
    adipisicing elit. Labore illum rerum facilis
    mollitia, explicabo cupiditate, eius fugit ea nemo
    saepe ex veritatis aliquid consequatur ratione quas
    asperiores minus dolorum odio.
</p>
```

By default, Emmet will add 30 words of text. But you can diversify the number of words, adding the correct number after lorem:

```
p>lorem10
```

We get:

```
<p>
    Lorem ipsum dolor sit amet, consectetur
    adipisicing elit. Necessitatibus, possimus!
</p>
```

It is not necessary to use lorem only with the p tag. You can apply it to a header of any level:

```
h1>lorem5
<h1>Lorem ipsum dolor sit amet.</h1>
h2>lorem3
<h2>Lorem ipsum dolor.</h2>
```

And so on…

- **Note:** *Emmet plugin uses abbreviations — character sequences, which allow you to display tags, attributes and text in a certain sequence and with certain levels of nesting.*

More on abbreviations:

- http://webdesign-master.ru/blog/html-css/2.html
- http://html-plus.in.ua/formatirovanie-teksta-s-po-moshhyu-emmet/
- http://ts-soft.ru/blog/emmet
- http://webtoks.ru/web/vvedeny-emmet/.

It is important to understand that there should not be any spaces inside the abbreviation (the exception is the text inside curly braces). And at the very end it is necessary to press the Tab key to expand the abbreviation and get the text formatted according to html rules. Again, after the abbreviation text, there should not be a space, and the cursor should be at the end, and not somewhere in the middle of the typed text.

Let's look at a slightly more complicated abbreviation. This time with a partially meaningful text:

```
h1{First heading}+h2{Second heading}+p*3>lorem20
```

We obtain the following structure:

```
<h1> Firstheading </h1>
<h2> Second heading </h2>
<p>
    Lorem ipsum dolor sit amet, consectetur
    adipisicing elit. Placeat corporis quasi
    perspiciatis? Aperiam eveniet dolorem culpa
    distinctio, rem quos adipisci.
</p>
<p>
    Debitis, dolorem! Sunt autem veritatis magnam!
    Ipsa, dolorem harum laborum praesentium,
    quas unde ab, alias saepe ullam similique nulla
    beatae.
</p>
<p>
    Quis soluta saepe incidunt voluptas consequuntur
    iste repellat, quasi quos provident, nostrum, a.
    Repellendus aspernatur, veritatis ea cum aliquid
    architecto!
</p>
```

If you decipher the abbreviation, you can see the following: h1 with the text 'Firstheading' is placed next to the h2 heading with the text 'Secondheading', and then there are 3 consecutive paragraphs with a dummy text of 20 words each. And the text in the paragraphs is different! Note that the text placed in curly braces was displayed exactly as it was typed.

Convenient, isn't it?

An abbreviation with a dummy text will look somewhat more difficult in headings and paragraphs. In it, you'll have to use the ^ character: exit to the parent element, that is body.

```
h1>lorem4^h2>lorem5^p*2>lorem15^h3>lorem4^p*3>lorem10
```

It looks frightening, but when you press the Tab key at the end it becomes quite readable:

```
<h1>Lorem ipsum dolor sit.</h1>
<h2>Lorem ipsum dolor sit amet.</h2>
<p> Lorem ipsum dolor sit amet, consectetur
    adipisicing elit. Recusandae molestias, ipsa non
    asperiores animi iste. </p>
<p> Eos eligendi aspernatur dolore voluptate natus,
    magnam, dolores, cupiditate nihil fugit asperiores
    doloribus ipsum accusamus. </p>

<h3> Lorem ipsum dolor sit. </h3>
<p> Lorem ipsum dolor sit amet, consectetur
    adipisicing elit. Porro, corporis. </p>
<p> Voluptates dicta eius vitae ipsam. Veniam rem
    consequatur, illum rerum. </p>
<p> Quis voluptatem, veniam recusandae, hic totam
    quisquam tempora itaque necessitatibus. </p>
```

Let's consider in detail what is written in this abbreviation:

```
h1>lorem4^h2>lorem5^p*2>lorem15^h3>lorem4^p*3>lorem10
```

- **h1>lorem4** — h1 heading with a filler text of 4 words;
- ^ — return from h1 the heading text to body;
- **h2>lorem5** — h2 heading with a filler text of 5 words;
- ^ — return from the h2 heading text to body;
- **p*2>lorem15** — 2 paragraphs with Lorem ipsum of 15 words;
- ^ — return from paragraph text to body;
- **h3>lorem4** — h3 heading with a filler text of 4 words;
- ^ — return from the h2 heading text to body;
- **p*3>lorem10** — 3 paragraphs with Lorem ipsum of 10 words.

This will look like the document in the browser.

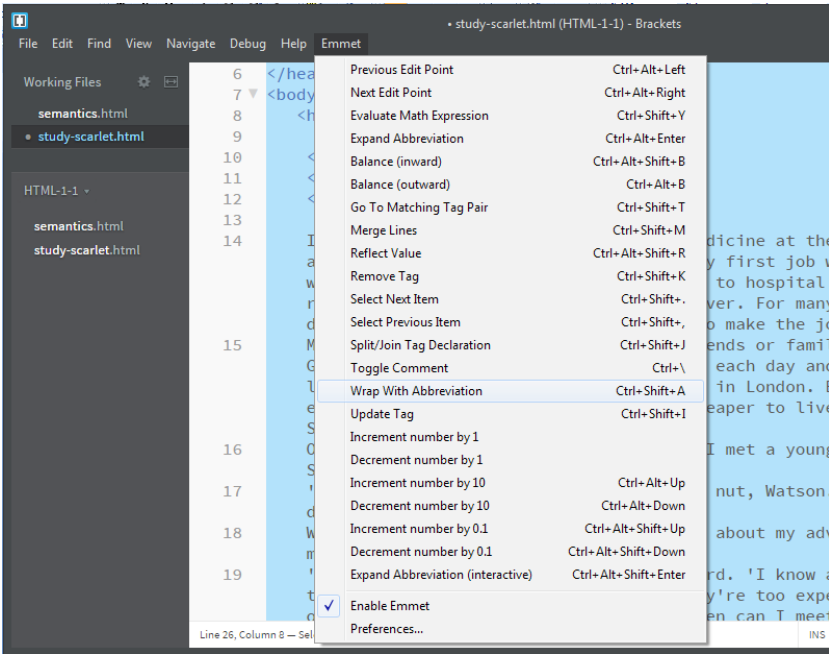
The figure below shows an approximate scheme of the abbreviation work with an emphasis on the symbol ^.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <title>Lorem ipsum Text</title>
</head>

<body>

<!--    h1>lorem4^h2>lorem5^p*2>lorem15^h3>lorem4^p*3>lorem10-->

    <h1>Lorem ipsum dolor sit.</h1>
    <h2>Lorem ipsum dolor sit amet.</h2>
    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Aperiam ex et sapiente
    suscipit nulla nihil.</p>
    <p>Sed eius ratione rem sit quas maiores praesentium explicabo assumenda hic,
    delectus animi, qui inventore?</p>
    <h3>Lorem ipsum dolor sit.</h3>
    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Optio, eaque.</p>
    <p>Reiciendis ea iure officiis, ullam debitis repellat provident nihil commodi.</p>
    <p>Distinctio veniam consectetur aspernatur tenetur facere. Totam quo enim, numquam.
    </p>

                    body - parent element
</body>

</html>
```

Try to expand the abbreviation yourself in the file loremipsum-emmet.html *(example files are attached to the PDF file of this lesson)*.

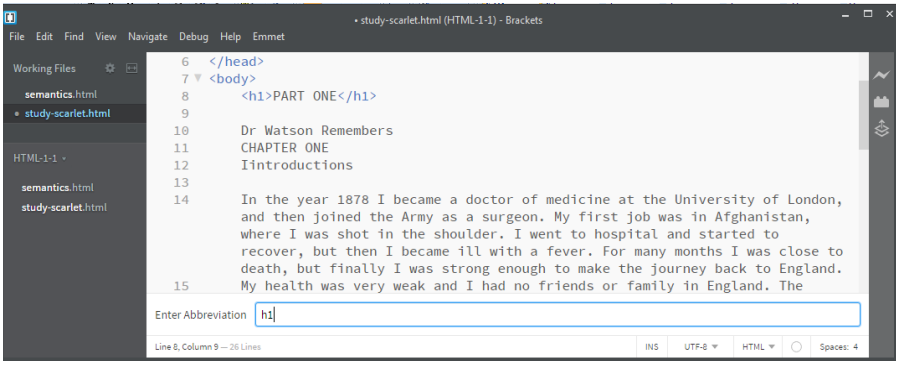
Wraps of Emmet abbreviations

There is another great feature in the Emmet plugin. It is called Wrap with Abbriviation (CTRL+SHIFT+A). You can find the corresponding item in the Emmet menu in the Brackets, but I still recommend that you remember the keyboard shortcut.

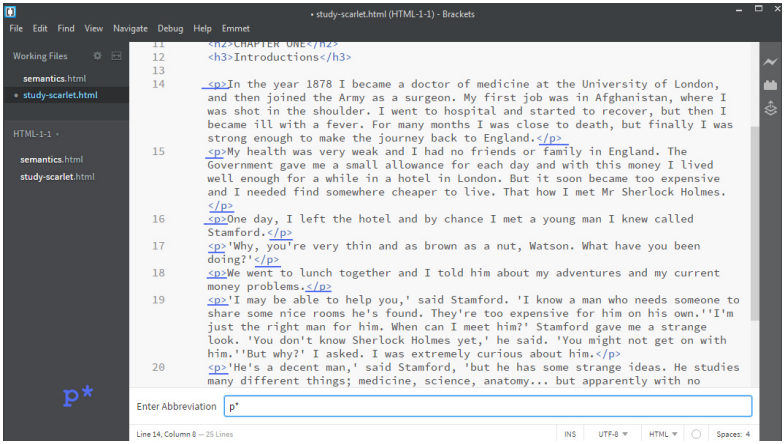


It is very simple to use wraps — select the desired text, press CTRL+SHIFT+A, and enter a tag or Emmet abbreviation. Look at the result and press Enter, otherwise the abbreviation will not apply.

For example, this would look like wrapping text in a header of the first level.



If you select all the text, divided into blocks using the Enter key, you can very quickly turn it into a number of paragraphs:



The abbreviation will be simple: p*.

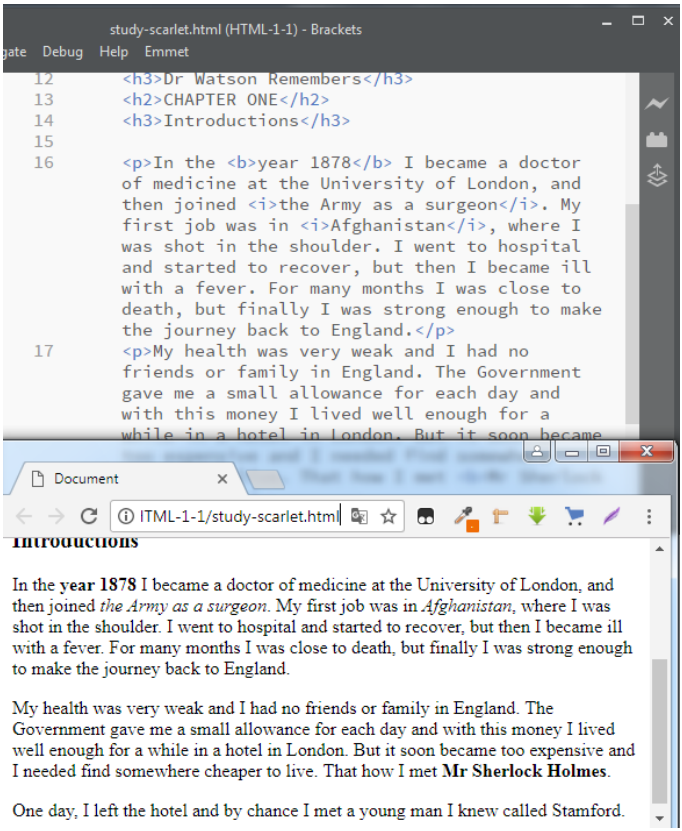
It's also easy to wrap all the text in the html document structure tags. It is enough to select all the text (CTRL+A), call the Enter Abbreviation field with CTRL+SHIFT+A and enter the exclamation mark:

- **Note:** *Be careful when entering abbreviations; Emmet will also create a tag that you entered, but which is not in the HTML specification.*

You can try Emmet wraps yourself using the a-study-in-scarlet-conan-doyle.txt *(example files are attached to the PDF file of this lesson)* file that is in the lesson folder. It uses the text from the website english-e-books.

Nested tags

To highlight a portion of text in bold or italic, you can use tags such as <b> or <i>:



```
<p> In the <b>year 1878</b> I became a doctor of
    medicine at the University of London, and then
    joined <i>the Army as a surgeon</i>. My first job
    was in <i>Afghanistan</i>, where I was shot in
    the shoulder.
    I went to hospital and started to recover, but
    then I became ill with a fever. For many months
    I was close to death, but finally I was strong
    enough to make the journey back to England.
</p>
```

The <b> tag is derived from the word bold, and <i> is from the word italics, respectively, the use of these tags leads to the highlighting a text between them using bold or italic style. These elements belong to the group of physical formatting tags.

Unlike block tags of headings and paragraphs, these tags are lowercase (they are also called inline).

Therefore, such tags are used to wrap not the whole sentence or block of text, but some part of it, for example, 2-3-4 words, as in the example in the screenshot.

Two more tags work in the same way: and , i.e. highlights the text in bold, and — in italics, but these 2 elements are part of the group of logical formatting tags. In the context of page layout, they act as an 'attention enhancer', i.e. are designed to show (to a greater extent, search engines than visitors) that the text highlighted in them is important to the user.

But both physical tags and logical formatting tags are inside the block elements (in the example this is a paragraph — <p>), i.e. they are nested or child relative to

the paragraph. A paragraph with respect to nested tags is the parent element.

For nested tags, it is important to observe the nesting rule: the tag that is opened first is to be closed last:

```
<b><i>Afghanistan</i></b>
```

```
medicine at the University of London, and then
joined <i>the Army as a surgeon</i>. My first
job was in <b><i>Afghanistan</i></b>, where I
was shot in the shoulder. I went to hospital and
started to recover, but then I became ill with a
```

Nested tags can be not only inline. And the level of nesting can be quite deep. But this rule still remains the same — the tag that is opened the very first should be closed the very last.

Div and span tags

As you know, you cannot build a full page from headings and paragraphs. Therefore, in the specification there are many more tags that we need to get acquainted with. And one of the most used elements are div-s. These are block elements that, unlike paragraphs, do not have an indentation from above and below. These are the elements which build the internal structure of the site.

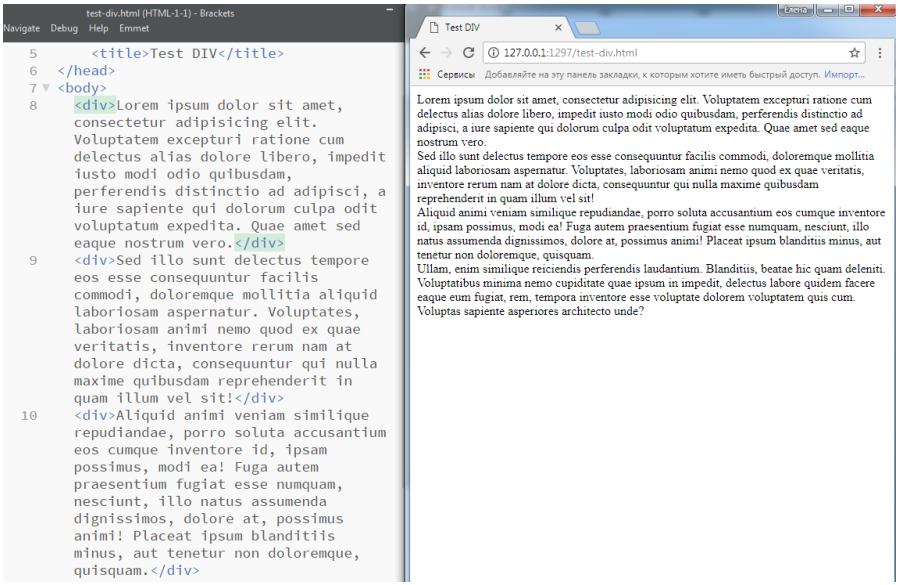
Their code is very simple:

```
<div>text</div>
```

Let's use a simple Emmet abbreviation:

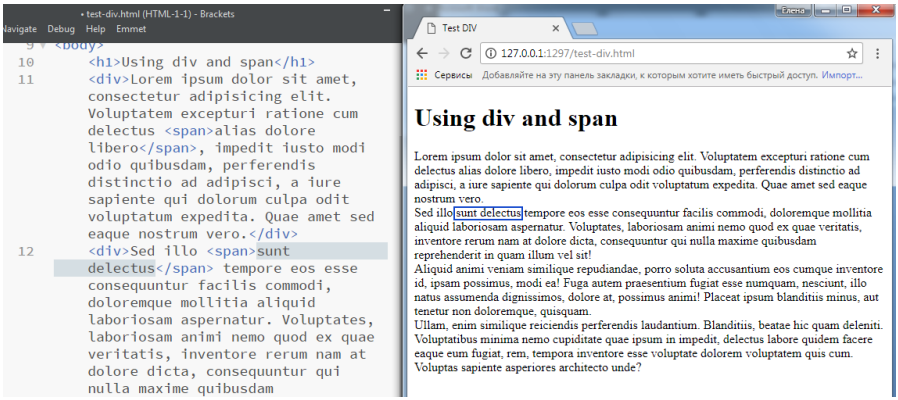
```
div*4>lorem40
```

We get the following file:



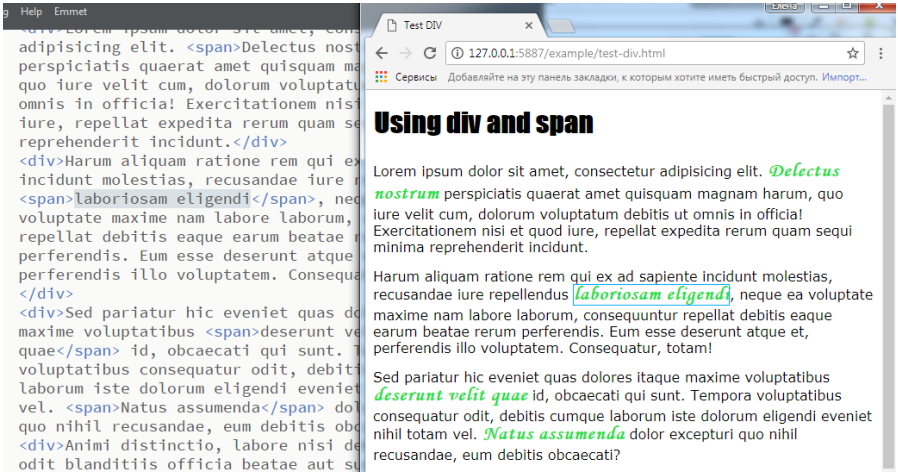
Visually, the tag does not look so good: a lot of text without formatting. But that's why it's good — for it you can specify the css-formatting that is necessary for this particular situation. And usually the class attribute is set for div-s, which allows you to diversify the appearance of these elements. But more on this later, when we consider the css styles.

Also they very often use the tags   inside the text. These are inline tags that are designed to combine a small amount of text with the purpose of specifying general formatting for it. The class attribute is almost mandatory for the span tag in this case. It allows you to diversify the formatting for various using the css rules.



The screenshot above shows that the selected text is in the tags. But visually it does not differ from the text before and after it.

But if you uncomment the code in the style tags in the file test-div-span.html *(example files are attached to the PDF file of this lesson)*, then the appearance of the file will immediately change in the absence of changes to the ht-ml-markup:



The example shows the same formatting for all in the text, but when you add the class attribute, you can vary it.

- **Note:** *To add or remove comment in Brackets, press CTRL+/. If you need to comment/uncomment 1 line, it's enough to have a cursor in it. To comment on a block of text, select it and press CTRL+/.*

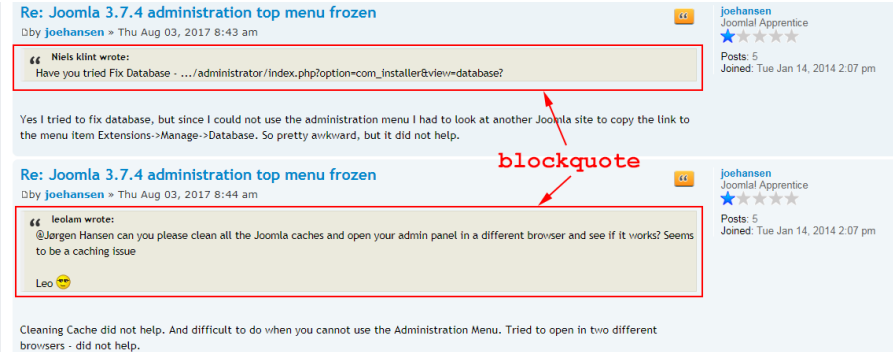
Blockquote tag

Let's consider one more block element, the blockquote tag, which is designed to format someone's statements — from famous people to the directors of companies for which the site is created.

```
<blockquote>Block quote</blockquote>
```

It should be noted that on regular sites this tag is used infrequently, but it is very popular in various forums.

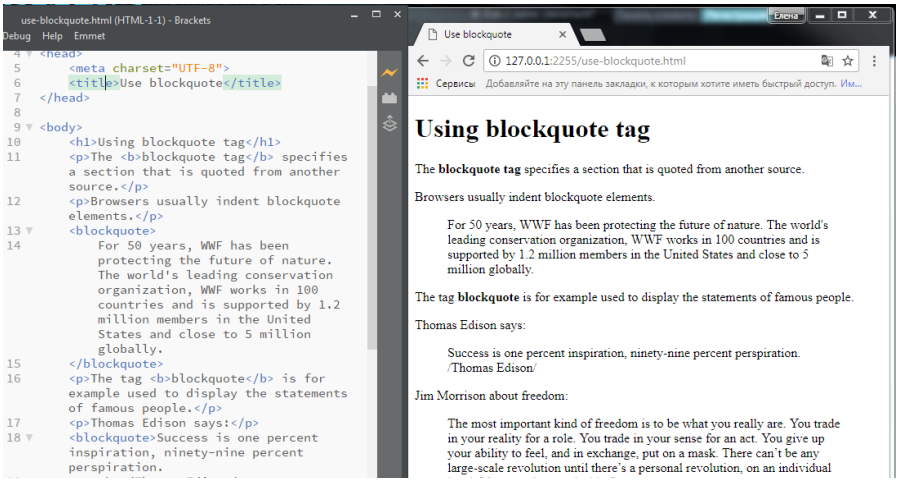
In the screenshot below is an excerpt from the forum about working with CMS Joomla, where the words of one of the users are quoted, and for this purpose the blockquote tag with the css-formatting different from the rest of the text was used.



By the way, the Emmet abbreviation for creating this tag is very simple:

```
bq
```

By default, blockquote has 40px padding to the right and left, as well as indentation from the top and bottom, as in paragraphs.



See the example in the file use-blockquote.html *(example files are attached to the PDF file of this lesson)*.

Single Tags

In the blockquote example, a tag
 was used inside the quote. It is designed to move the text following it to the next line and is inline, i.e. is usually placed in the text of paragraphs, div-s and other elements.

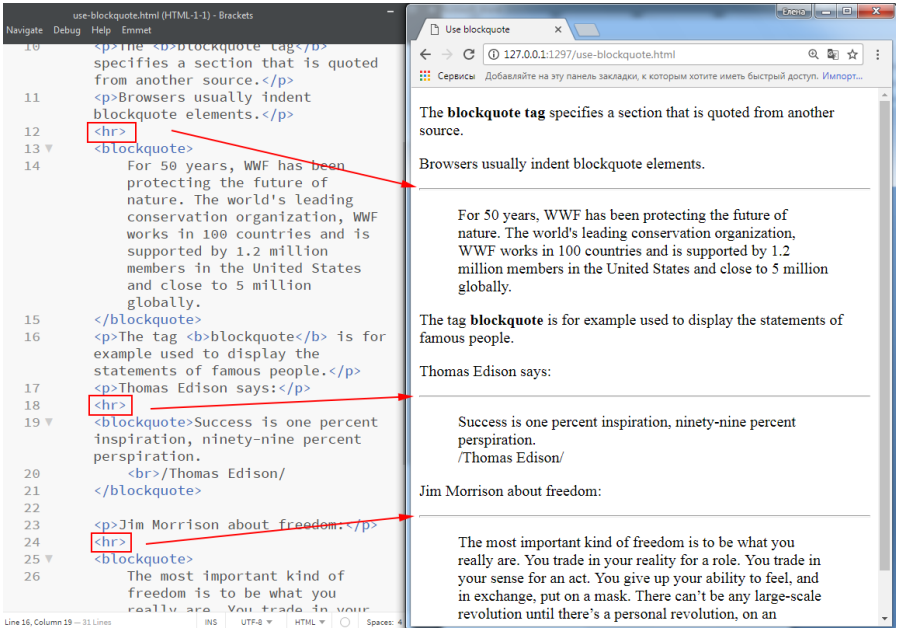
Since the tag does not have internal content, it does not need a closing tag, i.e. it refers to a group of single tags, or tags without content.

The same applies to the <img> tag for embedding images and the entire group of <input> tags that make up the forms.

Here we will consider one more such tag — this is <hr>, or a horizontal row.

Actually, it displays a horizontal row in the browser. In HTML4.01, for it, you could specify a number of attributes that are canceled in HTML5. Therefore, all the «beauty» should be set through css.

In the use-blockquote-hr.html *(example files are attached to the PDF file of this lesson)* file, you can add horizontal rows to visually separate quotes from the paragraph text. As you can see from the screenshot, <hr> is a block tag that occupies all available space in the browser.



- **Note:** *In XHTML, for all single tags it was mandatory to place the closing slash at the end separating from the tag name by a space, so on a number of sites you can find such a writing of tags:*

  At the moment, there is no need to add this slash.

```
<br />
<hr />
<meta http-equiv="Content-Type" content="text/html;
charset=utf-8" />
```

Let's summarize everything that we have learned about html-formatting at the moment and display it in the form of a list of rules.

HTML Rules

1. **All spaces, tabulation and the beginning of new lines are converted to a regular space.** Therefore, it is very desirable to format the text with indents:

```
<body>
    <h1>Заголовок 1 уровня</h1>
    <div id="test">

        <p class="block">Lorem ipsum dolor sit amet,
        consectetur adipisicing elit. Odio rerum,
        praesentium repellat cupiditate dolorum
        voluptatum. </p>

        <p class="block">Blanditiis similique, modi
        oficiis, dignissimos asperiores totam quae aut
        doloribus quam esse reprehenderit quos sed.</p>

        <p class="block">Eum facilis voluptates ut
        dicta, delectus, tempore animi repudiandae
        qui quis voluptas maxime ullam, assumenda.</p>
    </div>
</body>
```

2. **The entire text should be formatted in the form of tags.** There must be an opening and closing tag:

```
<element>element text</element>
```

3. **If the element has no content, the closing tag is NOT needed.**

```
<br>
<hr>
```

4. **There should always be base tags:** <html>, <head>, <title>, <body>. And they must correspond to the contents of the document.
5. **Nesting rule:** always observe the nesting of elements and prevent their intersection:

```
<b><i>Text inside tags</i></b>
```

6. **Attribute values are always in quotes**. An element can have several attributes, they are separated from each other by spaces:

```
<img src="images/someimage.jpg" alt="Alternative
Text" id="pic">
```

7. **Universal attributes** for all tags are: id, class, style, title, tabindex

```
<div id="box" class="block" title="Block element"
style="font-family: Verdana, Tahoma, sans-serif">
    Elementtext
</div>
```

8. **Comments** in HTML are designed to hide part of the code and are written like this:

```
<-- Comment  -->
```

CSS Styles

Well, we got to the visual formatting of the page. From the point of view of semantics, it is necessary to separate 'the wheat from the chaff', i.e. visual formatting of the site page from its html-structure. And CSS just does formatting. And it has a lot of properties for this in its arsenal.

You can add css-formatting in 4 ways:

1. Using the style attribute
2. Inside the <style> tags in the <head>
3. Using the <link> tag,
4. Using the @import directive.

In this lesson we'll look at the first 2 ways. The rest — in the next lessons.

Internal CSS styles (inline styles)

First, let's look at how you can set the formatting for each element. For this purpose, there is the style attribute, which we have already discussed as one of the universal ones, i.e. applicable for any tag.

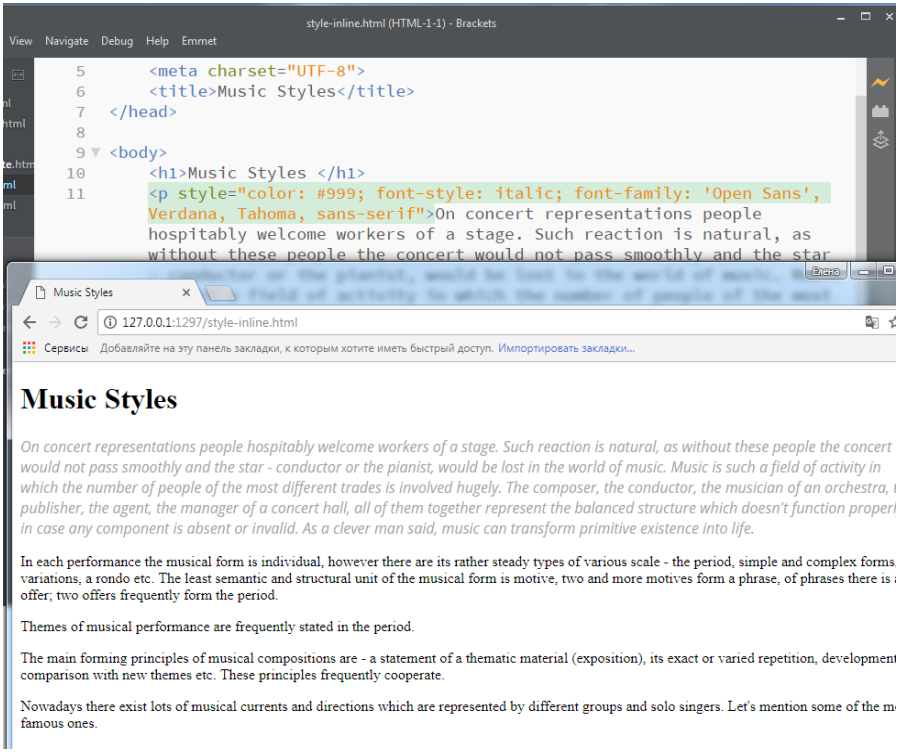
For example, we need to change the color and style of the text for a paragraph, and also assign a different font for it. This is done as follows:

```
<p style="color: #999; font-style: italic;
font-family: 'Open Sans', Verdana, Tahoma,
sans-serif">
```

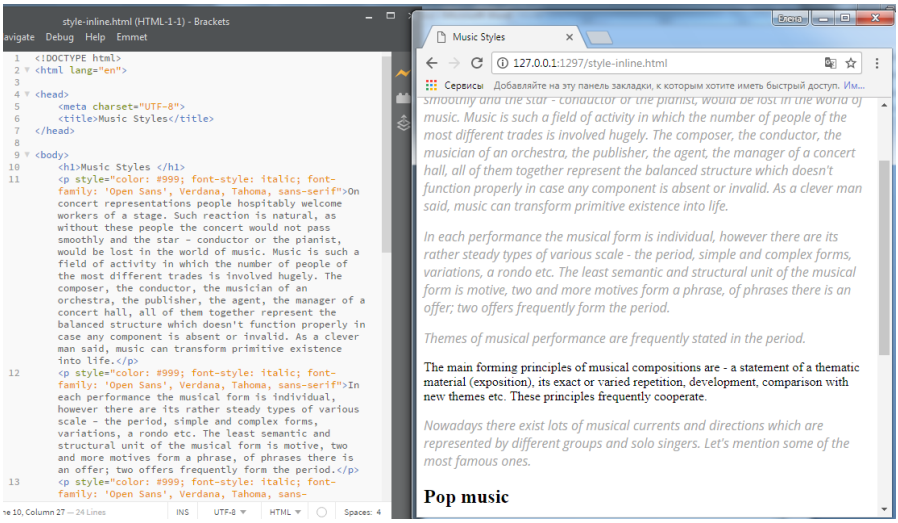
That is in the style attribute, which, I recall, is written only in the opening tag, you need to place the pairs

“property: value», which are separated from each other by a semicolon. After the last property value, a semicolon is optional. It’s important to add it if you still want to write one property.

The screenshot below shows the difference between the usual paragraphs and the paragraph with the formatting in the style attribute:



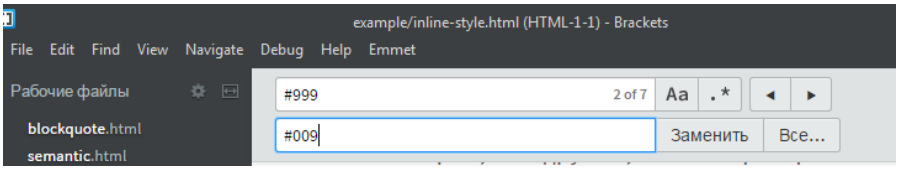
If we want to format all the paragraphs in the same way, you have to copy this attribute with all its values and add to all the paragraphs:



It already looks better, does not it?

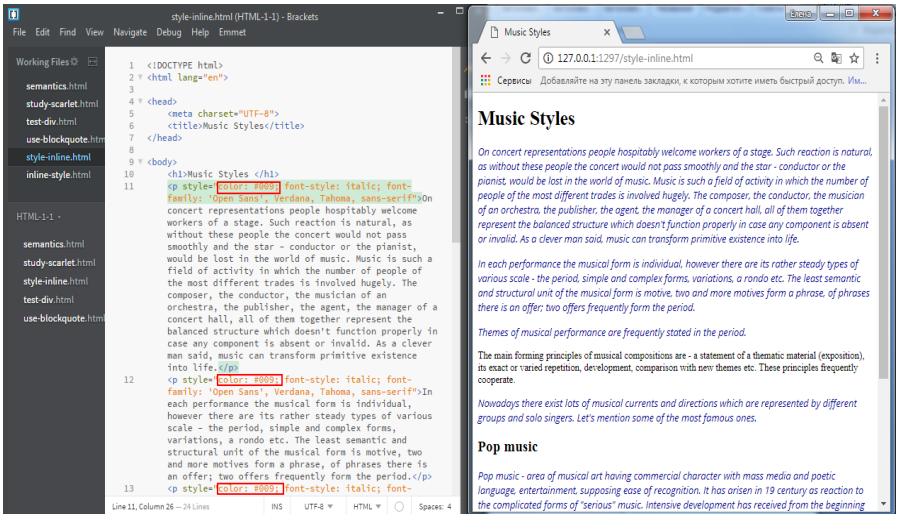
And now imagine that we have changed our mind about setting the color of the text in the form of color: #999, but we want to make it dark blue, for example ... For experiments, you can use the style-inline.html file *(example files are attached to the PDF file of this lesson)*.

In this case, of course, text search and replace help us (CTRL+H keys):



In the input fields, you need to specify what we replace in the first and with what in the second, and then click on the «All» button. All lines in which the text had a light gray color will be replaced.

Everything is done — the color became dark blue:



But, imagine that we have a lot of blocks, and we cannot change the color to the proposed one in all of them. Then the search and replacement will become more complicated. And if you still need to change both the font and the style, then you have much more problems. Therefore, this method — adding styles through the style attribute — is suitable either at the very beginning of the studying HTML/CSS, or when you have no other way out — for example, when editing articles in CMS Wordpress or Joomla, where you cannot select a part of the text in another way.

Styles for the page

It is much better, even at the initial stage, to use the second method — formatting inside the page in special tags <style> </ style>, which are placed at the bottom of the <head> block. It should be noted that in this case the styles are applied only for the page on which they will be posted. This is just an acceptable option to start learning

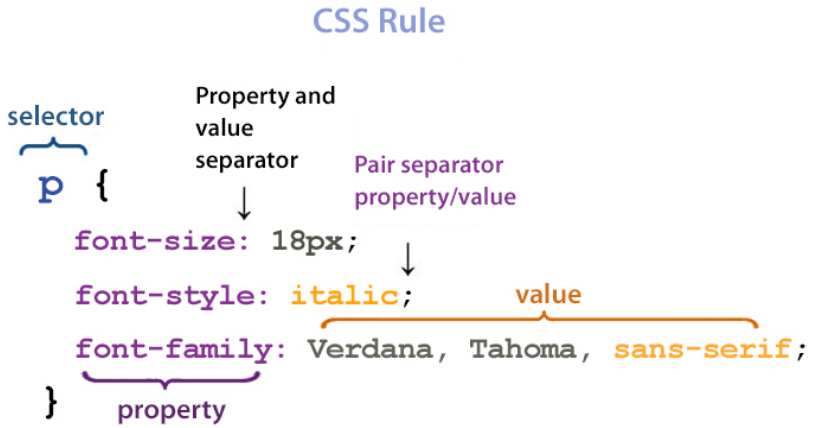
HTML and CSS — within the same document you see both html-markup and css-styles.

This kind of formatting allows you to specify styles for css selectors — special descriptions for an element or group of elements to which certain style rules apply.

At the moment CSS3 gives us a lot of different selectors, which we will learn sequentially on various examples. Now let's look at simple selectors, such as the universal selector, the element or tag selector, the group selector, the class selector, and the id selector.

Element selector

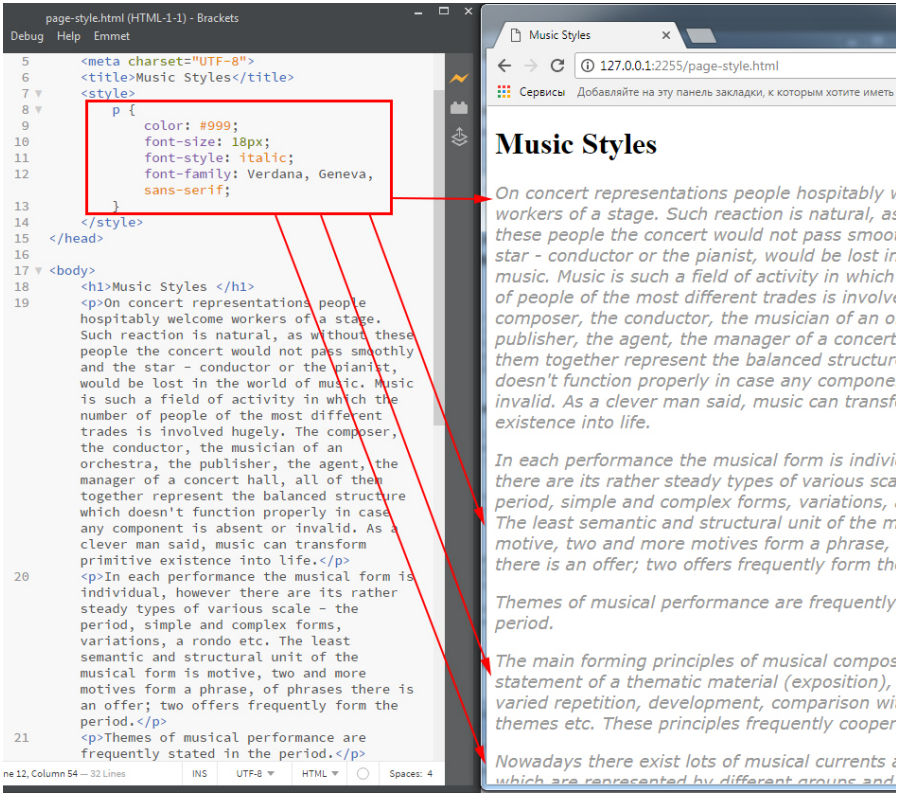
Let's start with the element selector and consider the general syntax. The figure shows that the selector is the p tag, the block of rules for this selector is placed in curly brackets, and the rules themselves are pairs «property: value» and are separated from each other by a semicolon.



It looks like a built-in style, is not it? The only difference is that such rules are applied immediately to all paragraphs

in the document — and you do not need to copy them into each. Plus, if you change any of the properties or when you add a new one, immediately all the paragraphs will change the formatting without the need to use «Search and Replace».

Add another color to the proposed rules, and here's what we get:

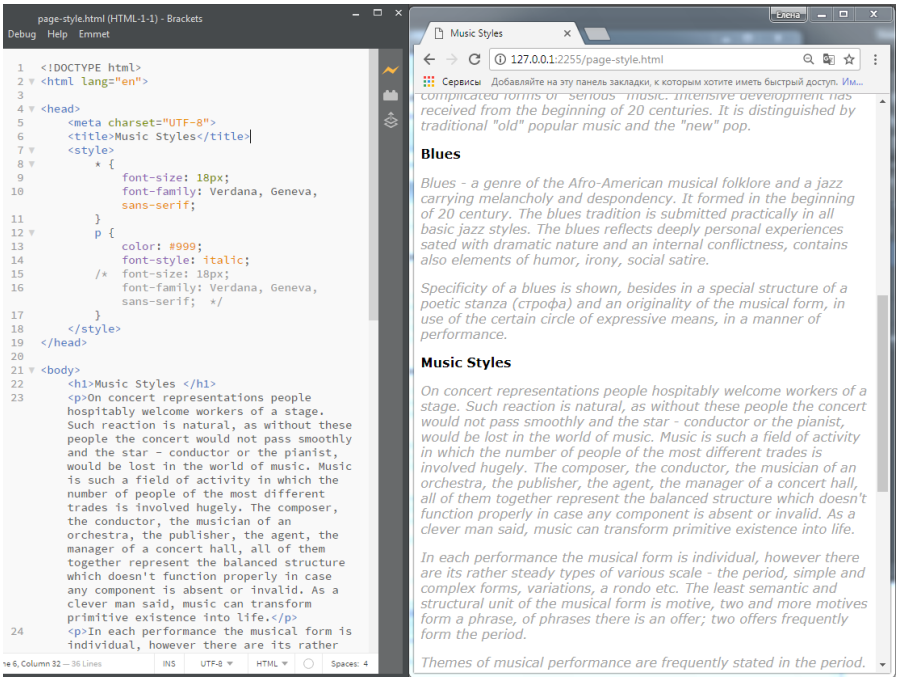


As you can see from the screenshot, all the paragraphs in the document changed the formatting without much effort to copy.

Universal selector

A universal selector is used to assign rules to all elements on the page, including html and body. It is indicated with the asterisk (* — SHIFT + 8 key).

It is necessary in order to establish the same rules for all elements of the page. For example, instead of setting the font-size and font-family only for paragraphs, we set them immediately for the entire page:



Because we do not have so many items on the page, the size and font family applied to paragraphs and headings. Note that the headings are smaller than before.

- **Note:** *in fact, they often set slightly different rules for a universal selector, but you will get acquainted with them*

*in the topic «Block model of elements». The same rules that we now set for * are usually written for the selector of the body element.*

Comments in CSS

Note that comments in CSS are different from comments in HTML and are set using

```
/*   ...   */
```

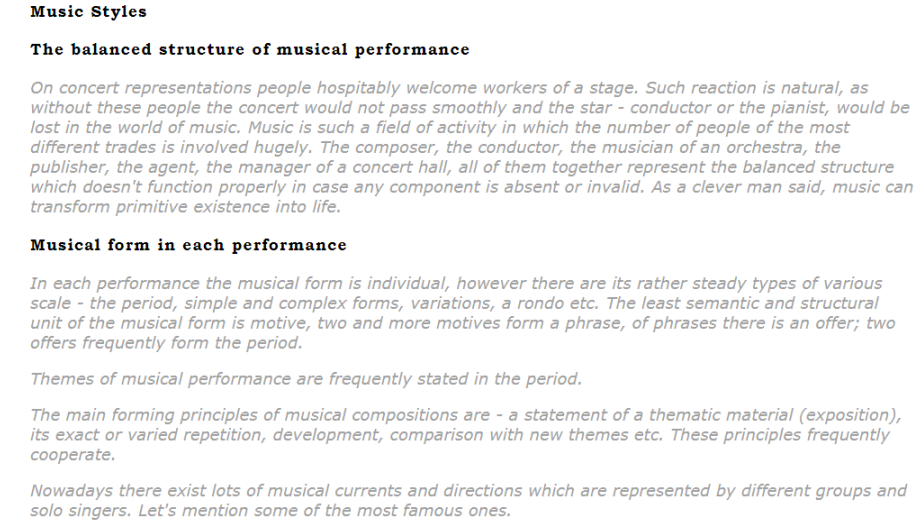
In the Brackets inside the style tags, these comments are also created with the CTRL+/ keys — the program itself determines which comments are needed in these tags.

Group selector

As its name suggests, the group selector is applied to a group of elements, classes, id, or combinations of these selectors. In it, all the selectors you need are listed comma-separated.

```
h1,h2,h3 {
    font-family: 'Bookman Old Style', monospace;
    line-height: 150%;
    letter-spacing: 1px;
}
```

We need a selector in order to reduce the amount of code for the same type of elements. In the example page-style.html *(example files are attached to the PDF file of this lesson)*, the font, the line spacing, and the spacing between the letters were changed for all headings.



Music Styles

The balanced structure of musical performance

On concert representations people hospitably welcome workers of a stage. Such reaction is natural, as without these people the concert would not pass smoothly and the star - conductor or the pianist, would be lost in the world of music. Music is such a field of activity in which the number of people of the most different trades is involved hugely. The composer, the conductor, the musician of an orchestra, the publisher, the agent, the manager of a concert hall, all of them together represent the balanced structure which doesn't function properly in case any component is absent or invalid. As a clever man said, music can transform primitive existence into life.

Musical form in each performance

In each performance the musical form is individual, however there are its rather steady types of various scale - the period, simple and complex forms, variations, a rondo etc. The least semantic and structural unit of the musical form is motive, two and more motives form a phrase, of phrases there is an offer; two offers frequently form the period.

Themes of musical performance are frequently stated in the period.

The main forming principles of musical compositions are - a statement of a thematic material (exposition), its exact or varied repetition, development, comparison with new themes etc. These principles frequently cooperate.

Nowadays there exist lots of musical currents and directions which are represented by different groups and solo singers. Let's mention some of the most famous ones.

A little later, an example with classes (*example file* style-class.html) will be considered, where the group selector looks like this:

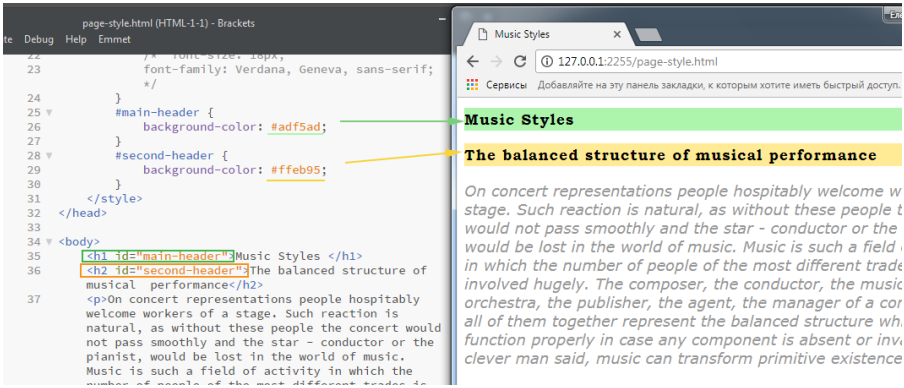
```
.cursive, .fantasy {
    font-weight: bold;
    letter-spacing: 2px;
    font-size: 1.5rem;
}
```

ID selector

The id selector implies that the id attribute (a unique identifier) is specified for any one element on the page. The value of this attribute in quotation marks cannot be repeated within one page, otherwise no uniqueness will be obtained. The id selector is denoted by the hash sign # next to the value of the id attribute of the desired element. In the example below,

we use 2 different id to specify a different background color for different headings:

```
#main-header {
    background-color: #adf5ad;
}

#second-header {
    background-color: #ffeb95;
}
```



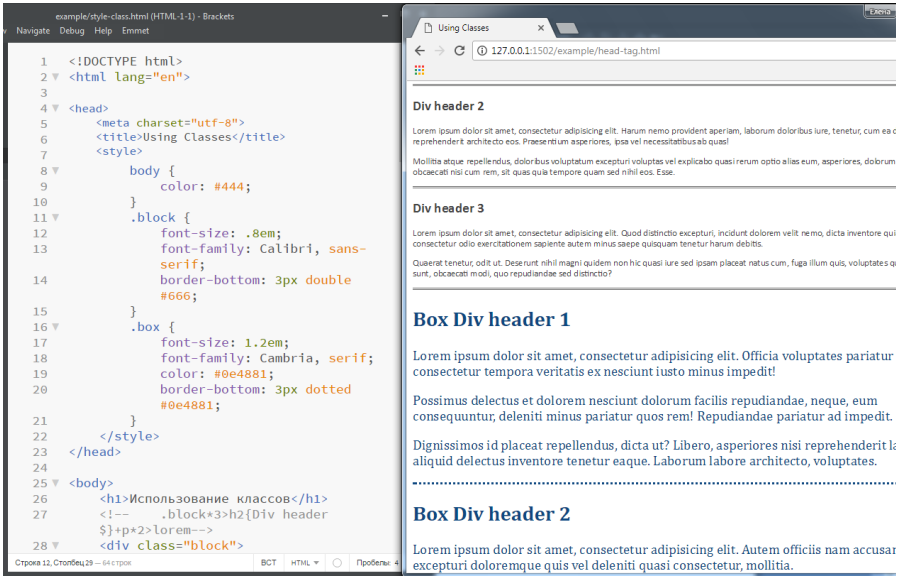
In html-markup, we added just such id for different headings. See the example in the file page-style.html *(example files are attached to the PDF file of this lesson)*.

Note that the value of each id is represented in a single instance. This is important, otherwise it is impossible to speak about the uniqueness and validity of the document.

Class selector

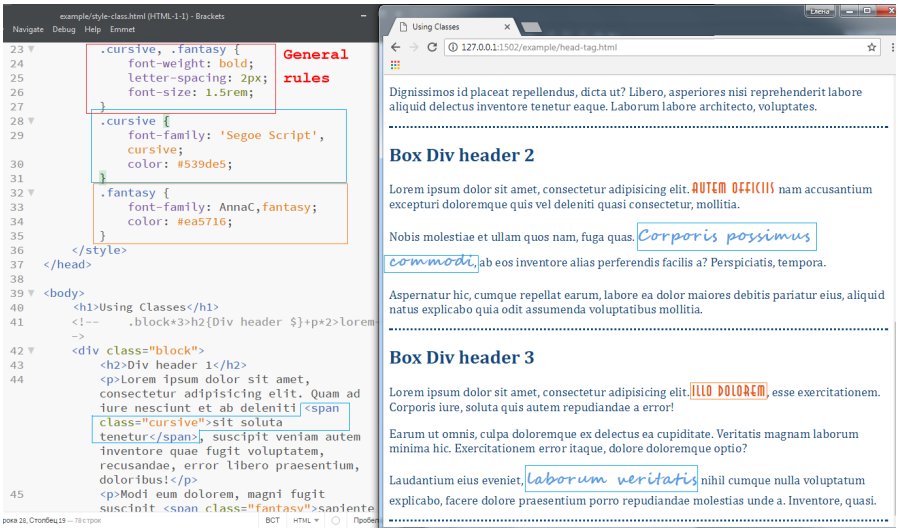
The class selector is the most popular for page layout. Any real page contains a lot of elements with different classes in the form of attributes. And most often such elements are div-s and span-s nested in them.

In the example below, the html-markup uses 3 div-s with the «block» class and 3 div-s with the «box» class. In the style-class.html file, you'll find Emmet abbreviations to do it yourself. For each of them a set of rules is specified, which changes the size and font family, and also forms a different type of border at the bottom of the div:



Plus, in <divclass=”block”> the text color is changed to dark blue. Now we will not add more complex style rules, but it is for class selectors that usually a lot of style rules are set.

We'll add the span elements with the “cursive” and “fantasy» classes to the markup, and set a number of general rules and special ones for them:



As a result, we will see that in the document (style-class. html) some of the text will turn orange and blue with a slightly larger font size. Immediately I will warn you that on computers not all of you have the appearance of the text that coincides with what is on the screenshot, because this depends on the fonts installed in the system.

CSS properties

We have already examined some of the CSS properties without being digging into their values. I think it was intuitively clear that the color property is responsible for the font color, and font-size for the font size of the element.

Now let's talk about the properties and their possible values in more detail. At once I will make a reservation, that within the limits of the first lesson we will consider the most necessary ones. All the rest will be considered as we study the course.

Color Assignment Options

To set the color of different elements, you can use several rules. We will consider today the color property, which is responsible for the color of the text, and background-color, which controls the background color of the element.

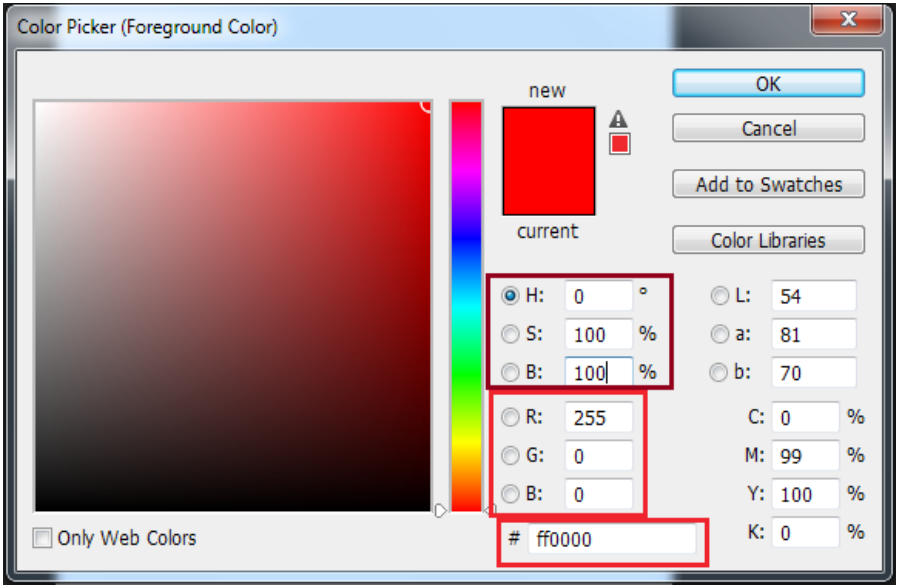
And in both properties it is possible to set values in the following formats:

```
color: red;/* the name of the color in English */
color: #ff0000;/ * hexadecimal color value that uses
       digits from 0 to 9 and letters from a to f*/
color: #f00;/* truncated hexadecimal value for colors
       that have the same pair of digits */
color: rgb(255, 0, 0);/* the color value in the rgb
       system. For each channel r — red, g — green,
       b — blue, the color values vary from 0 to
       255 units*/
color: rgba(255, 0, 0, .5);/* the color value in
       the system rgb + alpha channel, i.e.
       the ability to set the transparency for
       a color from 0 to 1 */
```

```
color: hsl(0, 100%, 50%);/* color value in the hsl:
hue-saturation — lightness */
color: hsla(0, 100%, 50%, .5);/* the color value
       in the hsl + alpha channel, i.e. the ability
       to set the transparency for a color from 0 to 1 */
```

Of all the options listed, the least understandable, perhaps, will be the last two, because the HSL color system (Hue, Saturation and Lightness) was hardly the subject of your study on the Photoshop course — there it is also rarely used (for the web, anyway).

All these options you can get from the Color Picker in Photoshop. Only in this program (the screenshot shows CS6version) instead of HSL — HSB (B — brightness).



As for the Brackets, one of it «goodies out of box» is a very simple selection and change of color properties. In order to

see this, create style tags, and for the selector of any element, for example for p, set the color property with any of the above values. And then put the cursor inside the color value and press CTRL + E. For fans of the right-click menu on the color value, you can select «Quick Editing».

- **Note:** *You can close the Quick Color Editing panel by pressing CTRL + E again, pressing the ESC key or the cross at the top left.*

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Color Property in CSS</title>
6      <style>
7          p {color: rgb(255
8      </style>
9  </head>
10 <body>
11
12 </body>
13 </html>
```

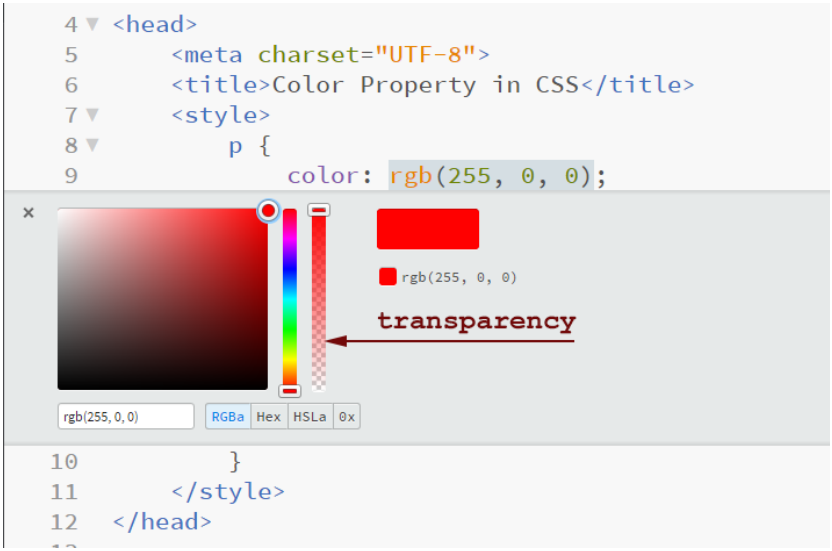
Quick Edit Ctrl+E
Quick Docs Ctrl+K
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Select All Ctrl+A
Minify Ctrl+M

Do not miss, please, because if the cursor is inside the property or outside the value of this property, pressing CTRL + E will cause an error message:

```
4  <head>
5      <meta charset="UTF-8">
6      <title>Color Property in CSS</title>
7      <style>
8          p {
9              color: rgb(255, 0, 0);
10         }
11     </style>
12 </head>
```

No Quick Edit available for current cursor position

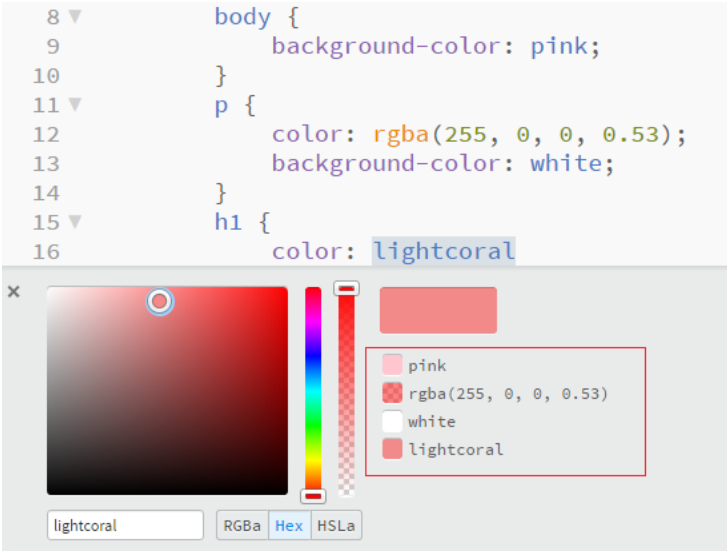
So, you pressed CTRL + E and you can choose between RGB, Hex, HSL. By default, you will have a Hex system with a hexadecimal color value. As for RGB or HSL systems, initially there is no transparency in them. You can set it by moving the slider on the bar to the right of the color palette.



A few more comments on the color:

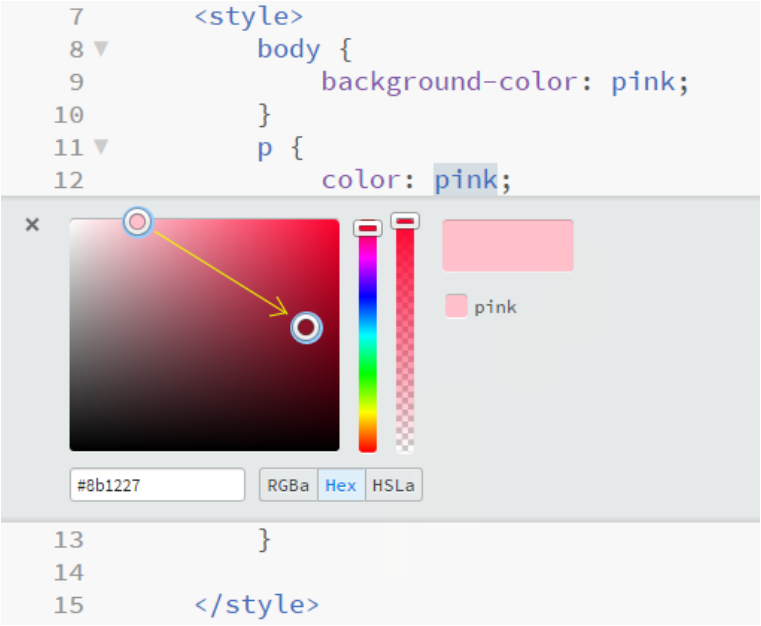
1. If you specify its value in the form of a word, Brackets will tell you all the variants of colors, where this word (or part of it) is contained.

2. If you used several colors in the project, assigning a color for the new element, in the quick edit panel you can see a list of all the colors that have already been used. Click on any of them and it will be assigned to your selector.



3. If you do not know what color you need or do not like the selected one, put the cursor after the colon and press CTRL+space — a prompt appears. And choose from the color values:

4. If you need to select the shades of near colors yourself, so that they suit each other, for example, for the background color — lighter, and for the text — darker, set the same values for both colors first, and then, by pressing CTRL + E, move to more dark area:



5. Finally, let's recall the Emmet abbreviations. In css they also work. For the color property, just write c and press TAB, and for background-color, type bgc and TAB. Abbreviations will open with default color values (black for text color and white for background color):

```
p {
    color: #000;
    background-color: #fff;
}
```

The values after the # sign are selected, so you can immediately type in the ones you need or press CTRL+E to select them.

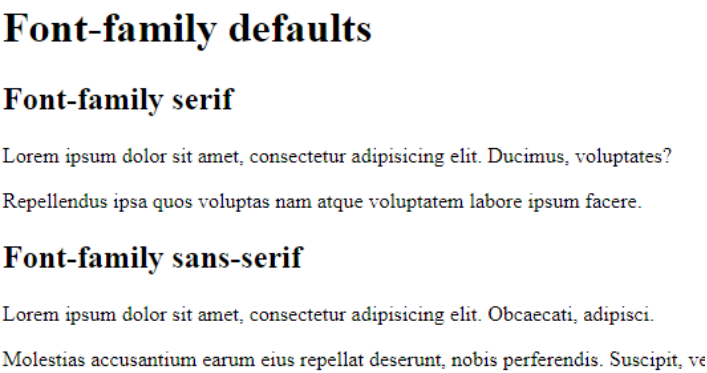
Font Properties

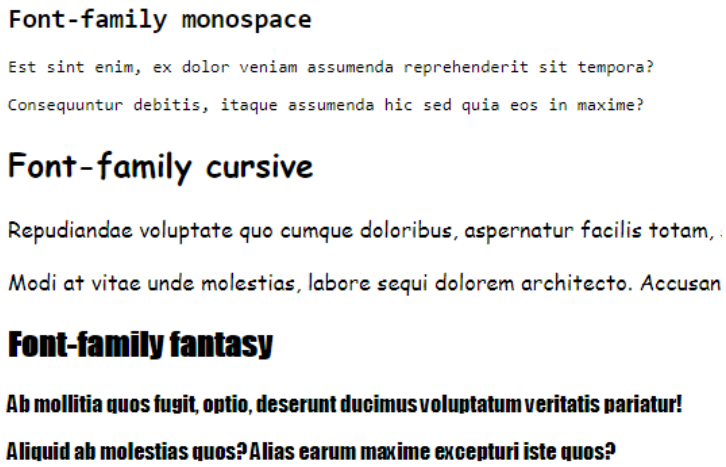
In css, the properties for the font begin with the word 'font'. There are several of them, and they perform the following font changes on the page:

1. The font-family property points to the font family. In the value of this property, you can list several fonts separated by commas in the order in which you (or the customer) would like them to be displayed on the site. For example,

```
body  { font-family: Lato, Verdana, Geneva, sans-serif; }
```

the record assumes that the browser first tries to display the text using the Lato font. But, if it is not on the user's computer, it will display the contents of the body with the Verdana font. If the Verdana font is also missing, the Geneva font will be used. And in the event that there is none of the listed fonts, the default font will be used, the type of which is set by a special keyword:





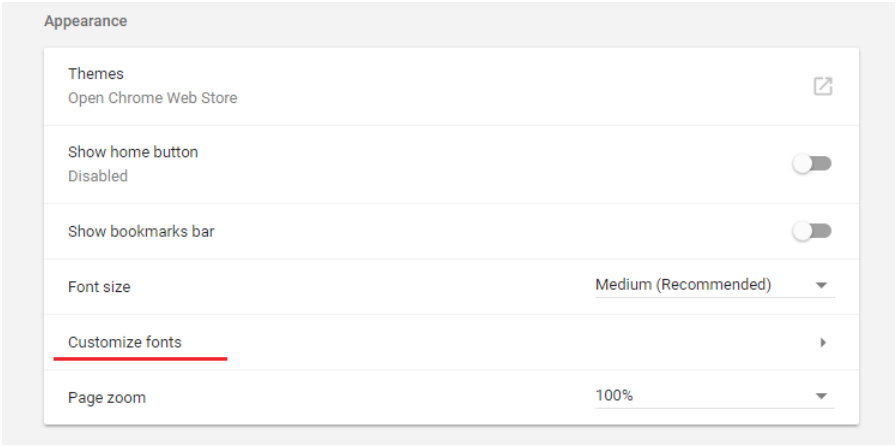
- serif — for serif fonts (default is usually Times New Roman);
- sans-serif — for sans-serif fonts (usually Arial);
- monospace — for monospaced fonts in which characters have the same width (usually Consolas or Courier New);
- cursive — for italic fonts (as an option — ComicSans MS);
- fantasy — for fancy, or decorative fonts (representative — Impact).

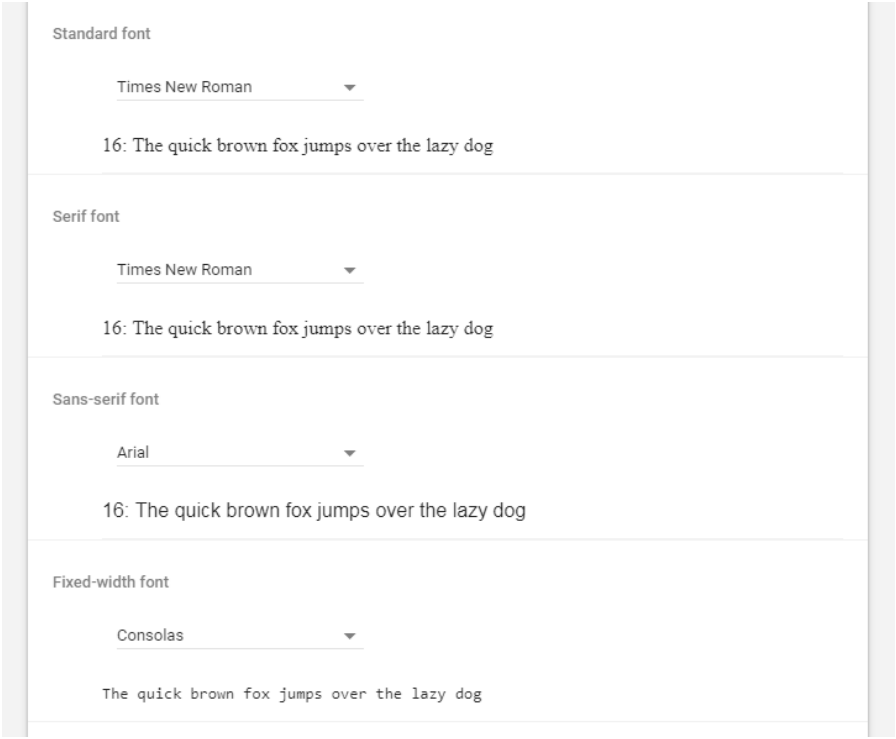
It is customary to specify the default font the last one, after all the more suitable variants. But, if you, for example, are quite satisfied with the standard sans-serif font Arial (and web pages usually use sans-serif fonts), then you can shorten the property record:

```
body { font-family: sans-serif; }
```

For tests, you can use the file font-family-test.html *(example files are attached to the PDF file of this lesson)*.

You can see which fonts are used by default in the browser settings. For example, for Chrome, they look like this:





- **Note:** *if you use a font whose name consists of 2-3 words, you should specify the font name in single or double quotes:*

```
body  { font-family: 'Arial Narrow',
        Calibri, sans-serif; }
h1 {font-family: "Bookman Old Style",
        Cambria, serif; }
Emmet abbreviation:  ff, ffv, ffa, fft, ffs, ffss,
        ffm, ffc, fff
```

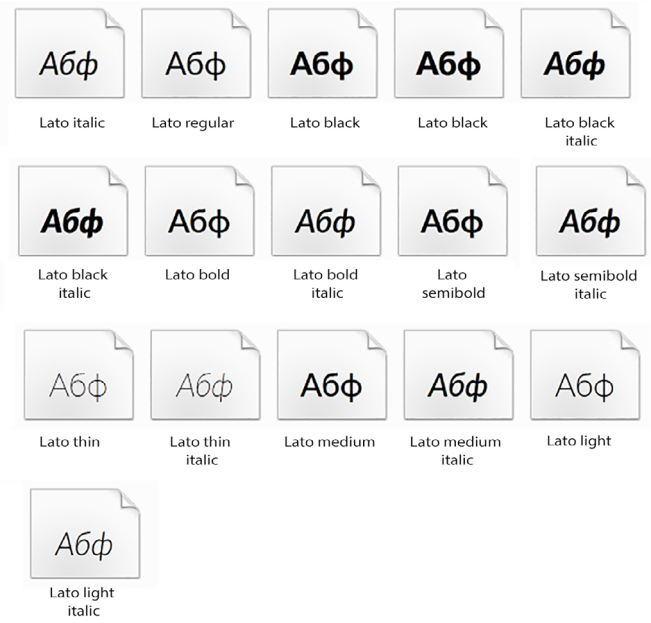
2. The font style is set by the font-style property. Values: normal | italic | oblique | inherit. The first value, normal, is default, i.e. the usual style of the text. The italic value is an italic font that assumes its presence in the user's system, oblique is the inclined font style, inherit inherits the value of the parent element.

```
Emmet abbreviation: fs, fsi, fso, fsn
```

3. The font-weight property corresponds to the font saturation, which has the values normal | bold | bolder | lighter | 100 | 200 … 800 | 900 | inherit. The default value is normal — the usual font saturation, which corresponds to font files that include the word Regular in the name. Accordingly, the bold value is responsible for the boldface. The bolder and lighter values change the thickness of the font to a greater or smaller relative to the parent value of this property. Units from 100 to 900 are indicated with a step of 100 and are distributed as follows: from 100 to 300 — this is usually a thin font, 400-500 — the usual outline, from 600 — a bold style. If we consider them in more detail, then the styke in units has the following values:

- 100 — Thin (Hairline)
- 200 — Extra Light (Ultra Light)
- 300 – Light
- 400 – Normal
- 500 – Medium
- 600 — Semi Bold (Demi Bold)
- 700 – Bold
- 800 — Extra Bold (Ultra Bold)
- 900 — Black (Heavy)

The actual display of fonts on the page is very much dependent on how many font files make up the font you selected for the page or item. For example, the Lato font, which is installed for my design purposes, and, most likely, is not available on your machine, has the following number of font files:



Lato Hairline Lato Hairline italic

Therefore, for this font you can choose different font-weight values, and they will differ from each other. For other fonts, the difference will be much less obvious, because they can consist of 1-3-5 files.

You can look at the different font options set with the font-weight property in the font-weight-test.html *(example files are attached to the PDF file of this lesson)* file, but for this you need to download the Lato font and install it to your system. This is done by copying the font files, for example, in the ttf format, to the system folder C:\Windows\Fonts.

```
body {
    font-family: 'Lato', sans-serif;
}
.w100 {
    font-weight: 100;
}
.w200 {
    font-weight: 200;
}
.w300 {
    font-weight: 300;
}
.w400 {
    font-weight: 400;
}
.w500 {
    font-weight: 500;
}
.w600 {
    font-weight: 600;
}
.w700 {
    font-weight: 700;
}
.w800 {
    font-weight: 800;
}
.w900 {
    font-weight: 900;
}
p.normal {
    font-weight: normal;
```

Сервисы Добавляйте на эту панель закладки, к которым хотите иметь быстрый доступ. Импортировать закладки...

Font-weight from 100 to 900

font-weight: 100. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Facilis nemo molestias quo, ut saepe repellat nobis unde voluptatum quam vero.

font-weight: 200. Dolore distinctio corporis ut nisi cumque commodi non officia itaque unde odit, at quis saepe aut, nobis sequi. A, voluptates.

font-weight: 300.Dolor animi inventore sed. Sequi nobis ipsam adipisci iste facere odit, ea amet aperiam recusandae vitae magni, explicabo! Blanditiis, placeat!

font-weight: 400. Quo eveniet minus iste odit adipisci aliquid voluptates, architecto nesciunt eligendi nam placeat itaque doloribus officia dolor magni sed earum.

font-weight: 500.Voluptas cumque suscipit dolorum, dignissimos ipsum praesentium nihil officiis obcaecati enim culpa a reprehenderit ullam quibusdam non accusantium molestias hic.

font-weight: 600. Tempora officia accusantium, unde facilis nobis at id minima earum rem, deleniti eos voluptas error ducimus vero ea quidem illo.

font-weight: 700. Dicta esse alias id saepe fuga asperiores minima. Libero aliquam, eveniet vero voluptatum veniam doloremque labore totam possimus hic quos.

font-weight: 800. **Incidunt voluptates natus, itaque vel non dolore sapiente quia, totam laudantium aut optio. Tenetur nemo provident, repellat maiores. Et, totam?**

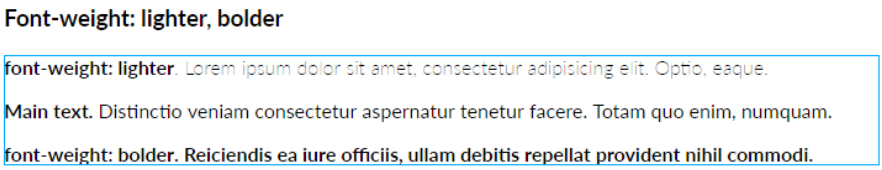
font-weight: 900. **Nostrum dignissimos qui ipsam architecto iure iusto voluptates, dolore cupiditate minus blanditiis incidunt illo, eum ab sapiente doloribus excepturi odio!**

Font-weight: normal, bold

font-weight: normal. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Optio, eaque.

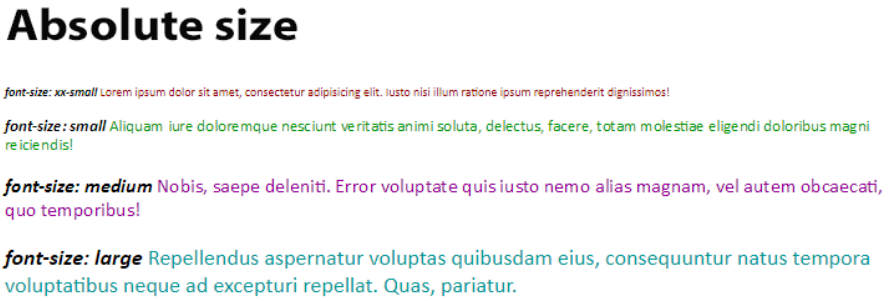
font-weight: bold. **Reiciendis ea iure officiis, ullam debitis repellat provident nihil commodi.**

In the file font-weight-test.html *(example files are attached to the PDF file of this lesson)*, at the very bottom, there are nested elements in with the bolder and lighter values of the font-weight property. The screenshot shows that in relation to the paragraph with the main text, the upper paragraph has thinner symbols, and the lower one — the bolder ones.



Emmet abbreviation: fw, fwb, fwbr, fwlr, fwn, fw200

4. Font-size. It is indicated in various units: cm, mm, pc, px, pt, em, rem, %, less often — ex, vh or vw, as well as in absolute and relative sizes. Font size can also have the inherit value, i.e. inherited from the parent. To specify the absolute size, the following keywords are used: xx-small, x-small, small, medium, large, x-large, xx-large. The larger and smaller values are used for the relative font size. The font size of the parent element is taken as 100%. You cannot use negative values.



font-size: x-large Labore doloremque, provident minus beatae similique cupiditate consectetur ea ad, porro quos necessitatibus accusantium officia.

font-size: xx-large Quod consequatur eum molestias consequuntur molestiae suscipit voluptates officia? Possimus facere ab nulla! Amet, maiores!

Relative size

font-size: smaller Lorem ipsum dolor sit amet, consectetur adipisicing elit. Alias, soluta dignissimos ipsa ad earum a pariatur nostrum ducimus. Iure, harum!

Source size Quos atque autem praesentium repellat amet consequatur dignissimos, veritatis nulla quam aliquid sit et quaerat suscipit perferendis, repudiandae aut ad.

font-size: larger Porro fuga voluptates id cumque, iure in assumenda rem, sapiente quia delectus error cupiditate quod, eum et! At, a, tempore.

Variants of the font size assignment are represented in the font-size-test.html *(example files are attached to the PDF file of this lesson)* file. It should be noted that now the font size is most often indicated in px, em, rem and %. All other options are rarely used.

```
Emmet abbreviation: fz, fz:15, fz:120%
```

5. Variants of the styles of lowercase letters — a font-variant property with the values normal | small-caps | inherit. Normal is the usual style of letters, and this is the default value: small-caps — all lowercase characters are displayed capitalized but of smaller sizes. The value inherit

indicates that this property is inherited from the parent element. At the bottom of the font-weight-test.html *(example files are attached to the PDF file of this lesson)* file, you will find an example of the font-variant property with a small-caps value:

font-variant: small-caps

LOREM IPSUM DOLOR SIT AMET, CONSECTETUR ADIPISICING ELIT. RECUSANDAE LABORUM DOLORUM FUGIT AUTEM, NECESSITATIBUS, SINT VOLUPTAS, TOTAM CUM ESSE MODI VEL PARIATUR NON ASSUMENDA, MOLLITIA SEQUI DELENITI OPTIO ALIAS VOLUPTATE!

```
Emmet abbreviation: fv, fvs
```

6. Line height, or leading, or line spacing — a line-height property. This property does not directly apply to the fonts-group, but is associated with these properties, because it's hard to imagine a well-formatted text without specifying the height of the line. For the values of this property, you can set: multiplier | a value in any units | percentage | normal | inherit. Usually, the line height is set at least 120% or by 1.2 multiplier, but it also depends on the font itself and is calculated automatically (normal value). Negative values are not allowed. Most often leading is indicated in the form of a multiplier or in percentage. You can look at the options for displaying different values of this property in the file line-height-test.html (example files are attached to the PDF file of this lesson).

```
Emmet abbreviation: lh, lh:n, lh:1.5, lh:140%
```

7. Universal property font, which allows you to specify several font features set by the previously listed properties. All

the values of its constituent properties are listed by a space. The exception is the font size and line height — they are separated by a slash. The syntax of this property is:

```
font: [font-style | font-variant | font-weight |
font-stretch] font-size [/line-height] font-family |
inherit
```

The font size and family are mandatory in this property. All other properties are specified as needed in the order shown above. You have learn the font-stretch property by yourself, because it is not used too often.

It should be noted that all properties that you have not specified acquire their default value.

To test some options for using the font property, you can use the font-test.html file *(example files are attached to the PDF file of this lesson)*.

Text Alignment

The text-align property is required to align text in the block element: left (the default value), right, center, or justify. And it's worth using it when you want to center the text without using the outdated tag.

```
text-align: left | right |center | justify | start | end
```

The new values start and end appeared in the CSS 3.0 specification and denote the following:

- **start** works the same as left if the text goes from left to right, and as right if the text goes from right to left;
- **end** works the same as right if the text goes from left to right, and as left if the text goes from right to left;

Test the text alignment using the file text-align-test.html *(example files are attached to the PDF file of this lesson)*.

```
Emmet abbreviation: ta:l, ta:r, ta:c, ta:j
```

Units of measurement in CSS

Important note: any units of measurement are written together with numbers. Thus, you cannot put a number, a space, and then specify a unit of measurement.

~~Incorrect: 16 px.~~ Correct: 16px

In CSS, there are absolute and relative units of measurement.

As for absolute units, in CSS you can use the centimeters (cm) and millimeters (mm), inches (in), points (pt — 1/72in) typical for printing, and picas (pc — 1/6in). But it is worth noting that in the real layout they are used extremely rarely. But one more absolute unit — a pixel, or px, — is used often, because it is pixels that measure the resolution of the monitor or another screen, and the browser also recounts many of the values in pixels.

But relative units are used very often. Perhaps the most frequently used units are em and a relatively recent rem, as well as percentage (%). What do they have in common? 1em and 100% is the font size of the parent element. That is if the following rule is written for the body:

```
body {font-size: 14px}
```

and the following is for the first level heading:

```
h1{font-size: 3em}
```

Then the browser will calculate the size of the heading in pixels, based on the font size of its parent (i.e. body), namely:

```
3*14px = 42px
```

If, however, the foloeing rule is written for h1:

```
h1{font-size: 300% }
```

then the calculation will be similar:

```
14px*300%/100% = 14px*3 = 42px
```

That is both em, and % are based on the font size of the parent element. Keep in mind that font properties are inherited by the chai, i.e. if h1 is in , for example, and not directly in the body, and for the font size is not set, then for the div, the font will also be 14px, and the heading will be calculated according to the same scheme. But, if the rule is given:

```
.logo {font-size: 1.3em }
```

The heading font will be of the following size:

```
14px (from body)*1.3 (for .logo) *3 (for h1) = 54.6px
```

And the same calculation is in %, only it is necessary to divide by 100.

The unit of measure 1rem (root em) indicates the font size, which is equal to the size specified for the font of the root element of the markup — the <html> tag. This size is set either in the user’s browser (usually 16px) or explicitly for the html page. And all calculations are made with respect to this size.

Another relative unit — 1ex — is based on the height of the "x" symbol of the element's font in lowercase. But in the presence of a more convenient unit, em is practically not used.

Currently, units such as 1vw (1% viewport width) 1vh (1% viewport height) are often used and are calculated relative to the size of the viewing area — width or height, respectively. And they are used more often not for specifying font sizes, but for setting such css-properties as width and height, which we'll talk about in the next lessons. You can also use the units vmin or vmax, which denote a smaller or larger value from vw and vh.

Homework Assignment

Today you will have a creative homework: you need to write your autobiography or CV using the tags and css-properties that were discussed in this lesson.

Do not forget to break the text into headings and paragraphs, highlight important things with or tags. You may use <hr> or <blockquote>, or or tags with classes. And css-formatting.

As an example, I will give an autobiography of one of my students. Humor and self-deprecation, as well as literacy + style, are welcome. ☺

Tatiana's beautiful life: from birth to the present day

My childhood

I was born 38 years ago. My childhood was quite happy. I had everything that the children of those times dreamed of. In addition to kilograms of cakes and marshmallows, always available in the house, I had something more interesting. On my eightieth birthday I got an electronic game «Mickey Mouse catches eggs». And a block of chewing gum «Donald Duck». Be sure, at that time I was the star of my street. Well, in any case, I felt that way.

The ABCs of my life

As for the rest, my life was quite ordinary if not trivial: school, technical college, university, marriage, birth of my son, realization that somewhere I turned the wrong way, the divorce.
And, of course, my career. Being equipped with my diploma with honors and trying to look for a job in my specialty «Accounting and Audit», I got into a different industry altogether. I became an appraiser. What can I say: the work was interesting, diverse, not boring, be sure. Ten years of life was spent on inspection,

search for analogs and formation of reports on the appraisal of real estate, goods, equipment and other interesting pieces.

Another stage of my work path

Some personal reasons led me to saying good-bye to the appraisal.

It's similar to a divorce with a football player, which makes you hate football too.

In general, my next step was to conquer the expanses of insurance business. More precisely, I got into the Claims Settlement Department at an excellent company Alfa Insurance. All would be fine, but after 2 years I was among those who got laid off. This was the so-called minimization of costs conditioned by the current difficult situation in the country.

How did I hit rock bottom…

```
And a year later I realized that I was not ready
to spend more time on work that did not bring
a good income. That's how I broke into copywrit-
ing. There I got the inspiration that I want to
learn how to create websites, not just fill them
with text content. That's why I'm here ;-)
```

Unit 1.
Introduction to the Web-Technologies. HTML Structure