

# HyperText Markup Language

## HTML & CSS



# Урок 1

## Введение. Структура HTML

# Оглавление

Что такое HTML. История создания .....	5
Стандарты HTML .....	7
Война браузеров.....	9
Стандарты HTML (продолжение) .....	12
Базовые определения HTML.....	14
Выбор редактора кода.....	19
Notepad++ .....	19
Sublime Text .....	20
Atom .....	22
Brackets .....	23
Использование Brackets для создания и редактирования файлов .....	24
Создание файлов в Brackets .....	25
Расширения Brackets .....	28
Темы для Brackets .....	30

Несколько слов о создании файлов HTML.....	32
Структура HTML-файла. DOCTYPE .....	36
Строгий тип документа.....	36
Переходный тип документа .....	37
Тип документа для фреймов .....	39
Валидация html-документов.....	40
Базовая структура html-документа.....	41
Кодировка документа .....	44
Тело документа — тег <body>.....	47
Комментарии в HTML.....	47
Использование плагина Emmet	
для создания структуры документа.....	50
Теги заголовков и абзацев .....	50
Что такое блочные элементы? .....	53
Абзацы в HTML .....	54
Несколько слов об атрибутах .....	55
Что такое Lorem Ipsum? .....	58
Обертки из аббревиатур Emmet .....	65
Вложенные теги.....	67
Теги div и span.....	70
Тег blockquote .....	73
Одиночные теги .....	74
Правила HTML .....	77
Стили CSS.....	79
Внутренние стили CSS (inline styles) .....	79
Стили для страницы .....	82

Селектор элемента.....	83
Универсальный селектор .....	85
Комментарии в CSS .....	86
Селектор группы, или групповой селектор .....	86
Селектор id .....	87
Селектор класса.....	88
CSS-свойства .....	91
Варианты назначения цвета.....	91
Свойства шрифта .....	97
Выравнивание текста.....	107
Единицы измерений в CSS .....	108
Домашнее задание.....	111

# Что такое HTML. История создания

HTML (от англ. *HyperText Markup Language* — «язык гипертекстовой разметки») — является на данный момент стандартом для документов, передаваемых в Интернете. HTML-разметка интерпретируется браузерами в виде отформатированного текста и может отображаться на экране монитора компьютера, планшета или смартфона.

HTML базируется на основе SGML (*Standard Generalized Markup Language* — стандартного обобщённого языка разметки), который появился в 1986 г., и соответствует международному стандарту ISO 8879. Этот язык был изначально предназначен для структурной разметки текста, но не содержал описания внешнего вида документа, который на нем можно было бы создать.

SGML подразумевал описание синтаксиса для написания главных элементов разметки текстов, причем уже тогда они получили название «тегов», он сам по себе не являлся системой для разметки текста и не имел списка структурных элементов языка для того, чтобы использовать их при создании документа.

Те не менее потребность в создании гипертекстового языка была, причем он должен был иметь:

- описание элементов и случаев их применения;
- перечень элементов для документа, который могут отображать специальные программы.

Именно поэтому в 1991 году Европейский институт физики частиц (**CERN**) в Женеве, Швейцария, объявил о необходимости разработки механизма, позволяющего передавать гипертекстовую информацию через Глобальную сеть. За разработку этого языка взялся сотрудник этого института Тим Бернес-Ли. И именно стандарт SGML лег в основу будущего языка **HTML** — *Hyper Text Markup Language*.



В соответствии с требованиями и нуждами CERN Тим Бернес-Ли разрабатывал HTML в первую очередь для обмена научной и технической документацией. Причем этот язык должны были без особых проблем использовать люди, не являющиеся специалистами в области верстки. Был определен небольшой набор структурных и семантических элементов — дескрипторов, которые затем стали называть тегами, которые позволяли создавать достаточно простые и в тоже время красиво оформленные документы. Таким образом, HTML успешно решал проблемы, оставшиеся от SGML.

Работая в CERN, Тим Бернерс-Ли занимался не только развитием HTML. В его задачи входило построение внутренней сети организации. Концепции, реализованные в ней, были доработаны и переросли в проект под названием «Всемирная паутина».

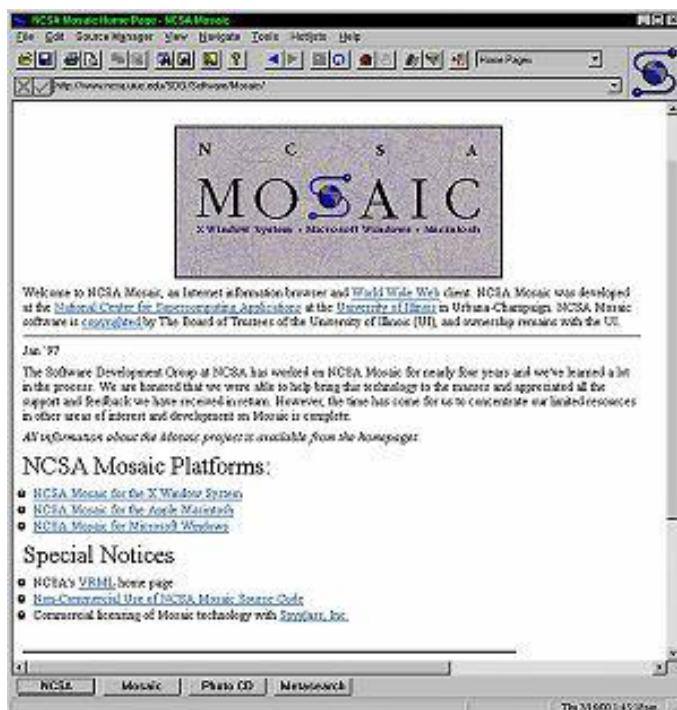
Проект подразумевал, что можно будет публиковать документы, размеченные при помощи HTML, в открытом доступе. Все документы должны иметь гиперссылки друг

на друга, что позволяло связать их между собой, превращая их в формацию, похожую на паутину.

Для реализации своей идеи Бернерс-Ли создал специальные программы: HTTP-сервер и WEB-браузер. Первый в мире веб-сайт был размещён *6 августа 1991 года* по адресу <http://info.cern.ch/> (вы можете посмотреть на его архивную версию [здесь](#)). В его содержании описывается принцип работы сети, как установить веб-сервер и создать простую страницу.

### Стандарты HTML

В 1993 г. появился HTML 1.2, который содержал всего 40 тегов. Эти теги, к сожалению, полноценного оформления страницы еще не давали.



Для отображения HTML-страниц была разработана специальная программа, которая получила название «браузер». И первым браузером стал «Mosaic », который был разработан в Национальном центре суперкомпьютерных приложений США (*National Center for Supercomputer Applications — NCSA*). За первый год было установлено около двух миллионов копий этой программы. Она поддерживала отображение картинок, распространялась бесплатно и помещалась на одну дискету.

Следующим этапом в развитии уже стандарта HTML было создание в апреле 1994 года Консорциума W3C (*World Wide Web Consortium*).

Поскольку официальной спецификации HTML 1.0 не существовало, именно W3C начал заниматься подготовкой спецификации HTML следующей версии. Ей сразу присвоили номер 2.0, чтобы подчеркнуть отличие от других версий HTML. Первый результат удалось получить спустя год насыщенной работы — в 1995 году. Из больших дополнений нужно отметить создание механизма форм для отсылки информации с компьютера пользователя на сервер.

Параллельно с разработкой HTML 2.0 велась разработка 3-ей версии стандарта, которая появилась в марте 1995 года и содержала в себе теги для создания:

- математических формул;
- страниц;
- примечаний;
- вставку рисунков, обтекаемых текстом;
- поддержку формата gif и т.д.

При том, что этот стандарт был совместим со второй версией, реализация его была сложна для браузеров того времени.

Кроме того, в 1995 году уже существовала необходимость в большем визуальном разнообразии страниц. Средств, предлагаемых HTML, особенно в рамках стандарта SGML, было недостаточно. Тогда корпорация W3C приступила к созданию дополнительной системы, не противоречащей основам HTML, но при этом позволяющей описывать визуальное оформление документов. В результате появился **CSS** — *Cascading Style Sheets* — Каскадные таблицы стилей, которые представляли собой иерархические стилевые спецификации. У них был собственные синтаксис, структура и задачи, которые позволяли дополнить теги HTML визуальным форматированием. Создание CSS было большим шагом вперед, т.к. потребности в визуальном представлении страниц сильно возросли, а HTML не предлагал для этого никаких средств.

### Война браузеров

Кстати, Mozaic уже не был единственным браузером. С 1994 года в сети появился **Netscape Navigator** и **Opera**, а с 1995 года — **Internet Explorer**. Чтобы отхватить как можно больший кусок потенциальной аудитории, привлечь максимальное количество новых пользователей, **Netscape** вводила в HTML все новые и новые усовершенствования, которые поддерживались, естественно, только одноименным браузером. Практически все новые теги были направлены на улучшение внешнего вида документа и расширение возможностей его форматирования.

Летом **1996** г. была выпущена версия Internet Explorer 3.0. Она поддерживала практически все расширения Netscape и обладала симпатичным и дружественным интерфейсом. Поэтому очень быстро утвердилась в качестве «второго главного браузера».

Четвертые версии обоих браузеров вышли практически одновременно и не отличались друг от друга особой скоростью работы или иными параметрами. Если Netscape нужно было покупать, то Internet Explorer начал поставляться бесплатно в операционной системе Windows и стал фактически стандартом отрасли. 90-е годы 20-го века ознаменовались таким понятием как [«Война браузеров»](#), которое, по сути, обозначает противостояние на рынке Интернета Netscape Navigator и Internet Explorer. Проблемой эта война стала для верстальщиков, т.к. каждый браузер старался внести свою лепту в развитие языка HTML и не слишком-то следовал стандартам W3C. Поэтому нередко на страницах сайтов можно было найти запись «Корректно отображается в браузере ...» со ссылкой на скачивание соответствующей программы.

Нужно отметить, что в конце 1990-х — начале 2000-х годов за счет того, что IE входил в Windows по умолчанию, он стал самым популярным браузером, особенно для не слишком опытных пользователей. Благодаря такому подходу в свое время корпорация Microsoft захватила львиную долю просмотров сайтов в Интернете, глубоко задвинув Netscape Navigator, хотя впоследствии свое первенство удержать не смогла.

Из Netscape Navigator со временем родился [Mozilla Firefox](#), который поддерживал стандарты W3C, имел

вкладки, а не окна и ряд фишек, переманивших на свою сторону продвинутых пользователей и верстальщиков. Достаточно много почитателей появилось и у браузера [Opera](#). Следующим новым браузером стал [Google Chrome](#), который был выпущен в свет 11 декабря 2008 года, и его исходный код стал доступен под названием [Chromium](#). С этого момента любая компания могла сделать свой браузер на основе этого кода. Например, так появились [Yandex-браузер](#) или [Amigo](#). Кроме того, Opera тоже перешла на движок WebKit, на котором был создан Chrome.

Chrome медленно, но уверенно занимал место на рынке браузеров, и к настоящему моменту является победителем, т.к. его используют порядка 50% пользователей для серфинга в Интернете. Кроме того, Google Chrome предоставляет удобные инструменты для разработчиков исходного кода — html-верстальщиков, JavaScript-программистов, поэтому и мы с вами будем его использовать в первую очередь.

Следует отметить, что *инструменты разработчика* сейчас присутствуют в любом браузере, в том числе и в Internet Explorer последних версий. И не последнюю роль здесь сыграло дополнение для FireFox с названием [FireBug](#), которое предоставляло информацию об html-элементах, css-правилах для них и позволяло отлаживать код JavaScript. В Opera для тех же целей существовало расширение [Dragonfly](#).

Остановимся еще на понятии «**кроссбраузерность**». Оно подразумевает одинаковое отображение сайта во всех браузерах. На данный момент эта проблема стоит уже не так остро, как в 2005–2012 гг., т.к. все современные

браузеры стараются поддерживать стандарты организации W3C. Тем не менее, сверстив сайт, следует проверить, как он отображается в самых популярных на данный момент браузерах. Для этого стоит посмотреть статистику на сегодняшний день, например на сайте <https://www.w3schools.com/Browsers/default.asp>:

<b>2017</b>	<b>Chrome</b>	<b>IE/Edge</b>	<b>Firefox</b>	<b>Safari</b>	<b>Opera</b>
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 %

<b>2016</b>	<b>Chrome</b>	<b>IE/Edge</b>	<b>Firefox</b>	<b>Safari</b>	<b>Opera</b>
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 %

Причем на данный момент стоит еще обращать внимание на использование мобильных браузеров.

## Стандарты HTML (продолжение)

Но вернемся к истории стандартов HTML.

Версия HTML 3.1 официально никогда не предлагалась, поэтому следующей версией стандарта **HTML стала 3.2**, выпущенная *14 января 1997 года*. В ней были опущены многие нововведения из версии 3.0. Зато были

добавлены нестандартные элементы, поддерживаемые браузерами Netscape Navigator и Mosaic.

18 декабря 1997 года была принята четвертая версия HTML. Стандарт HTML 4.0 содержал, как и в третьей версии, много элементов, специфичных для отдельных браузеров. Нужно отметить, что в 4-ой версии многие элементы были помечены как *устаревшие и не рекомендуемые к использованию*. Вместо них рекомендовалось использовать таблицы стилей CSS.

Далее идет стандарт HTML 4.01, который был утвержден 24 декабря 1999 года. В нем были приняты значительные изменения, и этот стандарт был популярен в сети довольно долго.

Параллельно с разработкой стандарта HTML ведется также разработка альтернативного стандарта XHTML (*Extensible Hypertext Markup Language*) — расширяемый язык разметки гипертекста, который использует подход XML. В нем предъявляются более строгие требования к синтаксису, чем в HTML. XHTML 1.0 был утвержден 26 января 2000 года в качестве рекомендации W3C. Вариант XHTML 1.1 одобрен в качестве рекомендации консорциума 31 мая 2001 года.



Фактически разработка новых стандартов HTML никогда не прекращалась, т.к. требования к отображению страниц постоянно увеличиваются. С 2007 года разрабатывался стандарт HTML5, который был окончательно утвержден 28 октября 2014 года. А с 17 декабря 2012 года уже ведется разработка стандарта HTML 5.1.

Мы будем изучать версию **HTML5**. Этот стандарт, по сравнению с предыдущими, добавил множество новых тегов, таких, как `<canvas>`, `<nav>`, `<section>`, `<header>`, `<footer>`, `<main>` и т.д., новых элементов форм, которые расширили возможности по структурированию html-документа.



Кроме того, мы будем использовать **стандарт CSS3** для визуального оформления страниц, в котором тоже добавили массу новых свойств, облегчающих процесс создания красивых веб-страниц без использования картинок или js-кода.

Следует отметить, что многие css-свойства, которые мы будем использовать, относятся к предыдущей версии — **CSS 2.1**. Но это и понятно — CSS3 стала наследницей предыдущего стандарта, дополнив и расширив его возможности.

Как и в стандартизации HTML, организация W3C не сидит на месте, а продолжает заниматься введением новых свойств, которые уже попадут в следующий стандарт — **CSS 4.0**. Прелесть этого процесса заключается в том, что новые свойства уже поддерживаются самыми последними версиями браузеров. А недостаток такой разработки стандарта — в том, что синтаксис свойства или целой группы свойств может меняться 2, а то и 3 раза, пока не будет окончательно утвержден, как это было с flexbox.

### Базовые определения HTML

Создание HTML-документа предполагает разметку всего его содержимого в виде тегов (первое название —

дескрипторов), которые записываются в угловых скобках. Текст должен располагаться между *открывающим (начальным) и закрывающим (конечным) тегами*. Например, теги абзаца с текстом выглядят так:

```
<p> Текст абзаца </p>
```

Существует еще одно понятие — это понятие **элемента**. Весь текст, заключённый между начальным и конечным тегом, включая и сами эти теги, называется **элементом**. Сам же текст между тегами является **содержанием элемента**. Учтите, что содержание элемента может включать в себя любой текст, в том числе и другие элементы.

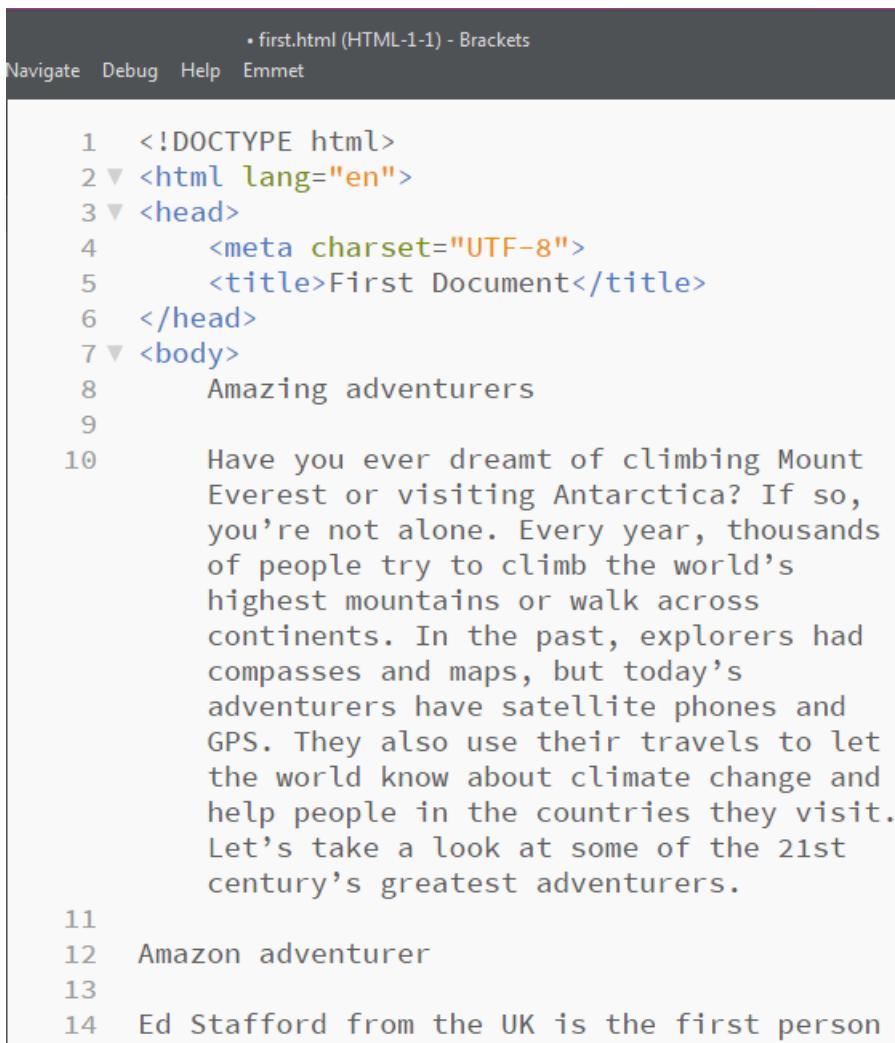
Для любого тега можно задать дополнительные свойства, которые называются **атрибутами**. Размещаются они только в открывающем теге и представляют собой пары вида «**свойство="значение"**», разделенные пробелами.

```
<p id="test" class="par"> Текст абзаца </p>
```

HTML является регистронезависимым языком разметки, поэтому допустимо писать теги как в верхнем, так и в нижнем регистре. Тем не менее, правилом де-факто стало уже давно написание тегов только **в нижнем регистре**.

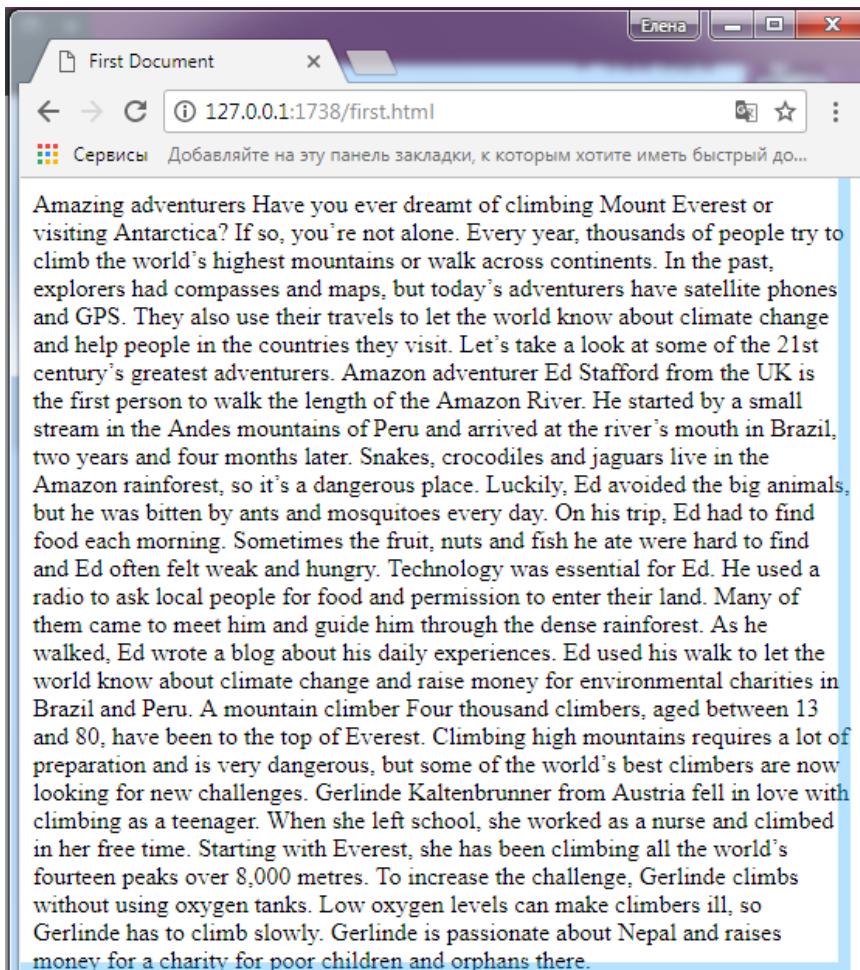
Документы HTML, по сути, являются текстовыми документами. Они имеют расширение **.html** или **.htm**, и для их просмотра необходим браузер, а для изменения — редактор кода, о выборе которого мы поговорим ниже.

Еще одной особенностью HTML является игнорирование пробелов и переносов строк. Как бы ни был отформатирован текст в вашем html-документе *ДО добавления тегов* — с отступами, сделанными клавишей TAB, с переносом каждой строки (клавиша ENTER) — в браузере он будет просто сплошным текстом.



The screenshot shows the Brackets IDE interface with the file 'first.html (HTML-1-1) - Brackets' open. The menu bar includes 'Navigate', 'Debug', 'Help', and 'Emmet'. The code editor displays the following HTML document:

```
1  <!DOCTYPE html>
2 ▼ <html lang="en">
3 ▼ <head>
4      <meta charset="UTF-8">
5      <title>First Document</title>
6  </head>
7 ▼ <body>
8      Amazing adventurers
9
10     Have you ever dreamt of climbing Mount
11        Everest or visiting Antarctica? If so,
12        you're not alone. Every year, thousands
13        of people try to climb the world's
14        highest mountains or walk across
15        continents. In the past, explorers had
16        compasses and maps, but today's
17        adventurers have satellite phones and
18        GPS. They also use their travels to let
19        the world know about climate change and
20        help people in the countries they visit.
21        Let's take a look at some of the 21st
22        century's greatest adventurers.
23
24     Amazon adventurer
25
26     Ed Stafford from the UK is the first person
```



Поэтому весь текст внутри html-документа должен быть разбит на теги соответственно *семантике*, или *внутренней логике* этого документа. И для различных уровней вложенности элементов, которые наверняка будут присутствовать на ваших страницах, **следует делать отступы** для визуального отделения родительских и дочерних элементов:

```
<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="Закрыть"><span aria-hidden="true">&times;</span>
      </button>
      <h2 class="text-center">Корзина</h2>
    </div>
    <div class="modal-body">
      <p>Пока товаров в корзине нет</p>
    </div>
    <div class="modal-footer">
      <button type="button" class="btn btn-order" data-dismiss="modal" data-
text="Продолжить покупки">Продолжить покупки</button>
      <button type="button" id="checkoutBtn" class="btn btn-success" data-
text="Оформить заказ">Оформить заказ</button>
    </div>
  </div>
```

Сейчас этот код кажется совершенно непонятным, но со временем станет привычным.

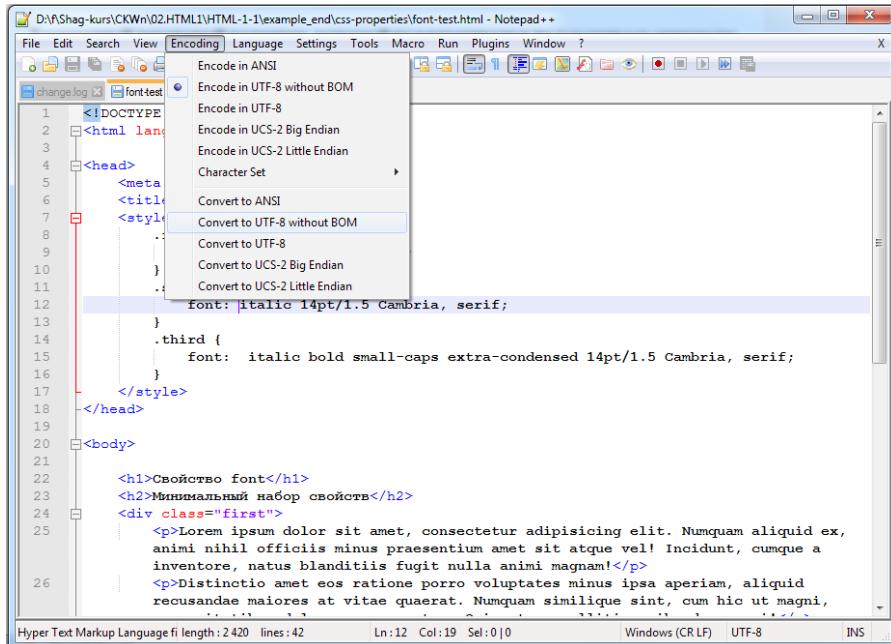
А теперь поговорим о главном инструменте верстальщика — редакторе кода.

# Выбор редактора кода

На данный момент существует масса редакторов кода для HTML/CSS. Рассмотрим наиболее популярные из них.

## Notepad++

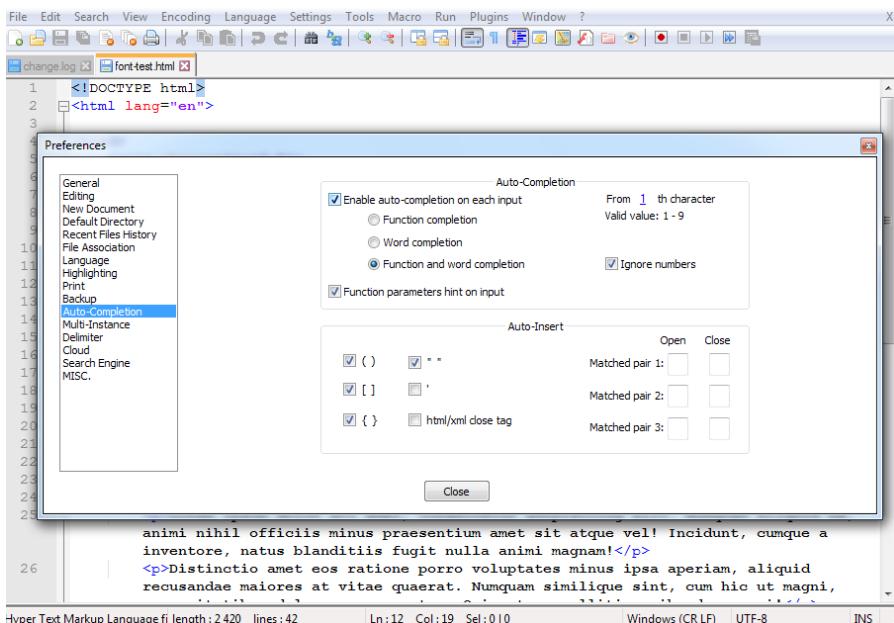
Бесплатный текстовый редактор, который поддерживает и подсвечивает синтаксис более 100 языков. Может открывать файлы в различных кодировках. На официальном сайте <https://notepad-plus-plus.org/> можно скачать последнюю версию программы.



Этот редактор кода удобен, быстро загружается. В нем можно настроить под себя некоторые функции

через меню Опции -> Настройки, например, автозаполнение, т.е. закрытие скобок, тегов и т.п.

Notepad++ разработан только для систем Windows 32- и 64bit.



## Sublime Text

Пожалуй, самый популярный текстовый редактор. Условно бесплатный. Т.е. вы можете скачать его без оплаты, но через определенное время он будет предлагать вам купить лицензию. Официальный сайт <https://www.sublimetext.com/> на данный момент предлагает 3-ю версию редактора для скачивания.

# Sublime Text

Sublime Text is a sophisticated text editor for code, markup and prose. You'll love the slick user interface, extraordinary features and amazing performance.

```

Demonstration - Sublime Text 2
File Edit Selection Find View Goto Tools Project Preferences Help
base64.cc x
31 void base64_encode(const uint8_t * data, size_t len, char * dst)
32 {
33     size_t src_idx = 0;
34     size_t dst_idx = 0;
35     for (; (src_idx + 2) < len; src_idx += 3, dst_idx += 4)
36     {
37         uint8_t s0 = data[src_idx];
38         uint8_t s1 = data[src_idx + 1];
39         uint8_t s2 = data[src_idx + 2];
40
41         dst[dst_idx + 0] = charset[((s0 & 0xfc) >> 2)];
42         dst[dst_idx + 1] = charset[((s0 & 0x03) << 4) | ((s1 & 0xf0) >> 4)];
43         dst[dst_idx + 2] = charset[((s1 & 0x0f) << 2) | (s2 & 0xc0) >> 6];
44         dst[dst_idx + 3] = charset[((s2 & 0x3f))];
45     }
46
47     if (src_idx < len)
48     {
49         uint8_t s0 = data[src_idx];
50         uint8_t s1 = (src_idx + 1 < len) ? data[src_idx + 1] : 0;
51
52         dst[dst_idx+0] = charset[((s0 & 0xfc) >> 2)];
53         dst[dst_idx+1] = charset[((s0 & 0x03) << 4) | ((s1 & 0xf0) >> 4)];
54         if (src_idx + 1 < len)
55             dst[dst_idx+2] = charset[((s1 & 0x0f) << 2)];
56     }
}

```

Line 31, Column 52      Spaces: 4      C++

Sublime Text позволяет работать со многими форматами. Он легко справляется с большими объемами текстовой информации. Редактор расширяется с помощью установки дополнительных пакетов. Минусом является то, что для настройки Sublime Text нужно потратить порядка 2-3 часов, т.к. вам будет недостаточно функциональности «из коробки».

# Atom

The screenshot shows the official Atom website. At the top, there's a navigation bar with links for Packages, Themes, Documentation, Blog, and Discuss. On the right, there's a "Sign in" button. The main area features the Atom logo (a stylized 'A' with a swirl) and the text "ATOM" in large letters. Below that is the tagline "A hackable text editor for the 21st Century". A red button with a download icon and the text "Download Windows Installer" is visible. To the right, there's a circular graphic with three people and a play button icon, suggesting a video. Below the video graphic, it says "For Windows 7 or later. Other platforms - Beta releases". At the bottom, there's a screenshot of the Atom interface showing a file tree on the left and a code editor window with some JavaScript-like code.

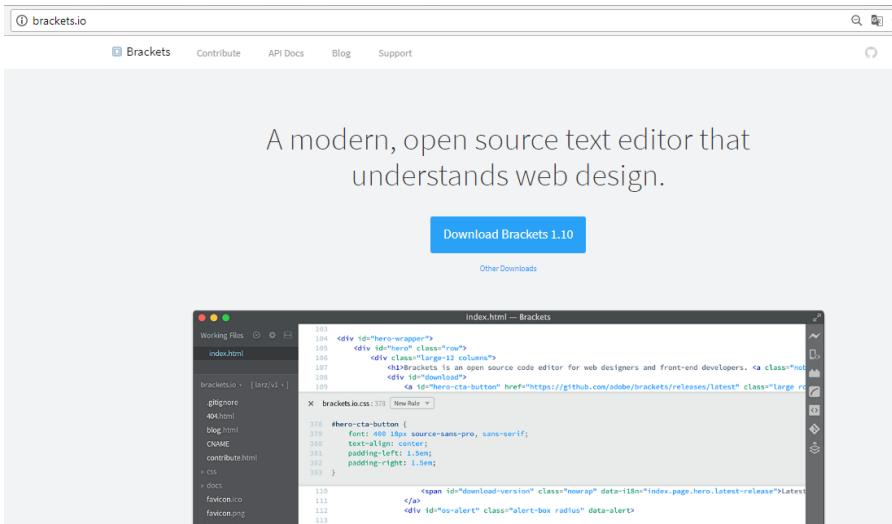
Цитата с официального сайта <https://atom.io/>:

«Atom — это текстовый редактор, который является современным, доступным, инструментом, который вы можете настроить для любых действий, но также продуктивно использовать, даже не касаясь файла конфигурации».

Этот редактор создавался командой Github. Он бесплатен, имеет открытый исходный код, подходит для macOS, Linux, Windows. Расширяется с помощью плагинов, написанных на Node.js, которые встраиваются под управлением Git Control. Также вы можете устанавливать и менять на нем темы, подбирая цветовые сочетания для подсветки кода, которые будут удобны для ваших глаз.

## Brackets

Бесплатный текстовый редактор, который поддерживает открытие файлов только в формате UTF-8 — самом распространенном на данный момент в web. Разработан он Adobe System для работы в первую очередь с HTML, CSS, JavaScript. Но также позволяет работать с php-файлами и имеет ряд расширений для создания тем для Wordpress, например. Официальный сайт <http://brackets.io/> позволяет скачать последнюю версию редактора.



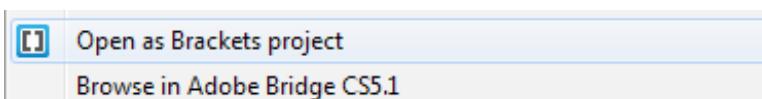
Brackets разработан для систем Windows, MacOS и Linux. Расширяется с помощью плагинов, также бесплатных, с открытым кодом, размещенных на Github.

Мы остановимся именно на этом редакторе, т.к. он имеет очень много «плюшек» прямо «из коробки» и позволяет упростить и ускорить написание кода. В условиях сжатого времени на обучение или выполнение заказа это бывает очень удобно.

## Использование Brackets для создания и редактирования файлов

Момент первый — в Brackets, как, впрочем, и в других редакторах, лучше работать с проектами — по сути дела, это просто папка на Вашем компьютере или на флешке, которая будет содержать html- и css-файлы, а также папки с изображениями и в дальнейшем с js-скриптами.

Поэтому первым делом создайте у себя на компьютере папку, в которой будете работать (у меня она называется HTML-1-1) и в контекстном меню для этой папки выберите пункт «Open as Brackets Project»:



В этом случае вам будут доступны все файлы и папки внутри проекта. Особенно вы это оцените, когда будете рассматривать тему о ссылках или об изображениях.

- **Примечание:** если окно Brackets уже открыто, вы можете просто перетащить папку с проектом в левую часть редактора или открыть нужную директорию, щелкнув по кнопке со стрелкой.

```

first.html (HTML-1-1) - Brackets
File Edit Find View Navigate Debug Help Emmet

Working Files
first.html
HTML-1-1 ▾
Open Folder...

drone-site - D:/f/Shag-kurs/CKWn/05.HTML2/09.LESS-SASS/LESS
music site - H:/Academy/CKWn/02.HTML1/12.Fixed&Fluid layout/DZ_guitar
tables - D:/f/Shag-kurs/CKWn/02.HTML1/10.Tables
t of climbing Mount

```

```

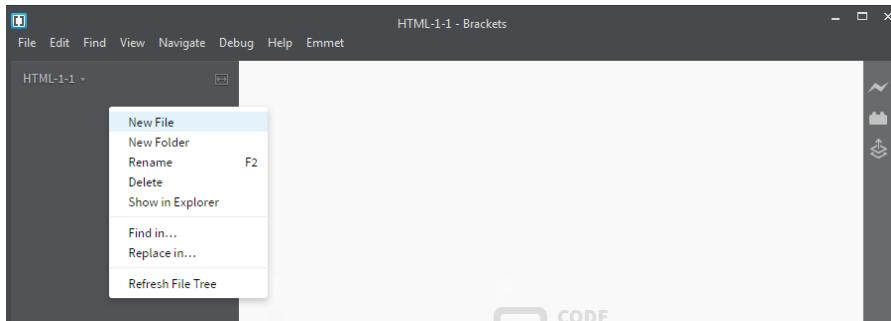
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5          <title>First</title>

```

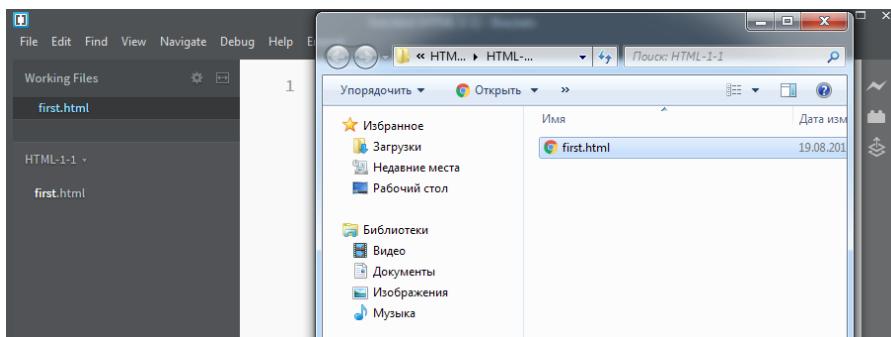
## Создание файлов в Brackets

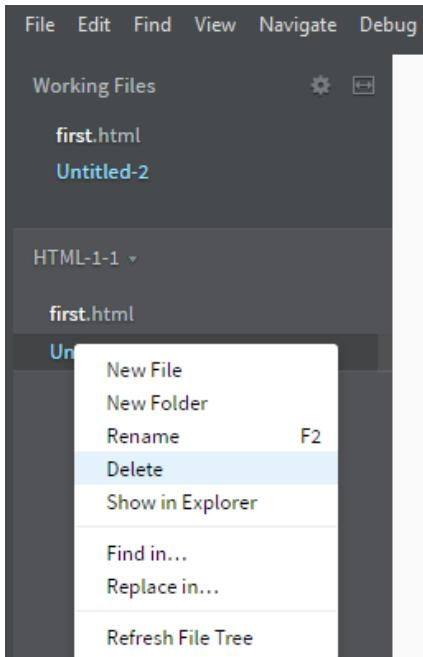
Вы можете создавать файлы в формате .html или .htm (а именно этот формат используется в веб) через меню файл->Новый (или **Ctrl+N**) и потом сохранять файл в нужную папку.

Но значительно удобней создавать файл в левой темно-серой области Brackets по правому клику на ней. В контекстном меню выбираем первую опцию «Новый файл»:



Давайте сразу договоримся, что веб не любит русскоязычных названий, поэтому все файлы мы будем называть латиницей. В идеале, конечно, нужно это делать на английском языке, но это не является строго обязательным.





По умолчанию новый файл будет иметь имя «Без названия-1» и (внимание!!!) не иметь расширения. Поэтому вводим вместо этого first.html — вуаля — файл одновременно создается в нужной папке и доступен для редактирования в Brackets.

Если вы создали лишний файл, то по правому клику вы можете его удалить или переименовать (клавиша F2 тоже поможет). Хорошой опцией является «Показать

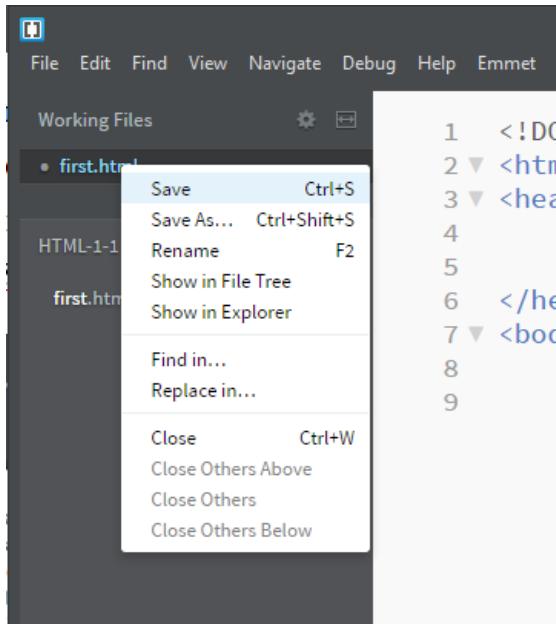
в проводнике» — в этом случае файл откроется в той папке, в которой он был создан. Временами это очень полезная опция, т.к. бывает так, что вы создаете файл совсем не в той директории, в которой собирались, а потом не можете его найти.

Кстати, в верхней строке редактора вы увидите названия файла, с которым работаете, а в скобках — папку, в которой он расположен.

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Document</title>
6  </head>

```



Еще одно замечание — когда вы набираете код, ваш файл перемещается в верхнюю часть левой темно-серой панели в блок с названием «Рабочие файлы». И если вы не сохранили изменения, рядом с ним появится точка — это сигнал о том, что нужно нажать **Ctrl+S** (или меню Файл->Сохранить).

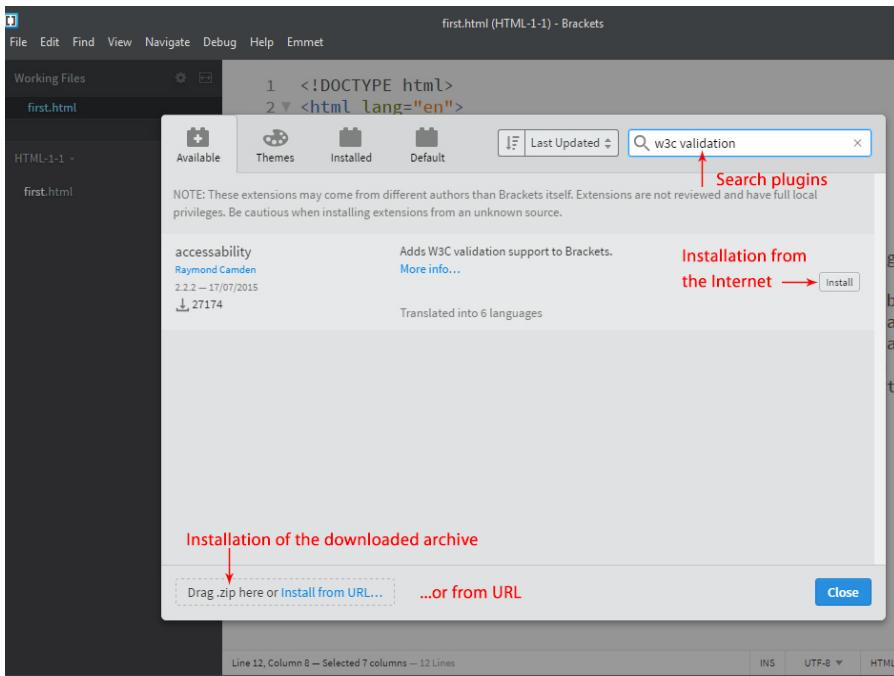
Для тех, кто не дружит с клавишами, а это не есть хорошо, по правому клику доступна опция «Сохранить» в контекстном меню, равно как и другие. Посмотрите — там справа написано **Ctrl+S** — запомните это сочетание клавиш и всегда его используйте!

Brackets, как и другие программы, временами может «зависать», и при его закрытии, например, через Диспетчер задач, все несохраненные изменения канут в Лету. И верстку придется начинать заново. А это так обидно!

### Расширения Brackets

Поверьте, это очень удобно — установить за несколько минут несколько важных расширений и использовать их в дальнейшей практике.

Плагины, или расширения Brackets, устанавливаются кликом на кнопке в виде лего справа на темно-серой панели Brackets или через меню «Файл -> Менеджер расширений».



Плагины подгружаются из Интернета на вкладке «Доступные», и их сразу можно устанавливать кликом на соответствующей кнопке. Либо можно найти плагин в репозитории на Github и, нажав на кнопку «Установить с URL», ввести адрес ссылки.

Плагинов много, поэтому стоит воспользоваться полем поиска. В этом случае из большого списка останутся только те, которые нужны.

Если соединение с Интернетом отсутствует, а плагины заранее были скачаны в какую-либо папку, то можно установить их, просто перетащив их на кнопку слева внизу с надписью «Drag .zip here».

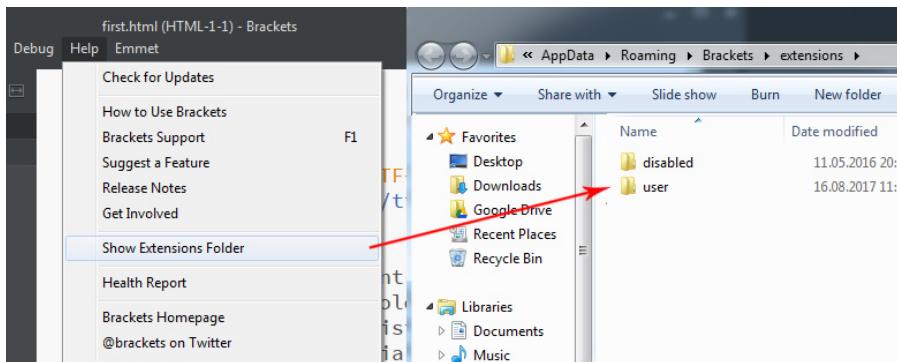
По поводу рекомендованных плагинов — установите следующий набор:

1. Emmet — <https://github.com/emmetio/brackets-emmet>. Этот плагин понадобится для ускорения верстки и написания css-стилей.
2. Jsbeautifier — <https://github.com/taichi/brackets-jsbeautifier> или Beautify — <https://github.com/brackets-beautify/brackets-beautify>. Само название плагинов — от английского слова beautify (украшать) — говорит о том, что любой из них позволит красиво отформатировать код.

Все остальные плагины мы будем устанавливать по мере необходимости.

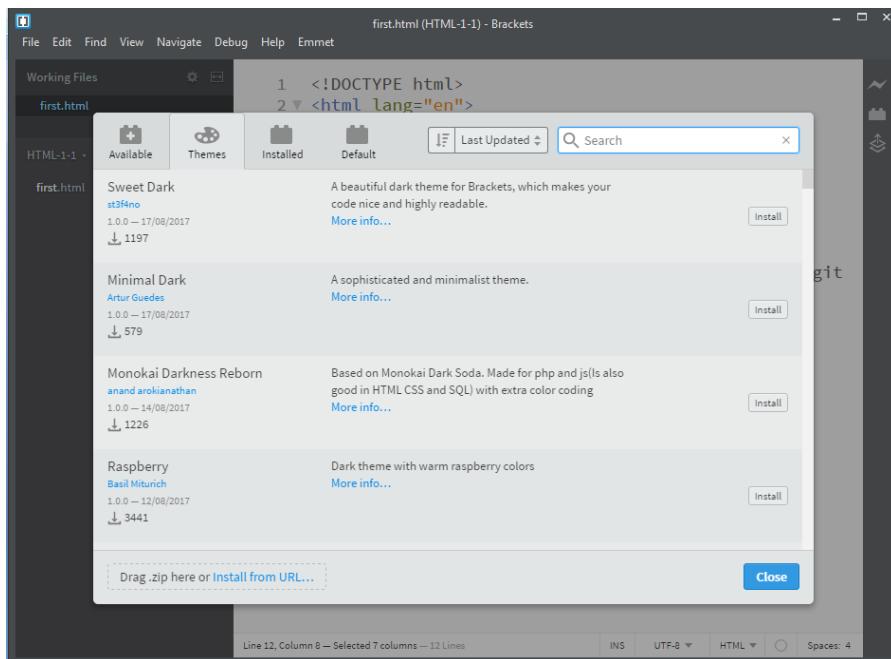
После успешной установки будет выдано сообщение, что установка завершена успешно.

Если по какой-то причине вы не смогли установить расширение, то можно разархивировать плагин из .zip-файла в папку с расширениями. Отобразить ее можно через меню Помощь->Показать директорию расширений. Откроется папка Roaming->Brackets. В ней будет папка user, в которую и необходимо распаковать архив. После этого **обязательна перезагрузка редактора**.

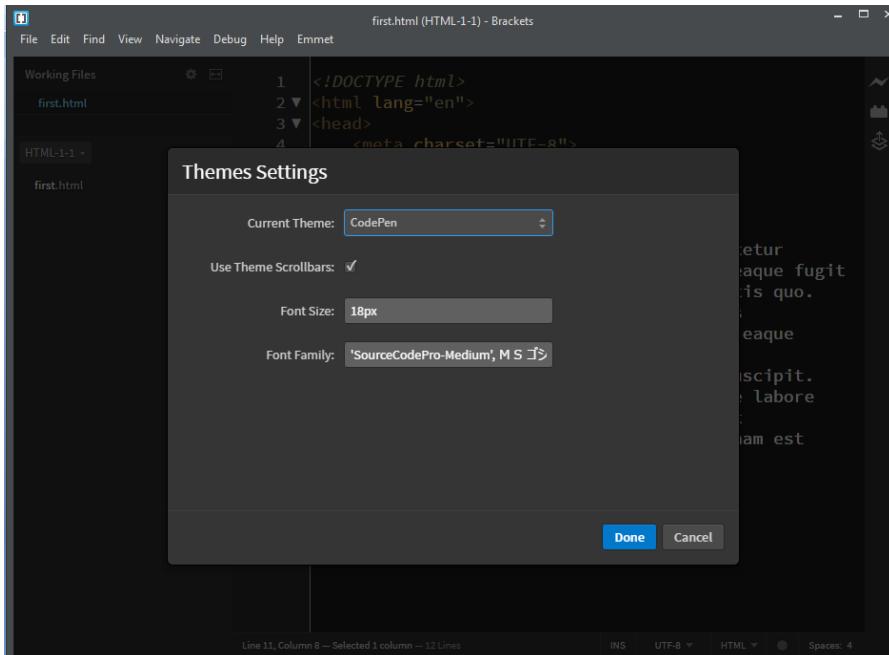


## Темы для Brackets

Загружаются так же, как и расширения, но на вкладке Themes. Выбирайте по названию или щелкайте по кнопке «Подробнее», чтобы посмотреть внешний вид темы на Github.



Чтобы тему можно было поменять, зайдите в меню Вид -> Themes... и выберите нужную. Вы сможете сразу же увидеть изменения. Остановливайтесь на той, которая наиболее приемлема для вас.



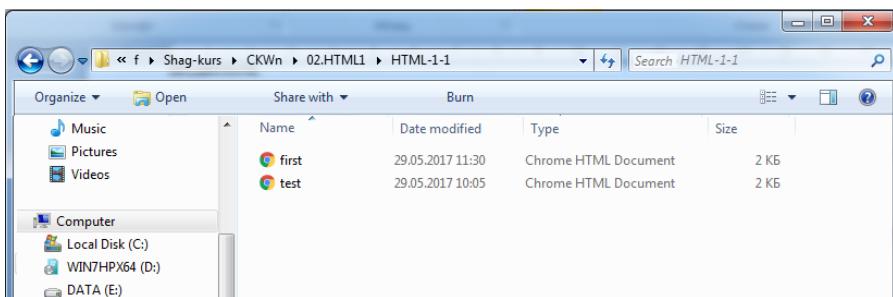
# Несколько слов о создании файлов HTML

Как уже было сказано выше, для работы с HTML необходимо создавать файлы с расширением **.html** или **.htm**.

В Brackets вы можете создать этот файл через меню Файл ->Новый или сочетанием клавиш **CTRL+N** и сохранить в нужной папке с нужным именем и расширением. Еще напомню про 2-й способ — это правый клик на темно-серой области слева с выбором опции «Новый файл».

Но наверняка, вам захочется создать файл и непосредственно в папке, с которой вы будете работать, используя проводник Windows. Здесь можно столкнуться с одним подвохом.

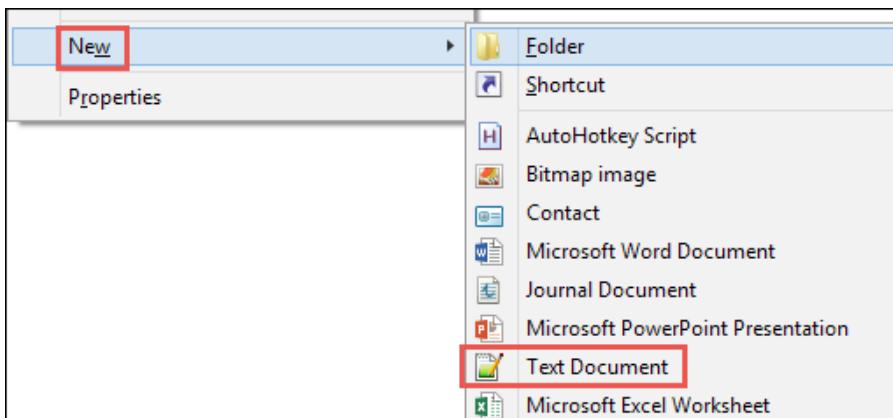
Дело в том, что в системах Windows, начиная с 7-й версии обычно принято скрывать расширения файлов. Поэтому внешний вид вашей папки будет примерно таким:



В папке видны иконки браузера Google Chrome, названия first и test и то, что они являются Chrome HTML

Document. Но, заметьте, что расширений файлов (.html) НЕТ!

Если попытаться создать в этой папке новый документ, то необходимо создавать обычный текстовый документ, т.к. по сути своей html-файлы являются именно текстовыми документами, но со своими особенностями — разметкой в виде тегов.



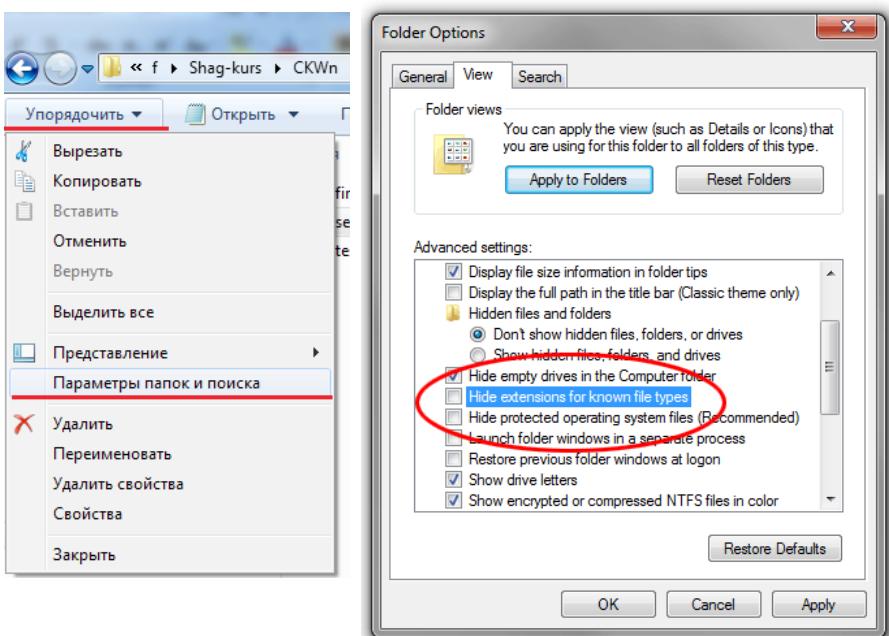
Name	Date modified	Type	Size
first	29.05.2017 11:30	Chrome HTML Document	2 КБ
test	29.05.2017 10:05	Chrome HTML Document	2 КБ
New text document	13:35	Text Document	0 КБ

Этот документ необходимо переименовать. Например, назовем его second.html:

Name	Date modified	Type	Size
first	29.05.2017 11:30	Chrome HTML Document	2 КБ
second.html	17.06.2017 22:35	Text Document	0 КБ
test	29.05.2017 10:05	Chrome HTML Document	2 КБ

Заметьте, что визуально НЕ поменялась иконка текстового документа и в колонке «Тип файла» осталась запись «Текстовый документ», хотя расширение .html присутствует в названии нашего файла. Если вы сделаете двойной клик на second.html, то откроется, скорей всего, стандартный Блокнот Windows.

Для того чтобы это изменить, необходимо в проводнике Windows в меню «Упорядочить» выбрать пункт «Параметры папок и поиска» и во вкладке «Вид» снять флажок «Скрывать расширения для зарегистрированных типов файлов»:

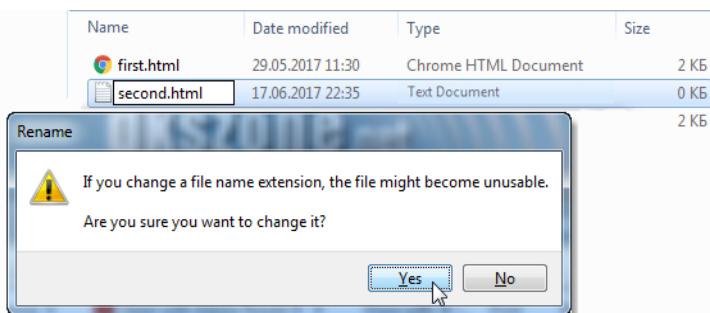


В этом случае сразу становится видно, что расширение нашего файла .txt, а .html — это часть названия файла.

## Несколько слов о создании файлов HTML

Name	Date modified	Type	Size
first.html	29.05.2017 11:30	Chrome HTML Document	2 КБ
second.html.txt	17.06.2017 22:35	Текстовый документ	0 КБ
test.html	29.05.2017 10:05	Chrome HTML Document	2 КБ

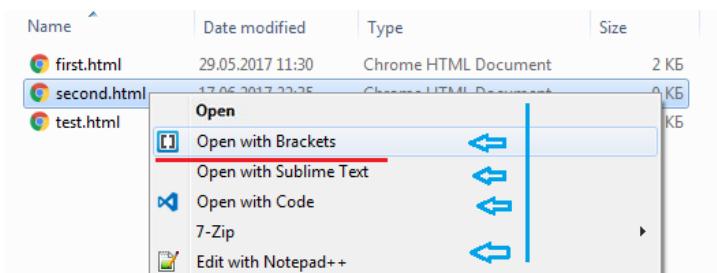
Теперь выделите файл, нажмите клавишу F2 и уберите расширение .txt.



Смело нажимайте кнопку «Да» в окне предупреждения и получите долгожданный html-файл.

Name	Date modified	Type	Size
first.html	29.05.2017 11:30	Chrome HTML Document	2 КБ
second.html	17.06.2017 22:35	Chrome HTML Document	0 КБ
test.html	29.05.2017 10:05	Chrome HTML Document	2 КБ

Теперь сделайте правый клик и выбирайте редактор кода. Мы будем открывать файл с помощью Brackets — пункт контекстного меню «Open file with Brackets».



# Структура HTML-файла. DOCTYPE

Любой html-файл имеет базовую структуру, которая состоит из тегов `html`, `head` и `body`. Но начинается она всегда с объявления типа документа — `DOCTYPE`.

Мы рассматриваем синтаксис последнего на данный момент стандарта HTML — это HTML5. Для него тип документа указывается крайне просто:

```
<!DOCTYPE html>
```

Предыдущими стандартами являются HTML4.01 и XHTML 1.0, которые существовали в различных вариантах: строгом, переходном и для фреймов, поэтому `DOCTYPE` необходимо было объявлять по-разному в зависимости от типа разметки, которая использовалась в html-документе.

## Строгий тип документа

Применяется строгий синтаксис языка соответствующего стандарта, а также допускается включать все теги и атрибуты, кроме осуждаемых.

Для HTML 4.01:

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

Для XHTML 1.0:

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

Синтаксис языка зависят от используемой версии, но в обоих вариантах должны соблюдаться следующие правила:

- Крайне нежелательно использовать такие теги, как: `<applet>`, `<basefont>`, `<center>`, `<dir>`, `<font>`, `<isindex>`, `<noframes>`, `<plaintext>`, `<s>`, `<strike>`, `<u>`, `<xmp>`, т.к. они относятся к осуждаемым (`deprecated`). Т.е. для данного типа документа использование перечисленных тегов является неприемлемым с точки зрения спецификации W3C. Вместо них необходимо задавать форматирование с помощью стилей `css`.
- Кроме того, нельзя добавлять в `<body>` любой текст, изображения и элементы форм напрямую. Все эти элементы должны находиться внутри других блочных элементов, например, `<p>` или `<div>`.
- Осуждается применение таких атрибутов, как `target` для ссылок (тег `<a>`), а также `start` (тег `<ol>`), `type` (теги `<li>`, `<ol>`, `<ul>`) и др. Мы сейчас не будем подробно останавливаться на том, что это за теги, т.к. этому будет посвящено отдельное занятие. Про [атрибуты](#) поговорим несколько позже уже в этом уроке.
- Еще не разрешено использовать фреймы.

### Переходный тип документа

В этом случае используется «мягкий» синтаксис языка, также можно использовать все теги и атрибуты, включая осуждаемые.

Для HTML 4.01:

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

Для XHTML 1.0:

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

Цель таких переходных доктайпов заключается в постепенном знакомстве с синтаксисом языка. Он может помочь на начальном этапе изучения языка, особенно, если вы делаете это по старым учебникам или руководствам в интернете. В документах с переходным доктайпом можно использовать атрибут `target`, который позволяет открывать ссылку в новом окне. Также можно использовать такие теги, как `<center>` или `<font>`, которые так милы сердцу большинства начинающий html-кодеров. Но в этом DOCTYPE фреймы также не разрешены.

Сразу хочу оговориться по поводу переходного DOCTYPE — если у вас есть элементы форматирования, которые, прямо скажем, на данный момент уже устарели — такой доктайп позволит создать валидный документ. Но, если вы всерьез решили заниматься версткой — забудьте о тегах `<center>` или `<font>` — такие CSS-свойства, как `text-align: center` или `font-family, font-size и color` дадут вам значительно больше вариантов управления внешним видом документа.

### Тип документа для фреймов

Фреймы в свое время (начало 2000-х годов) были очень популярным инструментом для загрузки в один html-файл 2-х, 3-х или более других html документов. Но на данный момент существует масса способов создать красивый html-файл, не прибегая к таким сложностям. Для фреймовой структуры — frameset — применяется специальный DOCTYPE, который по синтаксису аналогичен переходному.

Для HTML 4.01:

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

Для XHTML 1.0:

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

У вас может возникнуть вопрос: а зачем вам знать обо всех этих типах документов, если сейчас достаточно использовать простой `<!DOCTYPE html>`?

Дело в том, что в сети еще очень много сайтов, созданных 5-10-15 лет назад, для которых, возможно, вам придется делать редизайн (т.е. изменение внешнего вида) или просто править часть кода. И важно понимать, что можно, а что нельзя оставить в документе, чтобы код считался валидным, т.е. соответствующим стандартам организации W3C.

### Валидация html-документов

Валидация html-документов — это проверка отдельных html-документов и сайтов на правильность кода. Доступен по ссылкам:

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

На первых порах вам больше пригодится валидатор, позволяющий загрузить файл, а не указать ссылку на сайт в Интернете. Поэтому ссылка будет такой: [https://validator.w3.org/#validate\\_by\\_upload](https://validator.w3.org/#validate_by_upload)

The screenshot shows the W3C Markup Validation Service website. At the top, there's a blue header bar with the W3C logo and the text "Markup Validation Service". Below it, a sub-header says "Check the markup (HTML, XHTML, ...) of Web documents". There are three tabs: "Validate by URI", "Validate by File Upload" (which is selected), and "Validate by Direct Input". The main area has a light gray background. It contains a "Validate by File Upload" section with a sub-section for "Upload a document for validation". A file input field is present with the placeholder text "Выберите файл" and "файл не выбран". Below this is a link "More Options". At the bottom of the validation area is a "Check" button. A note at the very bottom states: "Note: file upload may not work with Internet Explorer on some versions of Windows XP Service Pack 2, see our [information page](#) on the W3C QA Website."

Вот, что выдал валидатор при загрузке документа с <!DOCTYPE html> и тегами <font> и <center> внутри:

### Nu Html Checker

This tool is an ongoing experiment in better HTML checking, and its behavior remains subject to change

Showing results for second.html

Checker Input

Show  source  outline  image report

Check by   Файл не выбран

Uploaded files with .xhtml or .xht extensions are parsed using the XML parser.

Use the Message Filtering button below to hide/show particular messages, and to see total counts of errors and warnings.

1. **Error** The `font` element is obsolete. [Use CSS instead.](#)

From line 13, column 1; to line 13, column 18

`01:<h2><--<font color="red">&lt;!--`

2. **Error** The `center` element is obsolete. [Use CSS instead.](#)

From line 15, column 1; to line 15, column 8

`</font><--<center>Для ХН`

3. **Error** The `font` element is obsolete. [Use CSS instead.](#)

From line 17, column 1; to line 17, column 20

`/center><--<font color="green">&lt;!--`

Как видно, он оба элемента отнес к категории **obsolete**, т.е. устаревших тегов. Как вы думаете, стоит их использовать в современном стандарте?

Почитать еще о DOCTYPE:

- <http://ruseller.com/htmlshpora.php?id=21>
- <http://htmlbook.ru/html/%21doctype>
- <http://htmlbook.ru/samlayout/rezhimy-brauzerov/doktaip>
- [https://habrahabr.ru/post/71364/.](https://habrahabr.ru/post/71364/)

### Базовая структура html-документа

Итак, с DOCTYPE мы разобрались. Теперь рассмотрим, какие теги составляют основу html-файла.

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>First HTML Document</title>
</head>

<body>
    <!--      Здесь будет размещен основной код-->
</body>
</html>

```

Корневым тегом является `<html>`. Все остальные теги вкладываются в него, как матрешки. У этого тега принято указывать атрибут `lang`, который отвечает за язык, на котором создана данная html-страница. В примере `lang="en"`, т.к. английский является наиболее популярным языком в мире, да и текст между тегами `<title>` тоже английский.

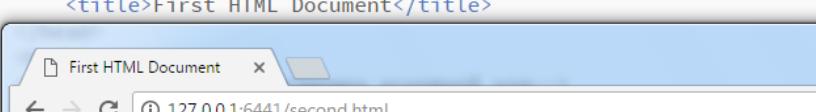
Если же вы создаете страницу с русским текстом, то значение этого атрибута должно быть «`ru`»:

Далее в структуре документа идет тег `<head>`, содержимое которого не отображается на странице. Тем не менее, это важный тег, выполняющий «служебные» функции. Во-первых, в нем указывается `<title>` документа, который отображается на вкладке в браузере.

```

3 ▼ <head>
4      <meta charset="windows-1251">
5      <title>First HTML Document</title>
6
7 ▼
8
9

```



Во-вторых, в нем располагаются meta-теги, которые выдают информацию для поисковиков и не только. Тег <met charset="UTF-8"> указывает кодировку документа, о которой мы поговорим отдельно. Сейчас замечу лишь то, что неправильное указание кодировки приведет к тому, что текст вашей страницы будет отображаться «кряко-зябрами», чего очень бы не хотелось:

## РўРµРі <head>

```
РўРµРі <head> РїСЂРµРіРпРРSP°Р РРSP°C‡РµРЅ Рр»СЏ С...СЂР°РРSPµРРSP‡СЏ РrСЂС‡РµРёС... СЌР»РµРјРµРSC.РsРl.Рt
C‡РµРѕСЂB РёРѕРѕСЂС... вЂ“ РїРѕРјРѕСЂСЂРє Р=СЂР°СРР РїСЂСЂРІ Р=СЂР°РР=РРС.Рµ С... РїР°РРSPSC.РРРе. РўР°РР=Рµ
РРPSC.СЂРµРёРѕРЅС.РµРНРРSPµСЂР° <head> РРSP°C... РsРrСlС.СfСl СРРµРµС... СЂР°РРSPµРРSPС.РsРlС.Р°С.РµРµРё, РёРѕС.РsСЂС.Рµ
РёСЂРµРµРРSP°Р РРSP°C‡РµРРSPРN° РrР»СЏР=СЂР°СРР РїСЂРµРІРё РїР°РРС.РtРµР°С... СЃРёСѓС.РµРј.
РkР°РtРcСЂРµРµС. РjРµС... Р°РРSPР РjС... РїР°РРС.РtРµР°С... СѓРёСѓС.РµРј РsР±СЂР°С%oР°СЂС.СfСl Рё
РµРС.Р°С.РµРРР°РР»СЏ РїСѓРµРРSPeСЏ РsРРeСѓРРР°РРSPeСЏ СѓР°РN°С.Р°, РёР»СЂС‡РµРІС.С... СѓР°Р»РsРlРё
РtСЂРµРёС... РrР°РРSPSC.С...
```

```
РЎРµРµРіСЂР°РР=РµРЅРµС.РµР° <head> РРSPµ РsС.РsР±СЂР°РР°РµС.СѓСl РРSP°РiСЂСlРjСtСh РРSP° РРµР±-
СѓС.СЂР°РРSPС.РtРµ. Р°Р° РёСѓРРeС...СЂСlРµРРSPeРµРjС.РµР° <title> СѓСfС.Р°РРSP°РР°РµРІР°РР°СtС%oРµРµРs
Р°Р°РРsР»РsРlРё РsРРeРРSP° РРµР±-СѓС.СЂР°РРSPeСt.
```

Такой вид страницы получился при указании кодировки в виде <meta charset="windows-1251"> тогда как документ был сохранен в кодировке utf-8.

- **Примечание:** в стандарте HTML5 упростились написание некоторых тегов, к которым, в том числе, относится и <meta>. Ранее (HTML 4.01) необходимо было указывать кодировку несколько более длинным способом:

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

А в XHTML еще и добавлять закрывающий слэш в конце:

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

Поэтому не удивляйтесь, если встретите такой синтаксис тега.

### Кодировка документа

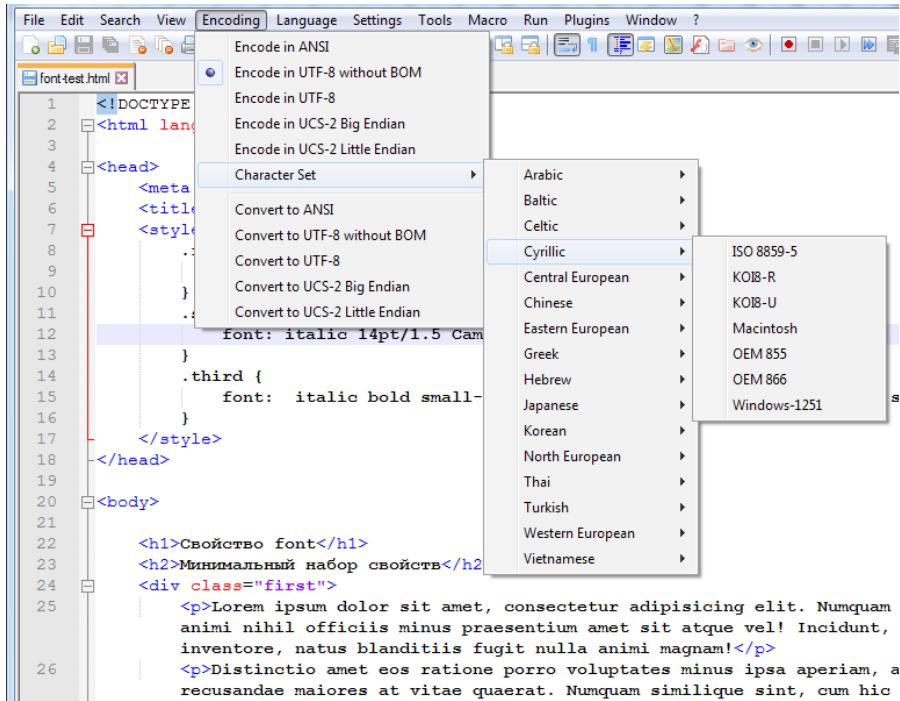
Дело в том, что в системе Windows по умолчанию для файлов введена кодировка символов в системе ASCII—*American Standard Code for Information Interchange* (американский стандартный код для обмена информацией), в которую включены латинские и русские символы, цифры, знаки препинания и др. В HTML ей соответствует кодировка windows-1251.

ASCII — это однобайтная кодировка, которая позволяет получить файл меньшего размера, но и по своим возможностям уступает двухбайтной кодировке UTF-8. Например, вам сложно будет создать на ней многоязычный сайт, который должен отображаться на русском, английском и, например, чешском языке. Ее возможностей, попросту не хватит, для отображения символов всех языков.

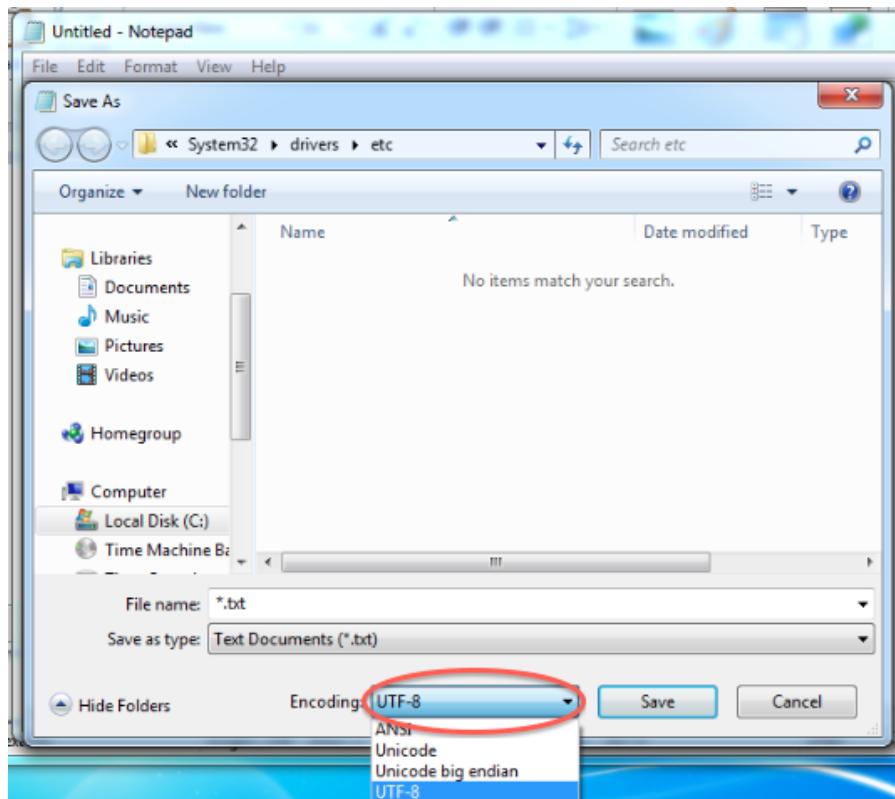
В этом смысле двухбайтная кодировка UTF-8 является более универсальной, т.к. вмещает в себя коды символов на большом количестве языков. Именно поэтому она стала самой популярной в настоящее время и является сейчас стандартом «де-факто» в мире веб-разработок.

Кроме указанных кодировок для русского языка есть еще ISO-8859-5 и KOI-8-R, но их настолько мало использу-

зуют, что нет смысла рассматривать их в данном курсе. Если вы установили на компьютер программу Notepad++, откройте ее и в меню Кодировки -> Кодировки -> Кириллица посмотрите доступные варианты.

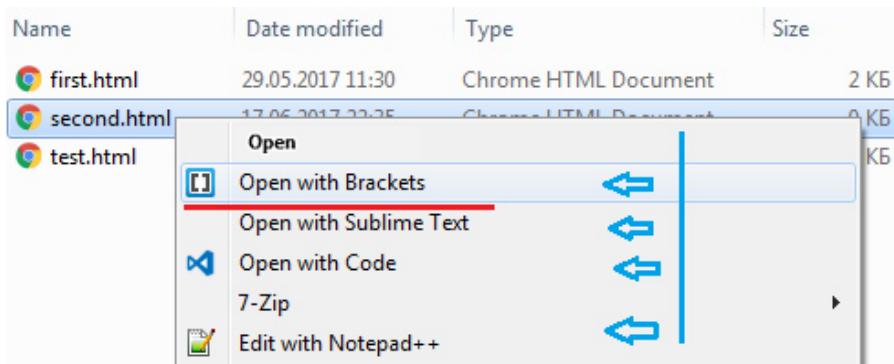


Чтобы посмотреть кодировку документа .txt, который мы рассматривали в качестве начального для создания html-файлов, нужно в программе «Блокнот» выбрать из меню «Файл» пункт «Сохранить» или «Сохранить как», если документ не изменился или был уже сохранен, и в нижней части окна настроек выбрать кодировку UTF-8 вместо ANSI.



Это самый простой способ, который доступен вам в системе Windows, т.к. программа «Блокнот» установлена там обычно по умолчанию. Но у него есть свои отрицательные стороны, которые не заметны в HTML, зато видны при создании и редактировании php-файлов, например для такой CMS, как Wordpress. Дело в том, что Блокнот добавляет метку BOM ([Маркер последовательности байтов](#) или метка порядка байтов — *Byte Order Mark*). Использование ее в шаблонах сайтов вызывает появление пустых строк в документе, поэтому ее следует удалять из файлов, предназначенных для веб (.html, .css, .php,.js).

Для этой цели лучше всего подойдет программа Notepad++, где в меню Кодировки предусмотрен пункт «Кодировка в UTF-8 без BOM» (см. скриншот выше). Для этого нужный файл нужно отредактировать с помощью Notepad++ (Edit with Notepad++) — опция, доступная из контекстного меню любого файла для веб.



## Тело документа — тег <body>

Тег `<body>` является тем местом, куда вы будете помещать основной html-код. Именно его содержимое является основным контентом страницы и выводится в браузер. Все, что будет находиться между открывающим `<body>` и закрывающим тегом `</body>`, также должно быть отформатировано в виде тегов.

## Комментарии в HTML

Комментарии в HTML не отображаются на странице. Весь текст, записанный между тегами

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

будет скрыт от посетителя вашего сайта.

Комментарии необходимы для того, чтобы указать, например, начало и конец форматирования какого-либо блока:

```

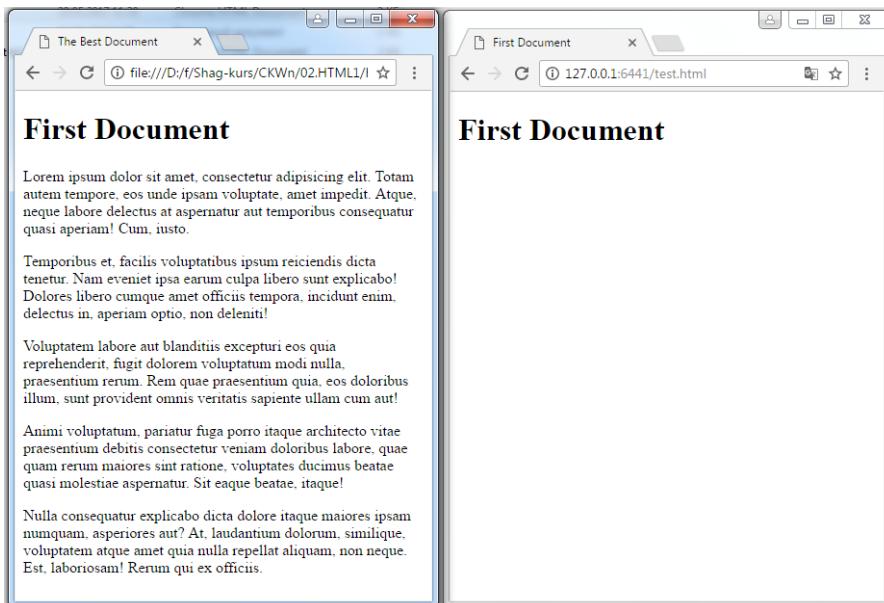
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="utf-8">
6      <title>First Document</title>
7  </head>
8
9  <body>
10     <h1>First Document</h1>
11     <!-- start #wrap -->
12  <div class="wrap">
13      <p>Lorem ipsum dolor sit amet, consectetur
           adipisicing elit. Totam autem tempore, eos unde ipsam
           voluptate, amet impedit. Atque, neque labore delectus
           at aspernatur aut temporibus consequatur quasi
           aperiam! Cum, iusto.</p>
14      <p>Temporibus et, facilis voluptatibus ipsum
           reiciendis dicta tenetur. Nam eveniet ipsa earum
           culpa libero sunt explicabo! Dolores libero cumque
           amet officiis tempora, incident enim, delectus in,
           aperiam optio, non deleniti!</p>
15  </div>
16  <!-- end #wrap -->
17 </body>
18
19 </html>
```

Или для того, чтобы этот блок спрятать:

```

9  <body>
10     <h1>First Document</h1>
11  <!--
12  <div class="wrap">
13      <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>
14      <p>Temporibus et, facilis voluptatibus ipsum
           reiciendis dicta tenetur. Nam eveniet ipsa earum
           culpa libero sunt explicabo! Dolores libero cumque
           amet officiis tempora, incident enim, delectus in,
           aperiam optio, non deleniti!</p>
15  </div>
16  -->
17 </body>|
```

На скриншотах видно, что весь текст, помещенный в теги <!-- --> становится светло-серого цвета в Brackets — как бы частично пропадает. Если сравнить 2 варианта отображения этой страницы с различным расположением комментариев, то увидим мы в браузере следующее:



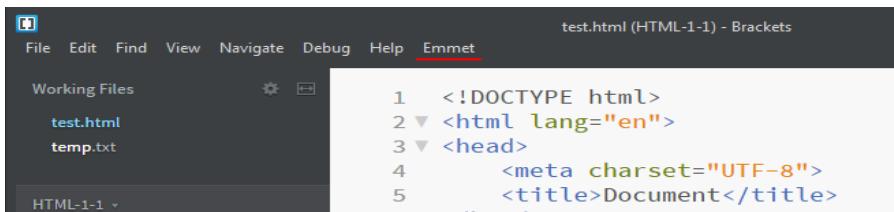
Там, где в комментарии попали именно комментарии начала и конца блока, мы не видим только этих слов. А в примере с комментированием самого блока — исчез весь текст после заголовка.

Следует заметить, что в Brackets (как и в других редакторах кода, впрочем) есть быстрые клавиши для добавления комментариев — это **CTRL+ /**. Если вы хотите закомментировать одну строку, достаточно установить в любом ее месте курсор и нажать **CTRL+ /**. Комментарии автоматически добавятся в начале и в конце данной

строки. Для комментирования нескольких абзацев, необходимо будет сначала их выделить, а уже потом нажать **CTRL+ /**.

## Использование плагина Emmet для создания структуры документа

Если вы установили рекомендуемый в начале этого урока [набор плагинов](#) для Brackets, то сейчас самое время воспользоваться возможностями плагина Emmet. Проверить, что он установлен у вас в редакторе очень просто — посмотрите на строку меню Brackets — если в конце вы видите пункт Emmet — значит, все в порядке.



Если не видите — очень рекомендую вам эту ситуацию исправить, потому что код базовой структуры набирается с помощью Emmet супер просто:

! и клавиша Tab.

## Теги заголовков и абзацев

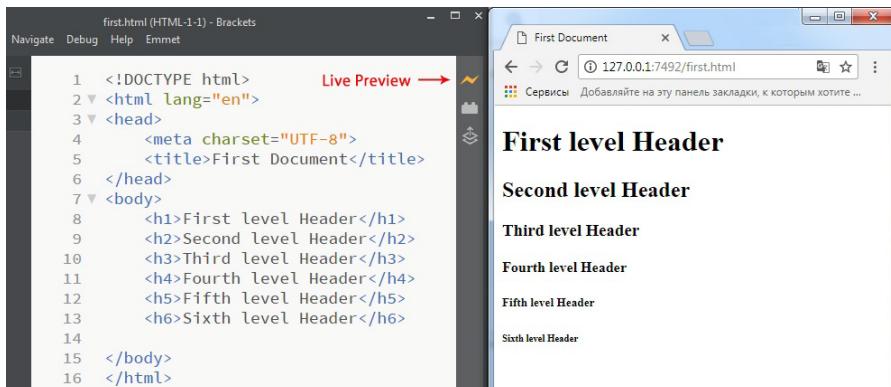
Заголовки — это обязательная часть html-документа. Есть даже 2 новых тега в HTML5, которые без заголовков не пройдут валидацию — это `<article>` и `<section>`. Но их мы будем рассматривать позже.

В HTML предусмотрено **6 уровней заголовков**, которые отличаются друг от друга размером шрифта. Обозначаются заголовки так:

```
<h1>Заголовок 1-го уровня</h1>
<h2>Заголовок 2-го уровня</h2>
<h3>Заголовок 3-го уровня</h3>
<h4>Заголовок 4-го уровня</h4>
<h5>Заголовок 5-го уровня</h5>
<h6>Заголовок 6-го уровня</h6>
```

Буква **h** в теге обозначает, что это заголовок ( от английского *heading*), а цифра — какой уровень заголовка. Чем больше цифра — тем меньше важность и размер шрифта заголовка.

Посмотрим, как это выглядит в редакторе кода и в браузере:



Разница между заголовками разных уровней, по-моему, очевидна. `<h1>` — самый крупный и заметный, а `<h6>` — уже почти нечитаемый. Собственно, и вес этих заголовков в глазах поисковиков тоже различен. Как правило, заголовков типа `h1` на странице должен быть один или два — это название компании и название страницы. Еще желательно, чтобы текст в `h1`, определяющим название страницы, совпадал с содержимым тега `<title>` в блоке `<head>`.

Все остальные заголовки вы можете использовать по усмотрению, необязательно в порядке возрастания их уровня, но все же желательно, чтобы за `h1` шел заголовок `h2`, т.к. он тоже важен для ранжирования вашей страницы.

- **Примечание 1:** на скриншоте стрелочкой показана кнопка в виде молнии в правом верхнем углу Brackets. Это **Live Preview** (Живой предпросмотр) в браузере Google Chrome, «вшиитый» в редактор кода. Нажатие этой кнопки при редактировании html-документа позволяет в реальном времени просмотреть все изменения, происходящие в редакторе. Чтобы это увидеть, имеет смысл расположить окно редактора и окно браузера рядом, как на скриншоте вверху.

Когда включен Live Preview, в браузере синей рамкой выделен элемент, на котором сейчас расположен курсор (на скриншоте это `h2`). Если вы будете перемещать курсор выше или ниже, например, клавишами  $\uparrow$  или  $\downarrow$ , то увидите, как перемещается рамка с одного элемента на другой.

- **Примечание 2:** если в Brackets вам приходится создавать блоки с похожим или идентичным содержанием, имеет смысл воспользоваться сочетанием клавиш **CTRL + D** (от англ. *duplicate* — дублировать). Если курсор находится в любом месте строки, то будет продублирована вся строка. Если вы выделите часть слов или несколько строк,

*то продублируется весь выделенный контент. Поверьте, это бывает очень удобно.*

Что еще касается заголовков, то вам следует знать, что все они являются блочными элементами, которые по умолчанию в браузерах выделяются жирным шрифтом и имеют отступы до и после составляющего их текста.

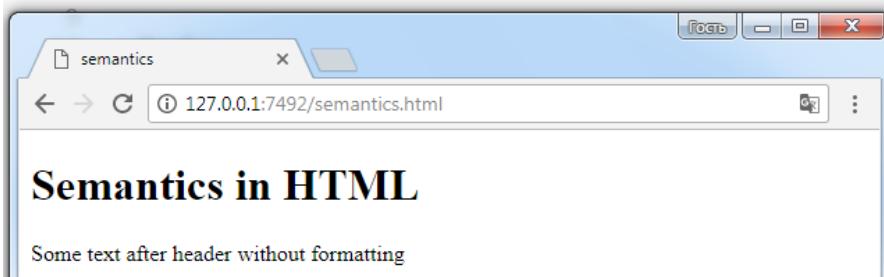
Нужно еще понимать, что нельзя всю страницу заполнить только заголовками — это будет неверно с точки зрения семантики html-документа. И поисковикам тоже вряд ли понравится. Вдумайтесь — разве может книга или статья в журнале или в газете состоять из одних заголовков? Естественно, не может. После заголовка должен обязательно идти основной текст, который в печатной прессе обычно разбивается на абзацы. Рассмотрим, как создавать их в html.

## **Что такое блочные элементы?**

Блочными называются такие элементы HTML, которые по умолчанию занимают все доступное пространство внутри браузера или родительского элемента, даже если их содержимое совсем небольшое. Т.е. даже при 2-3 словах внутри заголовка он будет занимать все пространство до правой границы браузера, и текст, размещенnyй рядом с закрывающим тегом, будет перенесен на следующую строку.

На скриншоте в примере видно, что заголовок h1, выделенный в Brackets в режиме Live Preview, обведен любой рамкой и занимает все место слева направо в браузере. А текст за ним перенесен на следующую строку.

```
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
```



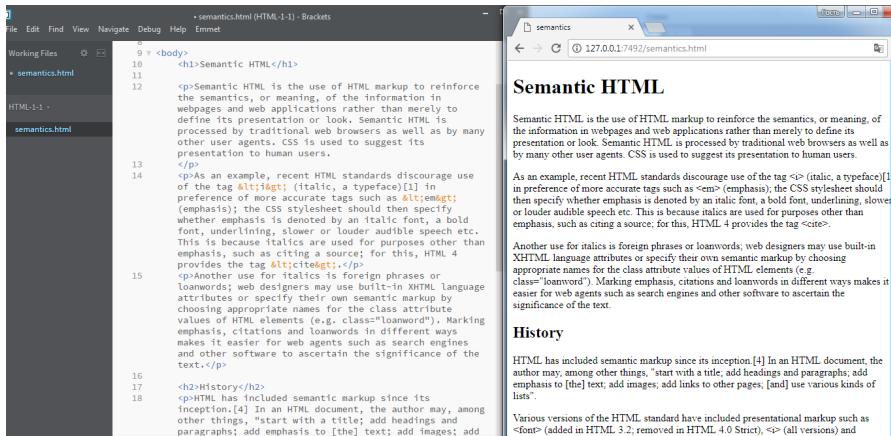
## Абзацы в HTML

Как и в печатных изданиях, абзацы являются основными элементами для форматирования текста, который составляет статью. Для создания абзацев предназначен тег `<p></p>` (от англ. *paragraph*).

Вы найдете вариант разбиения текста на абзацы в файле `semantic.html`, в котором в качестве текста был использован отрывок [статьи из Википедии](#).

На скриншоте видно, что текст, помещенный в абзацы, выводится в браузере обычным начертанием (а не жирным, как у заголовков), но при этом каждый абзац имеет отступы сверху и снизу. Это визуально отделяет один смысловой блок от другого и помогает легче воспринимать суть текста.

# Структура HTML-файла. DOCTYPE



```
File Edit Find View Navigate Debug Help Emmet
Working Files semantics.html (HTML-1.1) - Brackets
+ semantics.html
HTML-1.1
semantics.html
  6 <html>
  7   <head>
  8     <title>Semantic HTML</title>
  9   </head>
 10  <body>
 11    <h1>Semantic HTML</h1>
 12    <p>Semantic HTML is the use of HTML markup to reinforce the semantics, or meaning, of the information in webpages and web applications rather than merely to define its presentation or look. Semantic HTML is processed by traditional web browsers as well as by many other user agents. CSS is used to suggest its presentation to human users.</p>
 13    <p>As an example, recent HTML standards discourage use of the tag <i><em></em></i> (italic, a typeface)[1] in preference of more accurate tags such as <em>[1]<em> (emphasis); the CSS stylesheet should then specify whether emphasis is denoted by an italic font, a bold font, underlining, a larger font, a smaller font etc. This is because italics are used for purposes other than emphasis, such as citing a source; for this, HTML 4 provides the tag <em><em></em></em>[2].</p>
 14    <p>Another use for <i><em></em></i> is foreign phrases or loanwords. Web designers may use built-in HTML language attributes or specify their own semantic markup by choosing appropriate names for the class attribute values of HTML elements (e.g. class="loanword"). Marking emphasis, citations and loanwords in different ways makes it easier for web agents such as search engines and other software to ascertain the significance of the text.</p>
 15    <h2>History</h2>
 16    <p>HTML has included semantic markup since its inception.[4] In an HTML document, the author may, among other things, start with a title; add headings and paragraphs; add emphasis to [the] text; add images; add links to other pages; [and] use various kinds of lists.</p>
 17    <h2>Presentations</h2>
 18    <p>Various versions of the HTML standard have included presentational markup such as <font>[added in HTML 3.2; removed in HTML 4.0 Strict], <big>[all versions], <small>[...]
```

Вы можете посмотреть видео-урок про базовую структуру html-документа, теги заголовков и абзацев на моем блоге по ссылке <http://html-plus.in.ua/base-html-structure/>

## Несколько слов об атрибутах

Для любого тега можно задать атрибуты — дополнительные параметры в виде пар атрибут= "значение", которые в некоторой степени выделяют его среди других таких же. Атрибуты записываются только в открывающем теге и отделяются от названия тега и друг от друга пробелами. В общем случае это выглядит так:

```
<элемент атрибут1='значение' атрибут2='значение'
атрибут3='значение'>Текст элемента</элемент>
```

Пример:

```
<h2 title='Статья о пользе витаминов для детей'
id='article1' class='article-header'> Витамины для
детей </h2>
```

В этом примере для заголовка 2-го уровня использованы универсальные атрибуты title, id и class, т.е. такие, которые можно добавить для любого элемента. Еще к универсальным относятся tabindex, data-атрибуты и некоторые другие, редко используемые, например, contenteditable, hidden или contextmenu.

Но также существуют атрибуты, которые характерны только для определенных тегов. Например, для встраивания изображения нужен тег с атрибутами src и alt, которые встречаются характерны только для него (хотя src нужен еще для таких тегов, как <script> и <iframe>):

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

При добавлении атрибутов нужно соблюдать несколько простых правил:

1. Атрибуты отделяются друг от друга пробелами.
2. Значения атрибутов записываются в двойных или одинарных кавычках.
3. Если внутри значения атрибута необходимо использовать кавычки или апостроф, необходимо сочетать оба вида кавычек с учетом их вложенности:

```
<p title='Абзац из книги Б. Акунина "Азазель"'>
```

Здесь значение атрибута title взято в одинарные кавычки, а внутри использованы двойные для названия книги. Этот атрибут добавляет всплывающую подсказку к элементу, которая появляется при наведении на элемент в браузере.

The screenshot shows the Brackets IDE interface with the file 'study-scarlet.html' open. The code is as follows:

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Document</title>
6  </head>
7  <body>
8      <h1>PART ONE</h1>
9
10     <h2>Dr Watson Remembers</h2>
11     <h2>CHAPTER ONE</h2>
12     <h3>Introductions</h3>
13
14     <p>In the year 1878 I became a doctor of medicine at the University of London, and then joined the Army as a surgeon. My first job was in Afghanistan, 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>
15     <p>My health was very poor when I left the hospital, I was very weak and I had no friends or family in England. The Government gave me a small allowance for each day and with this money I lived well enough for a while in a hotel in London. But it soon became too expensive and I needed find somewhere cheaper to live. That how I met Mr Sherlock Holmes.</p>
16     <p>One day, I left the hotel and by chance I met a young man I knew called Stamford.</p>
```

The browser window shows the rendered content with the title 'Document'. The page has a main heading 'PART ONE', a chapter heading 'CHAPTER ONE', and a section heading 'Introductions'. The text content describes Dr. Watson's experiences in the Army and his meeting with Mr. Stamford.

4. Если значение атрибута можно представить в виде логической величины true (истина) или false (ложь), или по другому «есть» или «нет», то записывать его можно в нескольких вариантах, которые равнозначны:

```
<input type="checkbox" checked>
<input type="checkbox" checked="">
<input type="checkbox" checked="checked">
```

► **Примечание:** Все браузеры настроены на максимально правильное отображение тегов и их атрибутов, поэтому включают атрибут даже при наличии в нем значений вроде true или 1, которые на самом деле недопустимы. Лучше все-таки избегать таких вариантов, т.к. они противоречат спецификации HTML5.

А теперь поговорим об устаревших атрибутах. Например, атрибут align позволяет выравнивать текст в абзаце, заголовке или в другом блочном элементе по левому или правому краю, по ширине элемента или по центру

(align="left", или align="right", или align="justify", или align="center"). Вы наверняка найдете в Интернете кучу руководств, которые рекомендуют использовать именно этот атрибут для выравнивания текста.

Только вот со стандартом HTML5 этот атрибут несовместим. И валидатор выдаст следующее сообщение: «Атрибут align для элемента p устарел. Вместо него используйте CSS»



Так что использовать устаревшие атрибуты не стоит. HTML — очень популярный язык разметки. Он постоянно совершенствуется и развивается. С каждым новым стандартом пересматриваются подходы к верстке. Часть тегов или атрибутов переходит в разряд устаревших и не рекомендуется к использованию. Мы будем заменять их CSS-стилями, как и рекомендует валидатор. А если вы все-таки сделали это, проверьте свой документ на [валидность](#).

### Что такое Lorem Ipsum?

Дело в том, что далеко не всегда в вашей практике придется форматировать готовый текст, предоставленный заказчиком. К сожалению, заказчики с ним довольно часто запаздывают. Кроме того, вы можете создавать шаблоны сайтов, о реальном содержании которых на этапе разработки можете только догадываться. Поэтому вам наверняка пригодится шаблонный текст, который начинается словами *Lorem ipsum*. Его, кстати, еще называют «текст-рыба».

- **Цитата из Википедии:** Lorem ipsum — классическая панграмма, условный, зачастую бессмысленный текст-заполнитель, вставляемый в макет страницы. Используется в качестве заполнителя по крайней мере с XVI века[1]. Является искажённым отрывком из философского трактата Марка Туллия Цицерона «О пределах добра и зла», написанного в 45 году до н. э. на латинском языке, обнаружение сходства атрибутируется Ричарду МакКлентоку[1].

Т.е. это текст, который помогает заполнить сайт контентом, но при этом смысла не имеет. Это очень удобно, кроме всего прочего, на этапе обучения. Можно заполнять с помощью Lorem ipsum любые тестовые элементы (мы уже знаем о заголовках и абзацах), не задумываясь о количестве и содержании текста.

При соответствующем запросе Google предоставляет доступ к массе генераторов Lorem ipsum:

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

Но у нас с вами есть намного более удобный инструмент для создания такого текста, причем сразу «обернутого» в теги. И это ... опять плагин Emmet.

Для того чтобы получить абзац с текстом-рыбой, сразу после открывающего тега <body> или после уже существующего в нем другого тега, наберите аббревиатуру

```
p>lorem
```

и нажмите клавишу Tab.

И вуаля, — у вас готов абзац с текстом-заполнителем:

```
<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>
```

По умолчанию, Emmet добавит 30 слов текста. Но вы можете разнообразить количество слов, добавляя нужную цифру после lorem:

```
p>lorem10
```

Получим:

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

Необязательно использовать lorem только с тегом p. Вы можете применить его для заголовка любого уровня:

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

И так далее...

- **Примечание:** плагин Emmet использует абреквиатуры — последовательности символов, которые

*позволяют вывести теги, атрибуты и текст в определенной последовательности и с определенными уровнями вложенности.*

Подробнее об аббревиатурах:

1. <http://webdesign-master.ru/blog/html-css/2.html>
2. <http://html-plus.in.ua/formatirovanie-teksta-s-pomoshhyu-emmet/>
3. <http://ts-soft.ru/blog/emmet>
4. <http://webtoks.ru/web/vvedeny-emmet/>

Важно понимать, что внутри аббревиатуры не должно быть пробелов (исключение — текст внутри фигурных скобок). А в самом конце обязательно нужно нажать клавишу **Tab**, чтобы раскрыть аббревиатуру и получить отформатированный по правилам html текст. Опять-таки после текста аббревиатуры не должно быть пробела, и курсор должен находиться именно в конце, а не где-нибудь в середине набранного текста.

Давайте рассмотрим несколько более сложную аббревиатуру. На этот раз с частично осмысленным текстом:

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

Получим следующую структуру:

```
<h1> First heading </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 incident voluptas consequuntur
    iste repellat, quasi quos provident, nostrum, a.
    Repellendus aspernatur, veritatis ea cum aliquid
    architecto!
</p>

```

Если расшифровать аббревиатуру, то можно увидеть следующее: h1 с текстом «Firstheading» размещается рядом с заголовком h2 с текстом «Secondheading», а затем идут подряд 3 абзаца с шаблонным текстом по 20 слов в каждом. Причем текст в абзацах различен! Заметьте, что текст, размещенный в фигурных скобках, отобразился именно в том виде, в котором был набран.

Удобно, не так ли?

Несколько сложнее будет выглядеть аббревиатура с шаблонным текстом в заголовках и абзацах. В ней придется использовать символ ^ — выход в родительский элемент, которым у нас пока является body.

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

Выглядит пугающе, но при нажатии в конце на клавишу **Tab** принимает вполне читабельный вид:

```

<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>

```

Рассмотрим подробно, что же в этой аббревиатуре написано:

```

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

```

- **h1>lorem4** — заголовок h1 с текстом-рыбой из 4-х слов
- ^ — из текста заголовка h1 возвращаемся в body
- **h2>lorem5** — заголовок h2 с текстом-рыбой из 5 слов
- ^ — из текста заголовка h2 возвращаемся в body
- **p\*2>lorem15** — 2 абзаца с Lorem ipsum из 15 слов
- ^ — из текста абзацев возвращаемся в body
- **h3>lorem4** — заголовок h3 с текстом-рыбой из 4 слов
- ^ — из текста заголовка h2 возвращаемся в body
- **p\*3>lorem10** — 3 абзаца с Lorem ipsum из 10 слов.

Так будет выглядеть документ в браузере.



The screenshot shows a browser window with the title "Lorem ipsum Text". The address bar displays the URL "127.0.0.1:1362/loremipsum-emmet.html". The page content is as follows:

# **Lorem ipsum dolor sit.**

## **Lorem ipsum dolor sit amet.**

  Lorem ipsum dolor sit amet, consectetur adipisicing elit. Aperiam ex et sapiente suscipit nulla nihil.

  Sed eius ratione rem sit quas maiores praesentium explicabo assumenda hic, delectus animi, qui inventore'

### **Lorem ipsum dolor sit.**

  Lorem ipsum dolor sit amet, consectetur adipisicing elit. Optio, eaque.

  Reiciendis ea iure officiis, ullam debitis repellat provident nihil commodi.

  Distinctio veniam consectetur aspernatur tenetur facere. Totam quo enim, numquam.

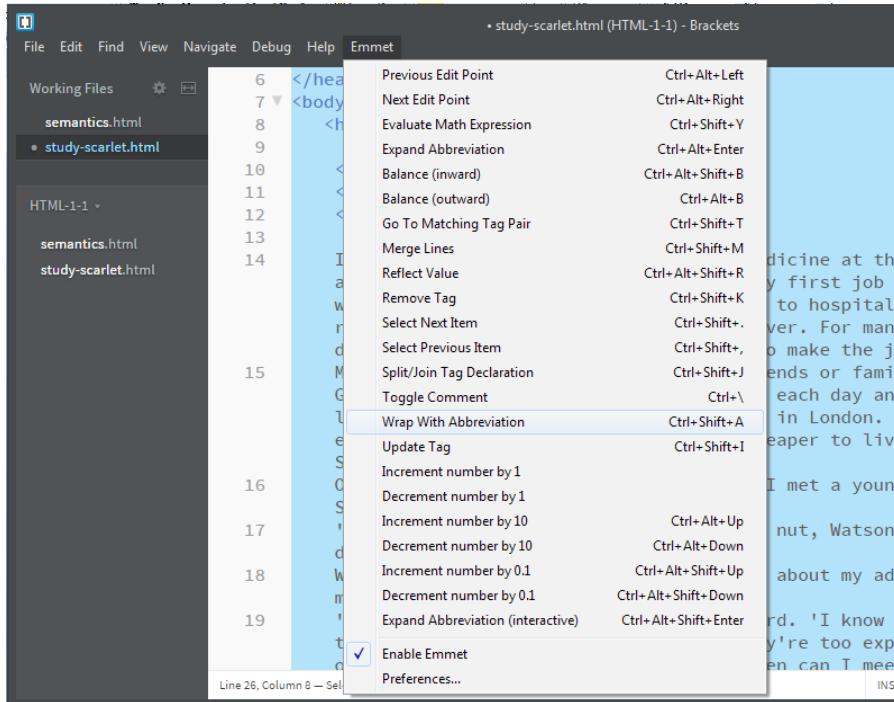
На рисунке ниже показана примерная схема работы аббревиатуры с упором на знак ^.

```
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="utf-8">
6      <title>Lorem ipsum Text</title>
7  </head>
8
9  <body>
10 <!--     h1>lorem4^h2>lorem5^p>2>lorem15^h3>lorem4^p>3>lorem10-->
11
12      <h1>Lorem ipsum dolor sit.</h1>
13      <h2>Lorem ipsum dolor sit amet.</h2>
14      <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Aperiam ex et sapiente
15      suscipit nulla nihil.</p>
16      <p>Sed eius ratione rem sit quas maiores praesentium explicabo assumenda hic,
17      delectus animi, qui inventore'</p>
18      <h3>Lorem ipsum dolor sit.</h3>
19      <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Optio, eaque.</p>
20      <p>Reiciendis ea iure officiis, ullam debitis repellat provident nihil commodi.</p>
21      <p>Distinctio veniam consectetur aspernatur tenetur facere. Totam quo enim, numquam.
22      </p>
23
24  </body>          body - parent element
25
26  </html>
```

Попробуйте раскрыть аббревиатуру сами в файле [loremipsum-emmet.html](#).

## Обертки из аббревиатур Emmet

Есть еще одна замечательная функция в плагине Emmet. Называется она *Wrap with Abbreviation* (**CTRL + SHIFT + A**). Вы можете найти соответствующий пункт в меню Emmet в Brackets, но я, все же, рекомендую вам запомнить сочетание клавиш.



Пользоваться «обертками» крайне просто — выделяете нужный текст, нажимаете **CTRL + SHIFT + A** — и вводите тег или аббревиатуру Emmet. Смотрите на результат и обязательно нажимаете **Enter**, иначе аббревиатура не применится.

Например, так будет выглядеть обрамление текста в заголовок первого уровня.

## Урок 1. Введение. Структура HTML

```
6  </head>
7  <body>
8    <h1>PART ONE</h1>
9
10   Dr Watson Remembers
11   CHAPTER ONE
12   Introductions
13
14   In the year 1878 I became a doctor of medicine at the University of London,
15   and then joined the Army as a surgeon. My first job was in Afghanistan,
16   where I was shot in the shoulder. I went to hospital and started to
17   recover, but then I became ill with a fever. For many months I was close to
18   death, but finally I was strong enough to make the journey back to England.
19   My health was very weak and I had no friends or family in England. The
20
21   Enter Abbreviation: h1
```

Если выделить весь текст, разбитый на блоки с помощью клавиши **Enter**, то можно очень быстро превратить его в целый ряд абзацев:

```
11  <h2>CHAPTER ONE</h2>
12  <h3>Introductions</h3>
13
14  <p>In the year 1878 I became a doctor of medicine at the University of London,
15  and then joined the Army as a surgeon. My first job was in Afghanistan, where I
16  was shot in the shoulder. I went to hospital and started to recover, but then I
17  became ill with a fever. For many months I was close to death, but finally I was
18  strong enough to make the journey back to England.</p>
19  <p>My health was very weak and I had no friends or family in England. The
20  Government gave me a small allowance for each day and with this money I lived
21  well enough for a while in a hotel in London. But it soon became too expensive
22  and I needed find somewhere cheaper to live. That how I met Mr Sherlock Holmes.
23  </p>
24  <p>One day, I left the hotel and by chance I met a young man I knew called
25  Stamford.</p>
26  <p>'Why, you're very thin and as brown as a nut, Watson. What have you been
27  doing?'</p>
28  <p>We went to lunch together and I told him about my adventures and my current
29  money problems.</p>
30  <p>'I may be able to help you,' said Stamford. 'I know a man who needs someone to
31  share some nice rooms he's found. They're too expensive for him on his own.' 'I'm
32  just the right man for him. When can I meet him?' Stamford gave me a strange
33  look. 'You don't know Sherlock Holmes yet,' he said. 'You might not get on with
34  him.' 'But why?' I asked. I was extremely curious about him.</p>
35  <p>'He's a decent man,' said Stamford, 'but he has some strange ideas. He studies
36  many different things; medicine, science, anatomy... but apparently with no
```

Аббревиатура будет простой: **p\***.

Так же просто обернуть весь текст в теги структуры html-документа. Достаточно выделить весь текст (**CTRL + A**), вызвать поле «*Enter Abbreviation*» с помощью **CTRL + SHIFT + A** и ввести восклицательный знак:

The screenshot shows the Brackets IDE interface with the following details:

- File Menu:** File, Edit, Find, View, Navigate, Debug, Help, Emmet.
- Working Files:** semantics.html, study-scarlet.html
- Project:** HTML-1-1
- Code Editor:** Content of study-scarlet.html file:

```
1 <!DOCTYPE html>
2   <html lang="en">
3     <head>
4       <meta charset="UTF-8">
5       <title>Document</title>
6     </head>
7     <body>
8       <h1>PART ONE</h1>
9
10      <h3>Dr Watson Remembers</h3>
11      <h2>CHAPTER ONE</h2>
12      <h3>Introductions</h3>
13
14      <p>In the year 1878 I became a doctor of medicine at the University of London, and then joined the Army as a surgeon. My first job was in Afghanistan, 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>
15      <p>My health was very weak and I had no friends or family in England. The Government gave me a small allowance for each day and with this money I lived well enough for a while in a hotel in London. But it soon became too expensive and I needed find somewhere cheaper to live. That how I met Mr Sherlock Holmes.</p>
16      <p>One day, I left the hotel and by chance I met a young man I knew called Stamford.</p>
17    </body>
18  </html>
```
- Bottom Bar:** Enter Abbreviation input field, Line 5, Column 24 — Selected 8 columns — 18 Lines, INS, UTF-8, HTML, Spaces: 4.

► **Примечание:** будьте внимательны, когда вводите аббревиатуры, т.к. Emmet создаст вам в том числе и такой тег, который вы ввели, но которого нет в спецификации HTML.

Вы можете сами попробовать обертки Emmet, используя файл a-study-in-scarlet-conan-doyle.txt, который прикреплен к PDF-файлу данного урока. В нем использован текст с сайта [english-e-books](#).

### Вложенные теги

Чтобы выделить часть текста жирным или курсивным шрифтом, можно использовать такие теги, как **<b>** или *<i>*:

study-scarlet.html (HTML-1-1) - Brackets

File Debug Help Emmet

```

12     <h3>Dr Watson Remembers</h3>
13     <h2>CHAPTER ONE</h2>
14     <h3>Introductions</h3>
15
16     <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>
17     <p>My health was very weak and I had no
        friends or family in England. The Government
        gave me a small allowance for each day and
        with this money I lived well enough for a
        while in a hotel in London. But it soon became

```

Document

INTRODUCTIONS

In the year 1878 I became a doctor of medicine at the University of London, and then joined the Army as a surgeon. My first job was in Afghanistan, 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.

My health was very weak and I had no friends or family in England. The Government gave me a small allowance for each day and with this money I lived well enough for a while in a hotel in London. But it soon became too expensive and I needed find somewhere cheaper to live. That how I met Mr Sherlock Holmes.

One day, I left the hotel and by chance I met a young man I knew called Stamford.

```

<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>

```

Тег **<b>** происходит от слова **bold** (*жирный*), а *<i>* — от слова **italics** (*курсив*), соответственно использование этих тегов приводит к выделению текста между ними с помощью жирного или курсивного начертания. Эти элементы относят к группе тегов физического форматирования.

В отличие от блочных тегов заголовков и абзацев эти теги являются строчными (еще их называют линейными от английского их наименования *inline*).

Поэтому оборачивают в такие теги не все предложение или блок текста, а какую-то его часть, например, 2-3-4 слова, как в примере на скриншоте.

Таким же образом работают еще 2 тега — **<strong>** и **<em>**, т.е. **<strong>** выделяет текст жирным, а **<em>** — курсивом, но эти 2 элемента относятся к группе тегов логического форматирования. В контексте разметки страницы они выполняют роль «усилителя внимания», т.е. призваны показать (в большей степени поисковикам, чем посетителям), что выделенный в них текст важен для пользователя.

Но и теги физического, и теги логического форматирования находятся внутри блочных элементов (в примере это абзац — **<p>**), т.е. являются вложенными или дочерними по отношению к абзацу. А абзац по отношению к вложенным тегам является родительским элементом.

Для вложенных тегов важно соблюдать «правило матрешки»: тег, который открыт самым первым, должен быть закрыт самым последним:

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

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

Вложенные теги могут быть не только строчными. И уровень вложения может быть достаточно глубоким. Но это правило все равно остается прежним — тег, который открыт самым первым, должен быть закрыт самым последним.

### Теги div и span

Как вы понимаете, из заголовков и абзацев не построишь полноценную страницу. Поэтому в спецификации есть еще масса тегов, с которыми мы должны познакомиться. И одними из самых используемых элементов являются **div**-ы (от англ. *division* — раздел). Это блочные элементы, которые, в отличие от абзацев не имеют отступа сверху и снизу. Это элементы, из которых строится внутренняя структура сайта.

Код их очень простой:

```
<div>текст</div>
```

Давайте используем простую Emmet-аббревиатуру:

```
div*4>lorem40
```

Получим такой файл:

The screenshot shows the Brackets IDE interface. On the left, the code editor displays the HTML file 'test-div.html' with the following content:

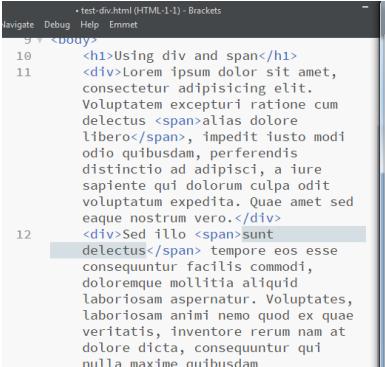
```
test-div.html (HTML-1-1) - Brackets
Navigate Debug Help Emmet
5 <title>Test DIV</title>
6 </head>
7 <body>
8 <div>Lorem ipsum dolor sit amet,
consectetur adipisicing elit.
Voluptatem excepturi ratione cum
delectus alias dolore libero, impedit
iusto modi odio quibusdam,
perferendis distinctio ad adipisci, a
iure sapiente qui dolorum culpa odit
voluptatum expedita. Quae amet sed
eaque nostrum vero.</div>
9 <div>Sed illo sunt delectus tempore
eos esse consequuntur facilis
commodi, doloremque mollitia aliquid
laboriosam aspernatur. Voluptates,
laboriosam animi nemo quod ex qua
veritatis, inventore rerum nam at
dolore dicta, consequuntur qui nulla
maxime quibusdam reprehenderit in
quam illum vel sit!</div>
10 <div>Aliquid animi veniam similique
repudiandae, porro soluta accusantium
eos cumque inventore id, ipsam
possimus, modi ea! Fuga autem
praesentium fugiat esse numquam, nesciunt, illo
natus assumenda dignissimos, dolore at, possimus animi! Placeat ipsum blanditiis minus,
aut tenetur non doloremque,
quisquam.</div>
```

On the right, a browser window titled 'Test DIV' shows the rendered content of the HTML. The page contains two large paragraphs of placeholder text ('Lorem ipsum' and 'Sed illo') enclosed in `<div>` tags.

Визуально тег `<div>` не слишком-то замечательно выглядит — масса текста без особого форматирования. Но этим он и хорош — для него можно задать то **css**-форматирование, которое необходимо для данной конкретной ситуации. И обычно для div-ов задают атрибут `class`, который и позволяет разнообразить внешний вид этих элементов. Но об этом чуть позже, когда будем рассматривать стили `css`.

Также очень часто внутри текста используют теги `<span></span>` (от англ. *интервал*). Это строчные теги, которые предназначены для объединения небольшого количества текста с целью задания для него общего форматирования. Практически обязательным в этом случае для тега `span` является атрибут `class`. Именно он позволяет сделать форматирование для различных `<span>` разнообразным с помощью правил `css`.

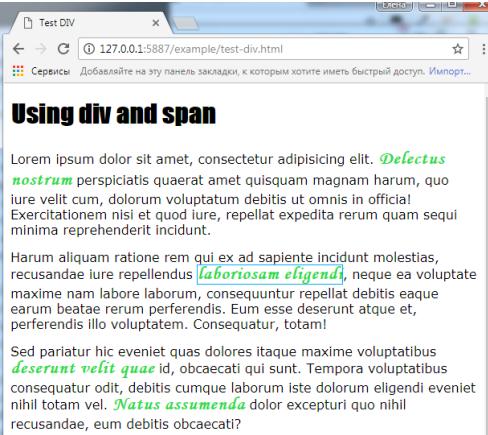
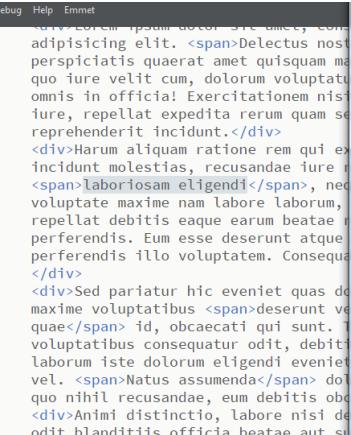
## Урок 1. Введение. Структура HTML



```
• test-div.html (HTML 1-1) - Brackets
navigate Debug Help Emmet
9 <body>
10   <h1>Using div and span</h1>
11   <div>Lorem ipsum dolor sit amet,
12     consectetur adipisicing elit.
13     Voluptatum excepturi ratione cum
14     delectus <span>sunt</span>
15     libero</span>, impedit iusto modi
16     odio quibusdam, perferendis
17     distinctio ad adipisci, a iure
18     sapiente qui dolorum culpa odit
19     voluptatum expedita. Quae amet sed
20     eaque nostrum vero.</div>
21   <div>Sed illo <span>sunt
22     delectus</span> tempore eos esse
23     consequuntur facilis commodi,
24     doloremque mollitia aliquid
25     laboriosam aspernatur. Voluptates,
26     laboriosam animi nemo quod ex quae
27     veritatis, inventore rerum nam at
28     dolore dicta, consequuntur qui
29     nulla maxime quibusdam
```

На скриншоте выше видно, что выделенный текст находится в тегах `<span>`. Но визуально он никак не отличается от текста ДО и ПОСЛЕ него.

Но если вы раскомментируете код в тегах `style` в файле `test-div-span.html` ([прикреплен к PDF-файлу данного урока](#)), то внешний вид файла при отсутствии изменений в html-разметке сразу преобразится:



```
• test-div-span.html (HTML 1-1) - Brackets
navigate Help Emmet
1 <h1>Using div and span</h1>
2 <div>Lorem ipsum dolor sit amet, consectetur adipisicing elit. <span>Delectus noster</span> perspicatis quaerat amet quisquam maiores qui iure velit cum, dolorum voluptatum omnis in officia! Exercitationem nisi iure, repellat expedita rerum quam sequi reprehenderit incidentum.</div>
3 <div>Harum aliquam ratione rem qui excepturi sunt molestias, recusandae iure <span>laboriosam eligendi</span>, neque ea voluptate maxime nam labore laborum, repellat debitis eaque earum beatae perferendis. Eum esse deserunt atque perferendis illo voluptatem. Consequatur id, obcaecati qui sunt. <span>Natus assumenda</span> doloremque nihil recusandae, eum debitis obcaecati qui sunt. <span>Delectus noster</span> perspicatis quaerat amet quisquam magnam harum, quo iure velit cum, dolorum voluptatum debitis ut omnis in officia! Exercitationem nisi et quod iure, repellat expedita rerum quam sequi minima reprehenderit incidentum.
4 <div>Sed pariatur hic eveniet quas dolores itaque maxime voluptatibus deserunt vel quae</span> id, obcaecati qui sunt. Tempora voluptatibus consequatur odit, debitis cumque labore laborum iste dolorum eligendi eveniet nihil totam vel. <span>Natus assumenda</span> dolor excepturi quo nihil recusandae, eum debitis obcaecati qui sunt.
```

В примере показано однотипное форматирование для всех `<span>` в тексте, но при добавлении атрибута `class` его можно варьировать.

- **Примечание:** чтобы добавить или убрать комментарий в Brackets, нужно нажать сочетание клавиш **CTRL + /**. Если нужно закомментировать/раскомментировать 1 строку, достаточно, чтобы в ней просто стоял курсор. Для комментирования блока текста необходимо его выделить и нажать **CTRL + /**.

## Тег **blockquote**

Еще из блочных элементов рассмотрим тег **blockquote**, который предназначен для форматирования чьих-либо высказываний — от известных людей до директоров компаний, для которых создается сайт.

```
<blockquote>Блочная цитата</blockquote>
```

Нужно отметить, что на обычных сайтах используется этот тег нечасто, зато он очень популярен на различных форумах.

На скриншоте ниже приведен отрывок с [форума о работе с CMS Joomla](#), где цитируются слова одного из пользователей, и для этого использован именно тег **blockquote** с отличным от остального текста css-форматированием.

The screenshot shows a forum post from the Joomla 3.7.4 administration top menu frozen thread. The post by user 'joehanSEN' at Thu Aug 03, 2017 8:43 am contains a blockquote from 'Niels klint' with the message: "Have you tried Fix Database - .../administrator/index.php?option=com\_installer&view=database?". A red box highlights this quoted text, and a red arrow points to the word 'blockquote' in the original post below it. Another red box highlights the user 'leolam' at Thu Aug 03, 2017 8:44 am, and a red arrow points to the word 'blockquote' in their reply. The original post by 'joehanSEN' includes a profile picture, a star rating of 5 stars, and the text 'Posts: 5 Joined: Tue Jan 14, 2014 2:07 pm'. The reply by 'leolam' also includes a profile picture, a star rating of 5 stars, and the text 'Posts: 5 Joined: Tue Jan 14, 2014 2:07 pm'. The bottom of the screenshot shows a footer note: 'Cleaning Cache did not help. And difficult to do when you cannot use the Administration Menu. Tried to open in two different browsers - did not help.'

Кстати, аббревиатура Emmet для создания этого тега очень простая:

```
bq
```

По умолчанию, `blockquote` имеет отступы в 40px справа и слева, а также отступы сверху и снизу, как в абзацах.

The screenshot shows the Brackets IDE interface. On the left, there's a code editor with the file 'use-blockquote.html' containing HTML code. The code includes a `<head>` section with a meta tag and a title. The `<body>` section contains a `<h1>Using blockquote tag</h1>`, followed by a `<p>` tag containing a `<b><blockquote></b>` element. This blockquote contains a source reference and a paragraph about WWF. Below this, another `<blockquote>` block contains a quote from Thomas Edison. The right side of the interface shows a browser-like preview window titled 'Use blockquote'. It displays the same HTML structure with visual styling: the first `blockquote` has a grey background and white text, while the second one has a white background with black text. The preview also shows the source code at the bottom.

```
use-blockquote.html (HTML-1.1) - Brackets
Debug Help Emmet
4 <head>
5   <meta charset="UTF-8">
6   <title>Use blockquote</title>
7 </head>
8
9 <body>
10   <h1>Using blockquote tag</h1>
11   <p>The <b><blockquote></b></p> specifies
12     a section that is quoted from another
13     source.</p>
14   <p>Browsers usually indent blockquote
15     elements.</p>
16   <blockquote>
17     For 50 years, WWF has been
18       protecting the future of nature.
19       The world's leading conservation
20       organization, WWF works in 100 countries and is
21       supported by 1.2 million members in the United
22       States and close to 5 million
23       globally.
24 </blockquote>
25   <p>The tag <b><blockquote></b></p> is for
26   example used to display the statements
27   of famous people.</p>
28   <p>Thomas Edison says:</p>
29   <blockquote>Success is one percent
30   inspiration, ninety-nine percent
31   perspiration.
32     <br/>/Thomas Edison/
33 </blockquote>
34
35 Column 10 — 29 Lines  INS  UTF-8  HTML  Spaces: 4
```

Посмотреть пример можно в файле `use-blockquote.html` (прикреплен к PDF-файлу данного урока).

### Одиночные теги

В примере с `blockquote` внутри цитаты был использован тег `<br>` — от англ. *break* — разбивать. Он предназначен для переноса текста, следующего за ним, на следующую строку и является строчным, т.е. обычно размещается в тексте абзацев, `div`-ов и других элементов.

Поскольку тег не имеет внутреннего содержимого, то закрывающий тег ему не нужен, т.е. он относится к группе одиночных тегов, или тегов без содержимого.

К ним же относится тег `<img>` для встраивания изображений и целая группа тегов `<input>`, из которых состоят формы.

Здесь же мы рассмотрим еще один такой тег — это `<hr>`, или горизонтальная строка (от англ. *horizontal row*).

Собственно, он и выводит в браузер горизонтальную линию. В HTML4.01 для нее можно было задать ряд атрибутов, которые в HTML5 являются отмененными. Поэтому все «красивости» нужно задавать через CSS.

В файле `use-blockquote-hr.html` (прикреплен к PDF-файлу данного урока), можно добавить горизонтальные линии для визуального отделения цитат от текста абзаца. Как видно из скриншота, `<hr>` — это блочный тег, который занимает все доступное пространство в браузере.

The screenshot illustrates the use of the `<hr>` tag in an HTML document. On the left, the Brackets code editor displays the file `use-blockquote-hr.html`. Several instances of the `<hr>` tag are highlighted with red boxes. The code includes examples of how `<hr>` is used to separate quoted text from the main text and between different quote sources. On the right, a screenshot of a web browser shows the rendered HTML. It features horizontal lines separating different sections of text, such as the introduction of the `<blockquote>` tag, the statement about WWF, the quote by Thomas Edison, the quote by Jim Morrison, and the concluding statement about freedom. The browser's address bar shows the URL `127.0.0.1:1297/use-blockquote.html`.

```

use-blockquote-hr.html (HTML-1-1) - Brackets
Navigate Debug Help Emmet
10 <p><b><code><blockquote></b></code></b></p>
  specifies a section that is quoted
  from another source.</p>
11 <p>Browsers usually indent
  blockquote elements.</p>
12 <hr>
13 <blockquote>
14   For 50 years, WWF has been
    protecting the future of
    nature. The world's leading
    conservation organization, WWF
    works in 100 countries and is
    supported by 1.2 million
    members in the United States
    and close to 5 million
    globally.
15 </blockquote>
16 <p>The tag <code><blockquote></code></b> is for
  example used to display the
  statements of famous people.</p>
17 <p>Thomas Edison says:</p>
18 <hr>
19 <blockquote>Success is one percent
  inspiration, ninety-nine percent
  perspiration.
    <br>/Thomas Edison/
  </blockquote>
20 <hr>
21 <blockquote>Jim Morrison about freedom.</p>
22 <hr>
23 <blockquote>The most important kind of
  freedom is to be what you
  really are. You trade in your
  sense for an act. You give up your ability to feel, and
  in exchange, put on a mask. There can't be any large-scale
  revolution until there's a personal revolution, on an
24 <hr>
25 <blockquote>

```

- **Примечание:** В XHTML для всех одиночных тегов был обязательным закрывающий слэш в конце через пробел от имени тега, поэтому на ряде сайтов вы можете встретить такое написание тегов:

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

На данный момент необходимости в добавлении этого слэша нет.

Давайте подытожим все, что мы узнали на данный момент об html-форматировании и выведем это в виде списка правил.

# Правила HTML

1. **Все пробелы, табуляция и начало новых строк преобразуются в обычный пробел.** Поэтому очень желательно форматировать текст с отступами:

```
<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 similiique, modi
        officiis, 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. **Весь текст должен быть отформатирован в виде тегов.** Обязательно присутствуют открывающий и закрывающий тег:

```
<элемент>Текст элемента</элемент>
```

3. **Если у элемента нет содержимого, закрывающий тег ему НЕ нужен.**

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

4. **Всегда должны быть теги основы:** <html>, <head>, <title>, <body>. И они должны соответствовать содержанию документа.
5. **Правило «матрешки»:** всегда соблюдать вложенность элементов и не допускать их пересечения:

```
<b><i>Текст внутри тегов</i></b>
```

6. **Значения атрибутов** всегда берутся **в кавычки**. Атрибутов у элемента может быть несколько, они отделяются друг от друга пробелами:

```

```

7. **Универсальными атрибутами** для всех тегов являются: id, class, style, title, tabindex

```
<div id="box" class="block" title="Блочный элемент"  
style="font-family: Verdana, Tahoma, sans-serif">  
    Текст элемента  
</div>
```

8. **Комментарии** в HTML предназначены для того, чтобы скрыть часть кода и записываются так:

```
<-- Комментарий -->
```

# Стили CSS

Ну, вот мы и добрались до визуального форматирования страницы. С точки зрения семантики, необходимо отделять «мух от котлет», т.е. визуальное форматирование страницы сайта от его html-структурь. А CSS как раз и занимается форматированием. И имеет массу свойств для этого в своем арсенале.

Добавлять css-форматирование можно 4-мя способами:

1. С помощью атрибута style
2. Внутри тегов <style> в блоке <head>
3. С помощью тега <link>
4. С помощью директивы @import.

В этом уроке мы рассмотрим первых 2 способа. Остальные — в следующих уроках.

## Внутренние стили CSS (inline styles)

Для начала рассмотрим, как можно задать форматирование для каждого элемента.

Для этого предназначен атрибут style, о котором уже шла речь, как об одном из универсальных, т.е. применимым для любого тега.

Например, нам необходимо изменить цвет и стиль текста для какого-либо абзаца, а также назначить для него другой шрифт. Делается это следующим образом:

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

Т.е. в атрибуте style, который, напомню, записывается только в открывающем теге, необходимо разместить пары «свойство: значение», которые отделяются друг от друга точкой с запятой. После последнего значения свойства ставить точку с запятой необязательно. Важно добавить ее, если вы еще захотите написать одно свойство.

На скриншоте ниже можем увидеть разницу между обычными абзацами и абзацем с форматированием в атрибуте style:

The screenshot shows a Brackets code editor window titled "style-inline.html (HTML-1-1) - Brackets". The code is as follows:

```
5 <meta charset="UTF-8">
6 <title>Music Styles</title>
7 </head>
8
9 ▼ <body>
10 <h1>Music Styles </h1>
11 <p style="color: #999; font-style: italic; font-family: 'Open Sans', Verdana, Tahoma, sans-serif">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
```

Below the editor is a browser window titled "Music Styles" showing the rendered HTML. The paragraph text is displayed in a light blue color and italicized.

### Music Styles

*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, i 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.*

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 a offer, two offers frequently form the period.

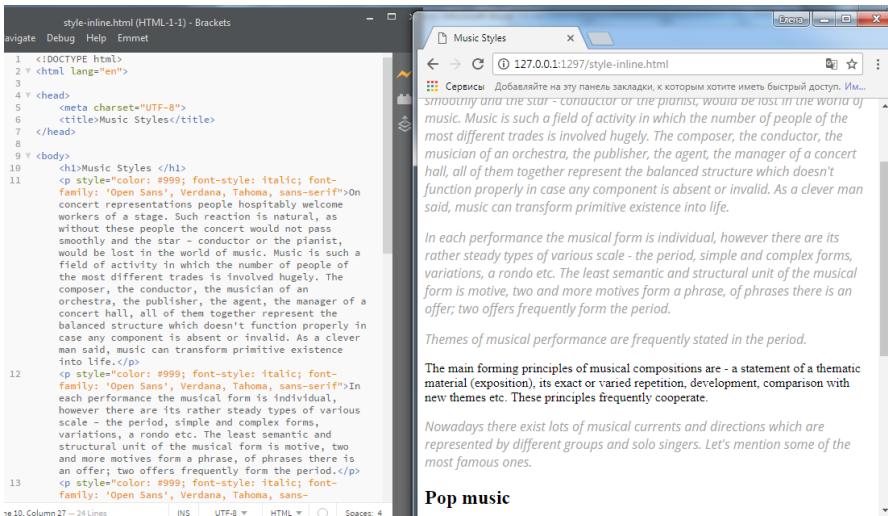
Themes of musical performance are frequently stated in the period.

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 famous ones.

Если мы хотим однобразно отформатировать все абзацы, то придется скопировать этот атрибут со всеми его значениями и добавить во все абзацы:

## Стили CSS

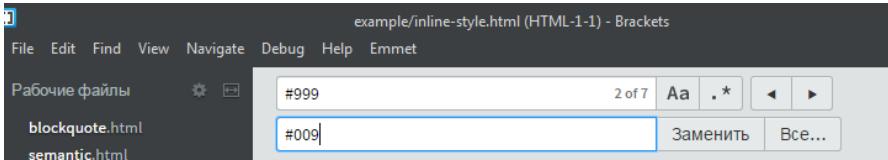


```
style-inline.html (HTML-1-1) - Brackets
avigate Debug Help Emmet
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Music Styles</title>
6 </head>
7 <body>
8   <h1>Music Styles </h1>
9   <p style="color: #999; font-style: italic; font-family: 'Open Sans', Verdana, Tahoma, sans-serif">On
10  concert stage the composition might be performed by
11  the workers of a stage. Such reaction is natural, as
12  without these people the concert would not pass
13  smoothly and the star – conductor or the pianist,
14  would be lost in the world of music. Music is such a
15  field of activity in which the number of people of
16  the most different trades is involved hugely. The
17  composer, the conductor, the musician of an
18  orchestra, the manager, the lighting designer, the manager of a
19  concert hall, all of them together represent the
20  balanced structure which doesn't function properly in
21  case any component is absent or invalid. As a clever
22  man said, music can transform primitive existence
23  into life.</p>
24   <p style="color: #999; font-style: italic; font-family: 'Open Sans', Verdana, Tahoma, sans-serif">In
25  each performance the musical form is individual,
26  however there are its rather steady types of various
27  scale – the period, simple and complex forms,
28  variations, a rondo etc. The least semantic and
29  structural unit of the musical form is motive, two
30  and more motives form a phrase, of phrases there is an
31  offer; two offers frequently form the period.</p>
32   <p style="color: #999; font-style: italic; font-family: 'Open Sans', Verdana, Tahoma, sans-serif">In
33  each performance the musical form is individual,
34  however there are its rather steady types of various
35  scale – the period, simple and complex forms,
36  variations, a rondo etc. The least semantic and
37  structural unit of the musical form is motive, two
38  and more motives form a phrase, of phrases there is an
39  offer; two offers frequently form the period.</p>
40 </body>
```

Уже смотрится лучше, не так ли?

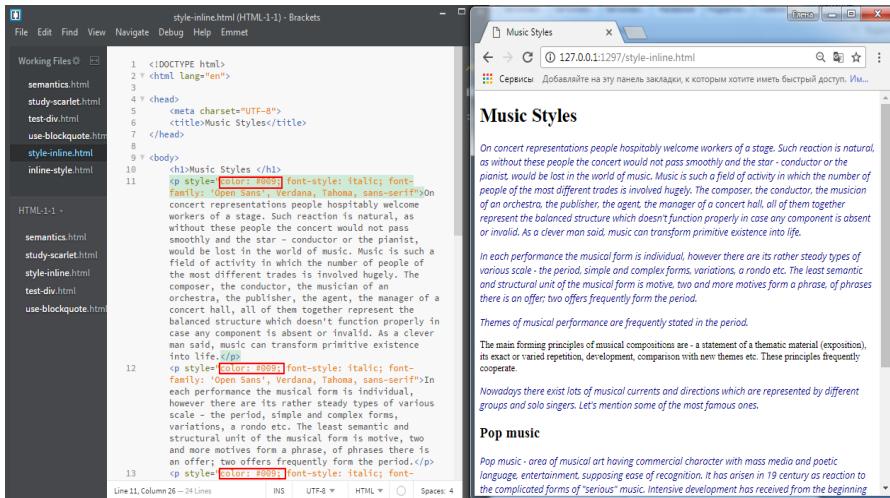
А теперь представим, что мы передумали задавать цвет текста в виде color: #999, а хотим его сделать темно-синим, например... Для опытов вы можете использовать файл style-inline.html (прикреплен к PDF-файлу данного урока).

В этом случае, конечно, выручит нас поиск и замена текста (клавиши **CTRL + H**):



В полях ввода здесь нужно указать, что мы заменяем в первом и на что — во втором, а затем нажать на кнопку «Все». Заменятся все строки, в которых текст имел светло-серый цвет.

Все сработало — цвет стал темно-синим:



```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Music Styles</title>
</head>
<body>
<h1>Music Styles</h1>
<p style="color: red; font-style: italic; font-family: 'Open Sans', Verdana, Tahoma, sans-serif">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 not be able to show his art. 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 company etc. 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.</p>
<p style="color: green; font-style: italic; font-family: 'Open Sans', Verdana, Tahoma, sans-serif">In each performance the musical form is individual, however there are other steady types of various scale - the period, simple and complex forms, variations, a rondo etc. The least semantic and structural units of the musical form is motive, two and more motives form a phrase, of phrases there is an offering, which offers frequently for the audience. In each performance the musical form is individual, however there are other steady types of various scale - the period, simple and complex forms, variations, a rondo etc. The least semantic and structural units of the musical form is motive, two and more motives form a phrase, of phrases there is an offering, which offers frequently for the audience. In each performance the musical form is individual, however there are other steady types of various scale - the period, simple and complex forms, variations, a rondo etc. The least semantic and structural units of the musical form is motive, two and more motives form a phrase, of phrases there is an offering, which offers frequently for the audience. In each performance the musical form is individual, however there are other steady types of various scale - the period, simple and complex forms, variations, a rondo etc. The least semantic and structural units of the musical form is motive, two and more motives form a phrase, of phrases there is an offering, which offers frequently for the audience. In each performance the musical form is individual, however there are other steady types of various scale - the period, simple and complex forms, variations, a rondo etc. The least semantic and structural units of the musical form is motive, two and more motives form a phrase, of phrases there is an offering, which offers frequently for the audience. In each performance the musical form is individual, however there are other steady types of various scale - the period, simple and complex forms, variations, a rondo etc. The least semantic and structural units of the musical form is motive, two and more motives form a phrase, of phrases there is an offering, which offers frequently for the audience. In each performance the musical form is individual, however there are other steady types of various scale - the period, simple and complex forms, variations, a rondo etc. The least semantic and structural units of the musical form is motive, two and more motives form a phrase, of phrases there is an offering, which offers frequently for the audience. In each performance the musical form is individual, however there are other steady types of various scale - the period, simple and complex forms, variations, a rondo etc. The least semantic and structural units of the musical form is motive, two and more motives form a phrase, of phrases there is an offering, which offers frequently for the audience. In each performance the musical form is individual, however there are other steady types of various scale - the period, simple and complex forms, variations, a rondo etc. The least semantic and structural units of the musical form is motive, two and more motives form a phrase, of phrases there is an offering, which offers frequently for the audience. In each performance the musical form is individual, however there are other steady types of various scale - the period, simple and complex forms, variations, a rondo etc. The least semantic and structural units of the musical form is motive, two and more motives form a phrase, of phrases there is an offering, which offers frequently for the audience. In each performance the musical form is individual, however there are other steady types of various scale - the period, simple and complex forms, variations, a rondo etc. The least semantic and structural units of the musical form is motive, two and more motives form a phrase, of phrases there is an offering, which offers frequently for the audience. In each performance the musical form is individual, however there are other steady types of various scale - the period, simple and complex forms, variations, a rondo etc. The least semantic and structural units of the musical form is motive, two and more motives form a phrase, of phrases there is an offering, which offers frequently for the audience. In each performance the musical form is individual, however there are other steady types of various scale - the period, simple and complex forms, variations, a rondo etc. The least semantic and structural units of the musical form is motive, two and more motives form a phrase, of phrases there is an offering, which offers frequently for the audience. In each performance the musical form is individual, however there are other steady types of various scale - the period, simple and complex forms, variations, a rondo etc. The least semantic and structural units of the musical form is motive, two and more motives form a phrase, of phrases there is an offering, which offers frequently for the audience. In each performance the musical form is individual, however there are other steady types of various scale - the period, simple and complex forms, variations, a rondo etc. The least semantic and structural units of the musical form is motive, two and more motives form a phrase, of phrases there is an offering, which offers frequently for the audience. In each performance the musical form is individual, however there are other steady types of various scale - the period, simple and complex forms, variations, a rondo etc. The least semantic and structural units of the musical form is motive, two and more motives form a phrase, of phrases there is an offering, which offers frequently for the audience. In each performance the musical form is individual, however there are other steady types of various scale - the period, simple and complex forms, variations, a rondo etc. The least semantic and structural units of the musical form is motive, two and more motives form a phrase, of phrases there is an offering, which offers frequently for the audience. In each performance the musical form is individual, however there are other steady types of various scale - the period, simple and complex forms, variations, a rondo etc. The least semantic and structural units of the musical form is motive, two and more motives form a phrase, of phrases there is an offering, which offers
```

Но, представим, что блоков у нас много, и не во всех можно заменить цвет на предложенный. Тогда поиск и замена усложняется. А если еще нужно менять и шрифт, и стиль, то головной боли станет намного больше. Поэтому этот метод — добавления стилей через атрибут style — годится либо в самом начале изучения HTML/CSS, либо когда у вас нет другого выхода — например, при редактировании текста статей в CMS Wordpress или Joomla, где по-другому вы часть текста не выделите.

## Стили для страницы

Намного лучше даже на начальном этапе использовать второй способ — форматирование внутри страницы в специальных тегах `<style></style>`, которые размещаются в нижней части блока `<head>`. Стоит заметить, что в этом случае стили применяются только для той страницы, на которой они будут размещены. Это как раз при-

емлемый вариант для начала обучения HTML и CSS — в пределах одного документа вы видите и html-разметку, и CSS-стили.

Этот вид форматирования позволяет задать стили для селекторов CSS — специальных описаний для элемента или группы элементов, к которым применяется определенные стилевые правила форматирования.

На данный момент CSS3 дает нам множество различных селекторов, с которыми мы будем знакомиться последовательно на различных примерах. Сейчас остановимся на простых селекторах, таких как универсальный селектор, селектор элемента, или тега, селектор группы, селектор класса и селектор id.

## Селектор элемента

Начнем с селектора элемента и рассмотрим общий синтаксис. На рисунке видно, что селектором является тег p, в фигурных скобках размещается блок правил для этого селектора, а сами правила представляют собой пары «свойство: значение» и отделяются друг от друга точкой с запятой.

### Правило CSS



Похоже на встроенный стиль, не так ли? Разница заключается только в том, что такие правила применяются сразу ко всем абзацам в документе — и не нужно копировать их в каждый. Плюс — при изменении в любом из свойств или при добавлении нового, сразу все абзацы изменят форматирование, без необходимости использовать «Поиск и Замену».

Добавим в предложенные правила еще другой цвет, и вот что получим:

```

page-style.html (HTML-1-1) - Brackets
Debug Help Emmet
5 <meta charset="UTF-8">
6 <title>Music Styles</title>
7 <style>
8   p {
9     color: #999;
10    font-size: 18px;
11    font-style: italic;
12    font-family: Verdana, Geneva,
13      sans-serif;
14  }
15 </style>
16 </head>
17 <body>
18   <h1>Music Styles </h1>
19   <p>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 huge composer, the conductor, the musician of an orchestra, the publisher, the agent, the manager of a concert hall, all 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.</p>
20   <p>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.</p>
21   <p><b>Themes of musical performance are frequently stated in the period.</b></p>

```

Music Styles

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 huge composer, the conductor, the musician of an orchestra, the publisher, the agent, the manager of a concert hall, all 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.

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 is statement of a thematic material (exposition), its examination, 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 artists.

Как видно из скриншота, все абзацы в документе поменяли форматирование без особых усилий по копированию с нашей стороны.

## Универсальный селектор

Универсальный селектор применяется для назначения правил сразу для всех элементов на странице, включая html и body. Обозначается он с помощью звездочки (\* — клавиши SHIFT + 8).

Он необходим для того, чтобы установить одинаковые правила для всех элементов страницы. Например, вместо того, чтобы задавать размер шрифта (*font-size*) и семейство шрифта (*font-family*) только для абзацев, установим их сразу для всей страницы:

The screenshot shows the Brackets IDE interface with a code editor and a browser preview window. The code editor contains the following CSS:

```

page-style.html (HTML-1-1) - Brackets
Debug Help Emmet

1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <title>Music Styles</title>
7   <style>
8     * {
9       font-size: 18px;
10      font-family: Verdana, Geneva,
11          sans-serif;
12    }
13    p {
14      color: #999;
15      font-style: italic;
16      /* font-size: 18px;
17      font-family: Verdana, Geneva,
18          sans-serif; */
19    }
20  </style>
21 </head>
22 <body>
23   <h1>Music Styles </h1>
24   <p>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.</p>
25   <p>In each performance the musical form is individual, however there are its rather
26

```

The browser preview shows the page title "Music Styles". The main content area contains the text from the code, with the first paragraph styled in bold italic font (18px Verdana sans-serif) and the second paragraph in regular font (16px Verdana sans-serif). A tooltip over the second paragraph provides a detailed explanation of the Blues genre.

**Blues**

*Blues - a genre of the Afro-American musical folklore and a jazz carrying melancholy and despondency. It formed in the beginning of 20 century. The blues tradition is submitted practically in all basic jazz styles. The blues reflects deeply personal experiences sated with dramatic nature and an internal conflictiness, contains also elements of humor, irony, social satire.*

**Musics**

*Specificity of a blues is shown, besides in a special structure of a poetic stanza (стrophe) and an originality of the musical form, in use of the certain circle of expressive means, in a manner of performance.*

*Themes of musical performance are frequently stated in the period.*

Поскольку у нас не так много элементов на странице, размер и семейство шрифта применились к абзацам и заголовкам. Обратите внимание, что заголовки стали меньше, чем были ранее.

- **Примечание:** на самом деле для универсального селектора чаще задают несколько иные правила, но с ними вы познакомитесь в теме «Блочная модель элементов». Те же правила, которые мы сейчас задали для \*, обычно записывают для селектора элемента *body*.

### Комментарии в CSS

Обратите внимание, что комментарии в CSS отличаются от комментариев в HTML и задаются с помощью

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

В Brackets внутри тегов style эти комментарии также создаются с помощью клавиш **CTRL + /** — программа сама определяет, какие комментарии нужны в этих тегах.

### Селектор группы, или групповой селектор

Как следует из его названия, селектор группы применяется к группе элементов, классов, id или сочетаниям этих селекторов. В нем все нужные вам селекторы перечисляются через запятую.

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

Нужен такой селектор для того, чтобы уменьшить количество кода для однотипных элементов. В примере *page-style.html* (прикреплен к PDF-файлу данного урока) для всех заголовков был изменен шрифт, межстрочное расстояние и расстояние между буквами.

**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.*

Чуть позже будет рассмотрен пример с классами style-class.html, где групповой селектор выглядит так:

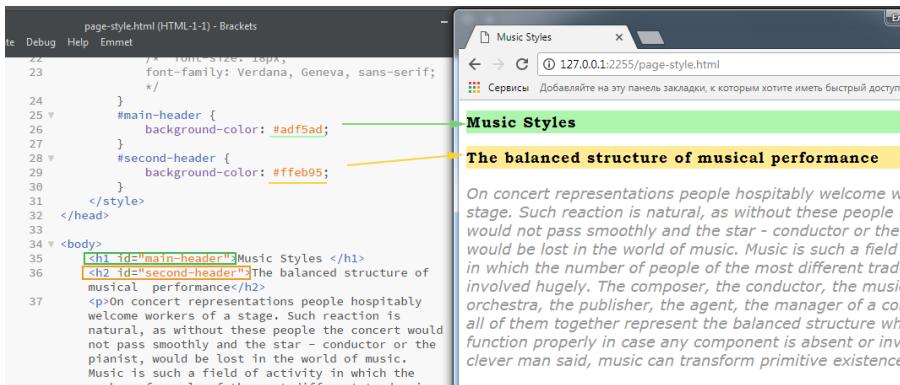
```
.cursive, .fantasy {
    font-weight: bold;
    letter-spacing: 2px;
    font-size: 1.5rem;
}
```

**Селектор id**

Селектор id подразумевает, что на странице для какого-либо одного элемента указан атрибут id — уникальный идентификатор (*identifier*). Значение этого атрибута в кавычках не может повторяться в пределах одной страницы, иначе никакой уникальности не получится. Обозначается селектор id символом решетка # рядом со значением атрибута id нужного элемента. В примере

ниже мы используем 2 разных id для того, чтобы задать различный цвет фона разным заголовкам:

```
#main-header {  
    background-color: #adf5ad;  
}  
  
#second-header {  
    background-color: #ffeb95;  
}
```



В html-разметке мы добавили именно такие id для разных заголовков. Посмотреть пример можно в файле page-style.html (прикреплен к PDF-файлу данного урока).

Обратите внимание, что значение каждого id представлено в единственном экземпляре. Это важно, иначе об уникальности и о валидности документа говорить нельзя.

### Селектор класса

Селектор класса — это №1 по популярности при верстке страницы. Любая реальная страница содержит массу элементов с различными классами в виде атрибутов.

И чаще всего такими элементами являются div-ы и вложенные в них span-ы.

В примере ниже в html-разметке использованы 3 div-а с классом «block» и 3 div-а с классом «box». В файле style-class.html (*прикреплен к PDF-файлу данного урока*) вы найдете аббревиатуры Emmet для того, чтобы сделать это самостоятельно. Для каждого из них задан набор правил, который изменяет размер и семейство шрифта, а также формирует разного типа границу в нижней части div-а:

The screenshot shows the Brackets IDE interface. On the left, the code editor displays the file `example-style-class.html` with the following content:

```

1  <!DOCTYPE html>
2  <html lang="en">
3
4    <head>
5      <meta charset="utf-8">
6      <title>Using Classes</title>
7      <style>
8        body {
9          color: #444;
10         }
11        .block {
12          font-size: .8em;
13          font-family: Calibri, sans-serif;
14          border-bottom: 3px double #666;
15        }
16        .box {
17          font-size: 1.2em;
18          font-family: Cambria, serif;
19          color: #0e4881;
20          border-bottom: 3px dotted #0e4881;
21        }
22      </style>
23    </head>
24
25    <body>
26      <h1>Использование классов</h1>
27      <!-- .block-->
28      <div class="block">

```

At the bottom of the code editor, there are status bars: "Строка 12, Страница 29 — 644 строк", "БЦТ", "HTML", and "Проблемы: 4".

On the right, a browser window titled "Using Classes" displays the rendered HTML. The browser address bar shows `127.0.0.1:1502/example/head-tag.html`. The page content includes three sections:

- Div header 2**: A heading with a double border at the bottom.
- Div header 3**: A heading with a dotted border at the bottom.
- Box Div header 1**: A heading with a solid border at the bottom.

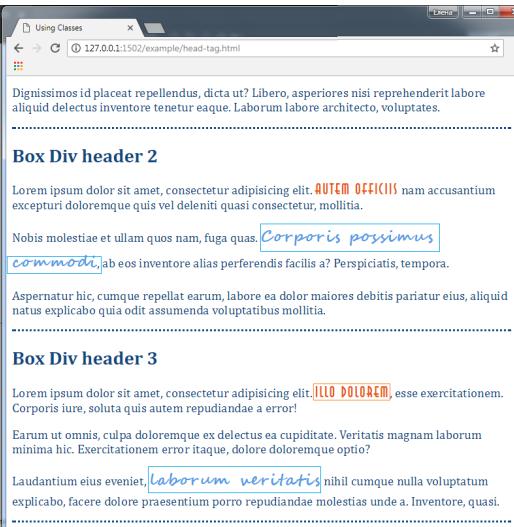
Below each heading, there is a paragraph of placeholder text: "Lorem ipsum dolor sit amet, consectetur adipisicing elit. Harum nemo provident aperiam, laborum doloribus iure, tenetur, cum ea reprehenderit architecto eos. Praesentium asperiores, ipsa vel necessitatibus ab quasi! Molitiae atque repellendus, doloribus voluptatum excepturi voluptas vel explicabo quasi rerum optio alias eum, asperiores, doloribus obsecrati nisi cum rem, sit quis qui tempore quen sed nihil eos. Esse..."

At the bottom of the browser window, there is a footer with the text "Quaeatur tenetur, odit ut. Deserunt nihil magni quidem non hic quasi iure sed ipsum placeat natus cum, fuga illum quis, voluptates qui sunt, obsecrati modi, quo repudiandae sed districto!"

At the very bottom of the browser window, there is another footer with the text "Box Div header 1" and "Box Div header 2".

Плюс в `<div class="block">` цвет текста изменен на темно-синий. Пока мы не будем добавлять более сложные правила стилей, но именно для селекторов классов задают обычно много стилевых правил.

Добавим еще в разметку элементы span с классами «cursive» и «fantasy» и зададим для них ряд общих правил и ряд специальных:



В результате увидим, что в документе (style-class.html) часть текста станет оранжевого и голубого цвета с несколько увеличенным размером шрифта. Сразу предупрежу, что не у всех вас на компьютерах внешний вид текста совпадет с тем, что на скриншоте, т.к. это зависит от установленных в системе шрифтов.

Почитать еще про селекторы:

- <http://htmlbook.ru/metki/selektory>
  - <https://learn.javascript.ru/css-selectors>
  - <http://html-plus.in.ua/css-selectors-simple-selectors/>
  - <http://html-plus.in.ua/podklyuchenie-css-stiley-pravila-zapisи-css/>
  - <http://vvz.nw.ru/Lessons/CSS/selectors.htm>

# CSS-свойства

Мы уже рассмотрели часть css-свойств, не слишком плотно останавливаясь на их значениях. Думаю, что было интуитивно понятно, что свойство `color` отвечает за цвет шрифта, а `font-size` — за размер шрифта элемента.

Теперь давайте поговорим о свойствах и их возможных значениях подробней. Сразу оговорюсь, что в рамках первого урока мы рассмотрим самые необходимые. Все остальные мы будем рассматривать по мере изучения курса.

## Варианты назначения цвета

Для установки цвета различных элементов можно использовать несколько правил. Мы рассмотрим сегодня свойство `color`, которое отвечает за цвет текста, и `background-color`, которое управляет цветом фона элемента.

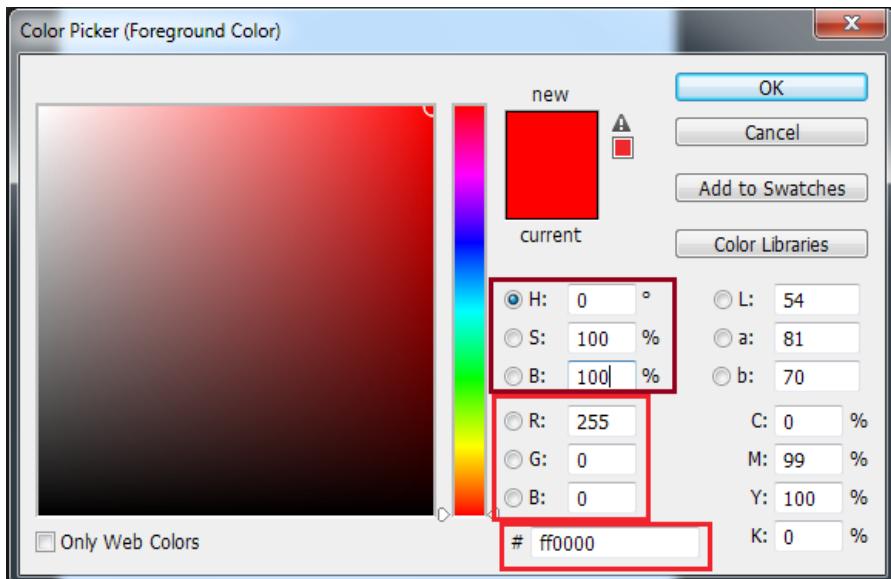
И в том, и в другом свойстве можно задавать значения в следующих форматах:

```
color: red; /* название цвета на английском языке */
color: #ff0000; /* 16-ричное значение цвета,
использующее цифры от 0 до 9 и буквы от a до f*/
color: #f00; /* усеченное 16-ричное значение для
цветов, у которых одинаковые пары цифр */
color: rgb(255, 0, 0); /* значение цвета в системе
rgb. По каждому каналу r — red, g — green,
b — blue значения цвета меняются от 0 до
255 единиц*/
color: rgba(255, 0, 0, .5); /* значение цвета
в системе rgb + альфа-канал, т.е. возможность
задать прозрачность для цвета от 0 до 1 */
```

```
color: hsl(0, 100%, 50%); /* значение цвета
    в системе hsl — hue -saturation - lightness */
color: hsla(0, 100%, 50%, .5); /* значение цвета
    в системе hsl + альфа-канал, т.е. возможность
    задать прозрачность для цвета от 0 до 1 */
```

Из всех приведенных вариантов наименее понятными, пожалуй, будут два последних, т.к. цветовая система **HSL** (*Hue, Saturation and Lightness* — тон, насыщенность и светлота) вряд ли была предметом вашего изучения на курсе Photoshop — там она тоже нечасто используется (для web, во всяком случае).

Все эти варианты вы можете почерпнуть из Палитры цветов (*Color Picker*) в Photoshop. Только в этой программе (на скриншоте версия CS6) вместо системы HSL — HSB (B — *brightness* — яркость).

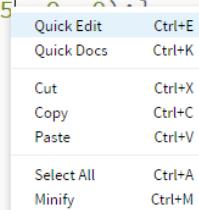


Что касается Brackets, то одной из его «плюшек из коробки» является очень простой выбор и изменение цветовых свойств. Для того чтобы на это посмотреть, создайте теги style, и для селектора любого элемента, например для p, задайте свойство color с любым из выше-приведенных значений. А затем поставьте курсор внутри значения цвета и нажмите **CTRL + E** (от англ. **edit** — *редактировать*). Для любителей контекстного меню по правому клику на значении цвета можно выбрать пункт «Быстрое редактирование».

- **Примечание:** Закрыть панель быстрого редактирования цвета вы можете повторным нажатием клавиши **CTRL + E**, клавишей **ESC** или на крестик слева вверху.

```

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>
```



Смотрите, не промахнитесь, т.к. если курсор будет находиться внутри свойства или за пределами значения этого свойства, нажатие на **CTRL + E** вызовет сообщение об ошибке:

```

4 ▼ <head>
5      <meta charset="UTF-8">
6      <title>Color Property in CSS</title>
7 ▼          p {
8 ▼              color: rgb(255, 0, 0);
9          }
10         }
11     </style>
12 </head>
13

```

Итак, вы нажали **CTRL + E** и можете выбирать между системой RGBa, Hex, HSLa. По умолчанию у вас будет система Hex с 16-ричным значением цвета. Что касается систем RGBa или HSLa, то изначально прозрачность в них не предусмотрена. Задать ее вы можете, перемещая ползунок на полоске справа от палитры цвета.



Еще несколько замечаний по цвету:

1. Если вы задаете его значение в виде слова, то Brackets подскажет вам все варианты цветов, куда это слово (или его часть) входит.

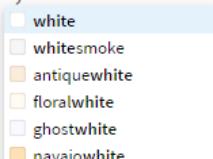
```

<style>
  p{
    color: rgba(255, 0, 0, 0.53);
    background-color: white;
  }

</style>
</head>

<body>

```



2. Если вы использовали несколько цветов в проекте, назначая цвет для нового элемента, в панели быстрого редактирования вы можете увидеть список из всех цветов, которые уже были использованы ранее. Щелкните на любом из них, и он будет назначен для вашего селектора.

```

7      <style>
8        body {
9          background-color: pink;
10         }
11        p {
12          color: rgba(255, 0, 0, 0.53);
13          background-color: white;
14         }
15        h1 {
16          color: lightcoral

```

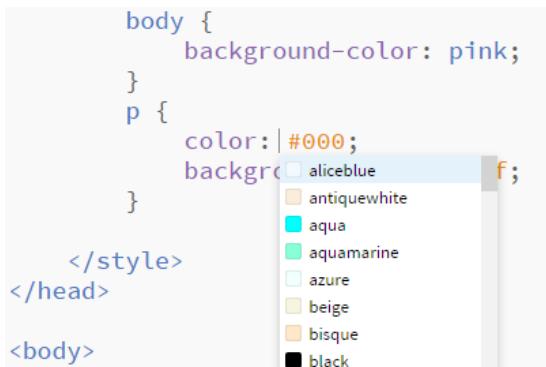


```

17        }
18      </style>

```

3. Если вы не знаете, какой цвет вам нужен или вам не нравится выбранный, поставьте курсор после двоеточия и нажмите **CTRL + пробел** — это вызов подсказки. И выбирайте из значений цвета:



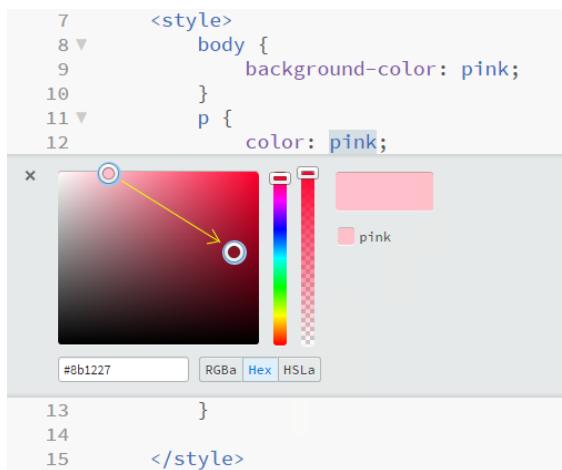
```

body {
    background-color: pink;
}
p {
    color: #000;
    background-color: aliceblue;
}
</style>
</head>
<body>

```

A color picker dropdown is open over the word "aliceblue". It shows a vertical color bar with various color names: antiquewhite, aqua, aquamarine, azure, beige, bisque, and black. The color bar has a light gray gradient at the bottom.

4. Если вам нужно самостоятельно подобрать оттенки близких цветов, чтобы они сочетались, например, для цвета фона — посветлее, а для текста — потемнее, задайте сначала одинаковые значения для обоих цветов, а затем, вызвав палитру нажатием **CTRL + E**, сместитесь в более темную область:



```

7 <style>
8 body {
9     background-color: pink;
10 }
11 p {
12     color: pink;

```

A color picker is open over the word "pink" in the "color" declaration. The picker shows a dark-to-light gradient. A yellow arrow points from the center of the gradient towards the left side, indicating a darker shade. Below the picker are buttons for "#8b1227", "RGBa", "Hex", and "HSLa".

```

13 }
14
15 </style>

```

5. И напоследок вспомним про аббревиатуры Emmet. В css они тоже работают. Для свойства color достаточно написать **c** и нажать **TAB**, а для background-color — набираем **bgc** и **TAB**. Аббревиатуры раскроются со значениями цвета по умолчанию (черным для цвета текста и белым — для цвета фона):

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

Значения после знака # выделены, поэтому сразу можете вводить нужные вам или нажимать **CTRL + E**, чтобы их подобрать.

## Свойства шрифта

По-английски шрифт — это **font**, поэтому свойства для шрифта в css начинаются именно с этого слова. Их несколько, и выполняют они следующие изменения шрифтов на странице:

- На семейство шрифтов указывает свойство **font-family**. В значении этого свойства можно перечислить через запятую несколько шрифтов в той последовательности, в которой вы (или заказчик) бы хотели, чтобы они отображались на сайте. Например, запись

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

предполагает, что браузер сначала попытается отобразить текст с помощью шрифта Lato. Но, если его нет на компьютере пользователя, выведет содержимое body шрифтом Verdana. Если же и шрифт Verdana отсутствует, то будет использован шрифт Geneva.

И в том случае, если нет ни одного из перечисленных шрифтов, будет использован шрифт по умолчанию, тип которого задается специальным ключевым словом (их еще называют шрифты-псевдонимы):

### **Font-family defaults**

#### **Font-family serif**

*Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ducimus, voluptates?*

*Repellendus ipsa quos voluptas nam atque voluptatem labore ipsum facere.*

#### **Font-family sans-serif**

*Lorem ipsum dolor sit amet, consectetur adipisicing elit. Obcaecati, adipisci.*

*Molestias accusantium earum eius repellat deserunt, nobis preferendis. Suscipit, ve*

#### **Font-family monospace**

*Est sint enim, ex dolor veniam assumenda reprehenderit sit tempora?*

*Consequuntur debitisi, itaque assumenda hic sed quia eos in maxime?*

#### **Font-family cursive**

*Repubiandae voluptate quo cumque doloribus, aspernatur facilis totam,*

*Modi at vitae unde molestias, labore sequi dolorem architecto. Accusan*

#### **Font-family fantasy**

**Ab mollitia quos fugit, optio, deserunt ducimus voluptatum veritatis pariatur!**

**Aliquid ab molestias quos? Alias earum maxime excepturi iste quos?**

- **serif** — для шрифтов с засечками (по умолчанию обычно Times New Roman);
- **sans-serif** — для шрифтов без засечек, или рубленых шрифтов(обычно Arial);
- **monospace** — для моноширинных шрифтов, в которых символы имеют одинаковую ширину (обычно Consolas или Courier New);

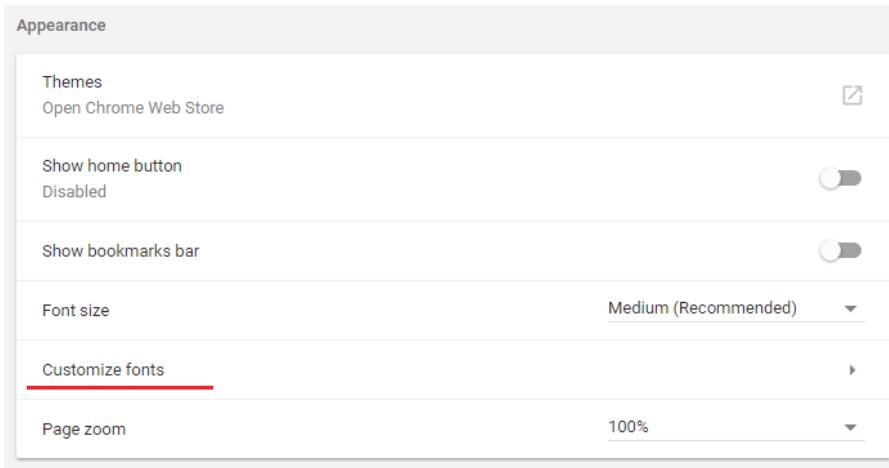
- **cursive** — для курсивных шрифтов (как вариант — ComicSans MS);
- **fantasy** — для фантазийных, или декоративных шрифтов (представитель — Impact).

Принято указывать шрифт-псевдоним последним, после всех более подходящих вариантов. Но, если вас, например, вполне устраивает стандартный шрифт без засечек Arial (а веб-страницы обычно используют именно рубленые шрифты), то вы можете сократить запись свойства:

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

Для тестов вы можете использовать файл `font-family-test.html` (*прикреплен к PDF-файлу данного урока*).

Посмотреть, какие шрифты используются по умолчанию, можно в настройках браузера. Например, для Хрома, они выглядят так:



Standard font

Times New Roman ▾

16: The quick brown fox jumps over the lazy dog

Serif font

Times New Roman ▾

16: The quick brown fox jumps over the lazy dog

Sans-serif font

Arial ▾

16: The quick brown fox jumps over the lazy dog

Fixed-width font

Consolas ▾

The quick brown fox jumps over the lazy dog

► **Примечание:** если вы используете шрифт, название которого состоит из 2-3 слов, следует указывать имя шрифта в одинарных или двойных кавычках:

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

2. Стиль шрифта задается свойством **font-style**. Значения: normal | italic | oblique | inherit. Первое зна-

чение — **normal** — это значение по умолчанию, т.е. обычное начертание текста. Значение **italic** — это курсивное начертание, которое предполагает наличие курсивного шрифта в системе пользователя, **oblique** — наклонное начертание шрифта, **inherit** — наследует значение родительского элемента.

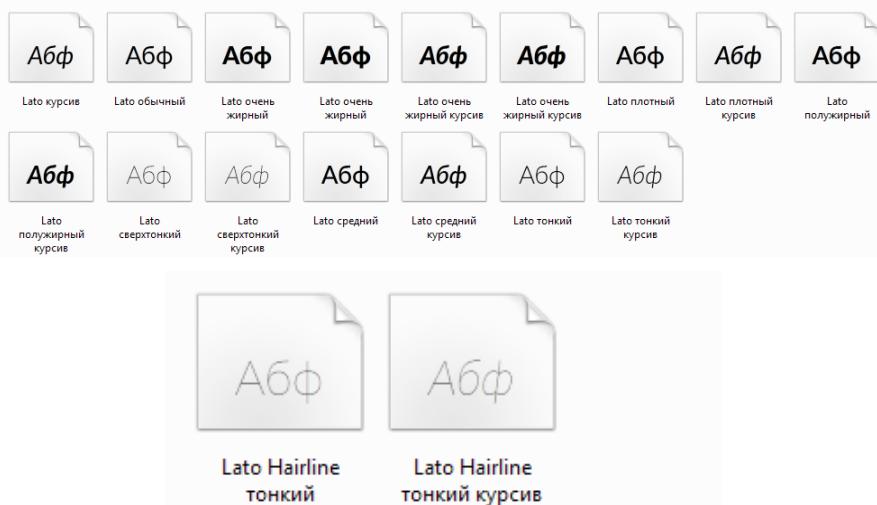
Аббревиатура Emmet: `fs`, `fsi`, `fso`, `fsn`

3. За насыщенность шрифта отвечает свойство **font-weight**, которое имеет значения **normal | bold | bolder | lighter | 100 | 200 ... 800 | 900 | inherit**. По умолчанию используется значение **normal** — обычная насыщенность шрифта, которая соответствует файлам шрифта, в название которых входит слово Regular. Соответственно, значение **bold** отвечает за полужирное начертание. Значения **bolder** и **lighter** изменяют жирность шрифта в большую или меньшую сторону относительно значения этого свойства у родителя. Единицы от 100 до 900 указываются с шагом в 100 и распределяются так: от 100 до 300 — это, как правило, тонкий шрифт, 400-500 — обычное начертание, от 600 — полужирное начертание. Если рассматривать их подробнее, то начертание в единицах имеет следующие значения:

- 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)

Реальное отображение шрифтов на странице очень сильно зависит от того, сколько файлов шрифта составляют выбранный вами шрифт для страницы или элемента. Например, шрифт Lato, который установлен у меня для дизайнерских целей, и, скорей всего отсутствует на вашей машине, имеет следующее количество файлов шрифта:



Поэтому для этого шрифта можно выбирать различные значения **font-weight**, и они будут отличаться друг от друга. Для других же шрифтов разница будет куда менее очевидна, т.к. они могут состоять из 1-3-5 файлов.

Вы можете посмотреть различные варианты начертания, заданные с помощью свойства `font-weight` в файле

font-weight-test.html (*прикреплен к PDF-файлу данного урока*), но для этого вам необходимо скачать шрифт Lato и установить его к себе в систему. Делается это обычным копированием файлов шрифтов, например, в формате ttf, в системную папку 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;
}

```

**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. Voluptates cumque suscipit dolorum, dignissimos ipsum praesentium nihil officiis obcaecati enim culpa a reprehenderit ullam quibusdam non accusantium molestias hic.

font-weight: 600. Tempora officia accusantum, 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 possumus 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 incident 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.

В файле font-weight-test.html (*прикреплен к PDF-файлу данного урока*) в самом низу использованы вложенные элементы в <div class="parent"> со значениями свойства font-weight bolder и lighter. На скриншоте видно, что по отношению к абзацу с основным текстом (Main text), верхний абзац имеет более тонкие символы, а нижний — более жирные.

#### Font-weight: lighter, bolder

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

**Main text.** Distinctio veniam consectetur aspernatur tenetur facere. Totam quo enim, numquam.

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

Аббревиатура Emmet: fw, fwb, fwbr, fwlr, fwn, fw200.

4. Размер шрифта **font-size**. Указывается он в различных единицах: *cm*, *mm*, *pc*, *px*, *pt*, *em*, *rem*, *%*, реже — в *ex*, *vh* или *vw*, а также в абсолютном и относительном размере. Еще размер шрифта может иметь значение *inherit*, т.е. унаследованный от родителя. Для указания абсолютного размера используются ключевые слова **xx-small**, **x-small**, **small**, **medium**, **large**, **x-large**, **xx-large**. Для относительного размера шрифта используются значения **larger** и **smaller**. Размер шрифта родительского элемента берется за 100%. Нельзя использовать отрицательные значения.

### Абсолютный размер

**font-size: xx-small** *Lorem ipsum dolor sit amet, consectetur adipisciing elit. Iusto nisi illum ratione ipsum reprehenderit dignissimos!*

**font-size: small** *Aliquam iure doloremque nesciunt veritatis animi soluta, delectus, facere, totam molestiae eligendi doloribus magni reiciendis!*

**font-size: medium** *Nobis, saepe deleniti. Error voluptate quis iusto nemo alias magnam, vel autem obcaecati, quo temporibus!*

**font-size: large** *Repellendus aspernatur voluptas quibusdam eius, consequuntur natus tempora voluptatibus neque ad excepturi repellat. Quas, pariatur.*

**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!*

## Относительный размер

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

**Исходный размер**  *Quos atque autem praesentium repellat amet consequatur dignissimos, veritatis nulla quam aliquid sit et quaerat suscipit perferendis, repudianda 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.*

Варианты назначения размера шрифта представлены в файле font-size-test.html (*прикреплен к PDF-файлу данного урока*). Нужно заметить, что сейчас чаще всего указывают размер шрифта в px, em, rem и в %. Все остальные варианты используются редко.

Аббревиатура Emmet: fz, fz:15, fz:120%

5. Варианты начертания строчных букв — свойство **font-variant** со значениями **normal** | **small-caps** | **inherit**. Нормальное начертание (**normal**) — это обычный вид букв, и это значение по умолчанию, а **small-caps**, или капитель, все строчные символы отображает в виде заглавных, но уменьшенного размера. Значение **inherit** указывает на то, что это свойство наследуется от родительского элемента.

В нижней части файла font-weight-test.html вы найдете пример свойства **font-variant** со значением **small-caps**:

**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!*

Аббревиатура Emmet: fv, fvs

6. Высота строки, или интерлиньяж, или межстрочный интервал — свойство **line-height**. Это свойство не относится напрямую к группе fonts-, но связано с этими свойствами, т.к. сложно представить хорошо отформатированный текст без указания высоты строки. В качестве значений этого свойства можно задать множитель | значение в любых единицах | проценты | normal | inherit. Обычно высота строки задается не менее 120% или множителем 1.2, но еще она зависит от самого шрифта и вычисляется автоматически (значение **normal**). Отрицательные значения не допускаются. Чаще всего интерлиньяж указывается в виде множителя или в процентах. Посмотреть на варианты отображения различных значений этого свойства можно в файле line-height-test.html (*прикреплен к PDF-файлу данного урока*). Еще стоит почитать эту статью — [Свойства «font-size» и «line-height».](#)

Аббревиатура Emmet: lh, lh:n, lh:1.5, lh:140%

7. Универсальное свойство **font**, которое позволяет указать сразу несколько характеристик шрифта, задаваемых ранее перечисленными свойствами. Все значения составляющих его свойств перечисляются через пробел. Исключение составляет размер шрифта и высота строки — они указываются через слэш. Синтаксис этого свойства таков:

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

Обязательными в этом свойстве являются размер и семейство шрифта. Все остальные свойства задаются по мере необходимости в том порядке, который представлен выше. Со свойством **font-stretch** вам придется познакомиться самостоятельно, т.к. оно не слишком часто используется.

Следует отметить, что все свойства, которые вы не указали, приобретают свой значение по умолчанию.

Протестировать некоторые варианты использования свойства font можно в файле font-test.html (*прикреплен к PDF-файлу данного урока*).

## Выравнивание текста

Свойство **text-align** необходимо для выравнивания текста в блочном элементе по левому краю (**left** — значение по умолчанию), правому краю (**right**), по центру (**center**) или ширине (**justify**). И именно его стоит использовать, когда вы хотите центрировать текст вместо устаревшего тега <center>.

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

Новые значения **start** и **end** появились в спецификации CSS 3.0 и обозначают следующее:

- **start** — работает так же, как и **left**, если текст идёт слева направо, и как **right**, если текст идёт справа налево;
- **end** — работает так же, как и **right**, если текст идёт слева направо, и как **left**, если текст идёт справа налево;

Протестируйте выравнивание текста с помощью файла `text-align-test.html` (*прикреплен к PDF-файлу урока*).

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

### Единицы измерений в CSS

**Важное замечание:** любые единицы измерений пишутся слитно с цифрами. Т.е. нельзя поставить цифру, пробел, а потом указать единицу измерения.

**Неверно: 16 px. Верно: 16px**

В CSS существуют абсолютные и относительные единицы измерений.

Что касается абсолютных единиц, то в CSS можно использовать привычные нам со школы сантиметры (**cm**) и миллиметры (**mm**), менее привычные дюймы (**in**), характерные для типографий пункты (**pt** — 1/72 дюйма) и пики (**pc** — 1/6 дюйма). Но стоит отметить, что в реальной верстке используются они крайне редко. Зато еще одна абсолютная единица — пиксель, или **px**, — используется часто, т.к. именно в пикселях измеряется разрешение монитора или другого экрана, и браузер также многие величины пересчитывает в пикселях.

А вот относительные единицы используются очень часто. Пожалуй, самыми часто используемыми единицами являются **em** и относительно недавно появившийся **rem**, а также проценты (%). Что в них общего? **1em** и **100%** — это размер шрифта родительского элемента. Т.е., если для `body` записано такое правило:

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

А для заголовка первого уровня такое:

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

То браузер рассчитает размер заголовка в пикселях, отталкиваясь от размера шрифта его родителя (т.е. body), а именно:

$$3 * 14\text{px} = 42\text{px}$$

Если же для h1 будет записано правило:

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

То расчет будет аналогичным:

$$14\text{px} * 300\% / 100\% = 14\text{px} * 3 = 42\text{px}$$

Т.е. и em, и % отталкиваются от размера шрифта родительского элемента. Имейте в виду, что шрифтовые свойства наследуются по цепочке. Т.е., если h1 находится в `<div class="logo">`, например, а не напрямую в body, и для `<div class="logo">` не задан размер шрифта, то для div-а шрифт тоже будет составлять 14px, а заголовок будет рассчитан по той же схеме. А вот, если будет задано правило:

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

То шрифт заголовка будет следующего размера:

$$14\text{px} \text{ (от body)} * 1.3 \text{ (для .logo)} * 3 \text{ (для h1)} = 54.6\text{px}$$

И такой же расчет в %, только нужно еще на 100 разделить.

Единица измерения **1rem (root em)** указывает на размер шрифта, который равен размеру, указанному для шрифта корневого элемента разметки — тега `<html>`. Этот размер устанавливается либо в браузере пользователя (обычно это `16px`), либо явно задается для `html`-страницы. И все расчеты производятся относительно этого размера.

Еще одна относительная единица — **1ex** — отталкивается от высоты символа «x» шрифта элемента в нижнем регистре. Но при наличии более удобной единицы **em**, реально практически не используется.

На данный момент часто используют такие единицы, как **1vw** (1% viewport width) **1vh** (1% viewport height) рассчитываются относительно размеров области просмотра — ширины или высоты, соответственно. И используются они чаще не для указания размеров шрифта, а для установки таких CSS-свойств, как `width` и `height`, о которых мы поговорим в следующих уроках. Также можно использовать единицы **vmin** или **vmax**, которые обозначают меньшее или большее значение из `vw` и `vh`.

Почитать о единицах измерений:

- <https://learn.javascript.ru/css-units>
- <http://htmlbook.ru/content/edinitisy-izmereniya>
- <http://htmlbook.ru/css/value/size>
- <http://html-plus.in.ua/edinicy-izmereniy-v-css/>

# Домашнее задание

Сегодня у вас будет творческое домашнее задание: вам необходимо написать свою автобиографию или свое резюме, используя те теги и css-свойства, которые были разобраны в этом уроке.

Не забудьте разбить текст на заголовки и абзацы, выделить важные вещи тегами `<strong>` или `<em>`. Возможно, вы используете `<hr>` или `<blockquote>`. Или теги `<div>` или `<span>` с классами. И css-форматирование.

Как пример, приведу автобиографию одной из моих студенток. Юмор и самоирония приветствуются ☺. Как и грамотность + стиль.

## Прекрасная жизнь Татьяны: от рождения до настоящего времени

### Мои детские годы

Родилась я 38 лет назад. Мое детство было вполне себе счастливое. У меня было все, о чем мечтали дети тех времен. Кроме килограммов пирожных и зефиров, постоянно имеющихся в доме, у меня было кое-что поинтереснее. На свое восемнадцатилетие я получила электронную игру "Микки Маус ловит яйца". А еще блок жвачек "Donald Duck". Что уж там говорить, в то время я была звездой улицы. Ну, во всяком случае, я таковой себя ощущала.

### Краткое изложение жизни от А до Б

В остальном вполне обычна, если даже не банальная, жизнь. Школа, техникум, ВУЗ, замужество, рождение сына, осознание того, что где-то я свернула не туда, развод. А, еще карьера. Пытаясь искать работу по специальности, а в красненьком дипломе она значится у меня как "Бухгалтерский учет и аудит", я попала вообще в другую отрасль. Я стала оценщиком. Что могу сказать: работа интересная, разнообразная, не скучно было, так точно. 10 лет жизни ушло на осмотр, поиск аналогов и формирование отчетов по оценке недвижимости, товаров, оборудования и прочих интересных штук.

### Еще один этап моего трудового пути

Некоторые причины личного характера привели к тому, что с оценкой я благополучно попрощалась.

**Это примерно как разводишься с футбольистом, и больше терпеть не можешь футбол.**

В общем, дальше я отправилась покорять просторы страхового бизнеса. Точнее, попала в отдел по урегулированию убытков в отличной компании "Альфа Страхование". Все бы ничего, но спустя 2 года я оказалась в числе тех, кого сократили. Это была так называемая минимизация затрат, причина которой - нынешнее нелегкое положение в стране.

### Как я дошла до жизни такой

А еще через год поняла, что не готова больше тратить свое время на не приносящую нормального дохода работу. Так я попала в копирайтинг. Где мне снизилось озарение, что я хочу научиться делать сайты, а не только наполнять их текстовым контентом. **Поэтому я здесь;-)**



## Урок 1. Введение. Структура HTML

© Слуцкая Елена.  
© STEP IT Academy, [www.itstep.org](http://www.itstep.org).

All rights to protected pictures, audio, and video belong to their authors or legal owners.

Fragments of works are used exclusively in illustration purposes to the extent justified by the purpose as part of an educational process and for educational purposes in accordance with Article 1273 Sec. 4 of the Civil Code of the Russian Federation and Articles 21 and 23 of the Law of Ukraine "On Copyright and Related Rights". The extent and method of cited works are in conformity with the standards, do not conflict with a normal exploitation of the work, and do not prejudice the legitimate interests of the authors and rightholders. Cited fragments of works can be replaced with alternative, non-protected analogs, and as such correspond the criteria of fair use.

All rights reserved. Any reproduction, in whole or in part, is prohibited. Agreement of the use of works and their fragments is carried out with the authors and other right owners. Materials from this document can be used only with resource link.

Liability for unauthorized copying and commercial use of materials is defined according to the current legislation of Ukraine.