

Применяйте для отступов пробелы, не используйте табуляцию или сочетания пробелов и табуляции.


```
<div class="parent">
  <div class="child">
  </div>
</div>
```

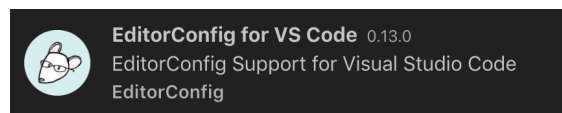
**Почему:** отступы в два пробела делают код компактнее. Такое форматирование чаще используется в стандартах IT-компаний.

В большинстве редакторов кода удобно нажимать клавишу Tab для создания вложенности. Этот символ интерпретируется как табуляция.

Чтобы изменить эту настройку и установить для клавиши Tab отступ в два пробела, в редакторе Visual Studio Code можно использовать специальные плагины. Самый популярный из них — [Editorconfig](#).

Процесс настройки параметров Visual Studio Code описан в [официальном справочнике](#).

На практике вам достаточно установить сам плагин, найдя его в пункте меню "Extentions"  по имени Editor Config for VS Code



После установки плагина поместите в корень проекта файл конфигурации с именем `.editorconfig` с таким содержимым:

```
# http://editorconfig.org
```

```
# A special property that should be specified at the top of the file outsi
```

```
# any sections. Set to true to stop .editor config file search on current
```

```
root = true
```

```
[*]
```

```
# Indentation style
```

```
# Indentation size in single-spaced characters

# Possible values – an integer, tab

indent_size = 2

# Line ending file format

# Possible values – lf, crlf, cr

end_of_line = lf

# File character encoding

# Possible values – latin1, utf-8, utf-16be, utf-16le

charset = utf-8

# Denotes whether to trim whitespace at the end of lines

# Possible values – true, false

trim_trailing_whitespace = true

# Denotes whether file should end with a newline

# Possible values – true, false

insert_final_newline = true
```

## Применяйте кодировку UTF-8

Установите кодировку UTF-8 в HTML шаблонах и документах:

`<meta charset="utf-8">` Сохраняйте файлы шаблонов и документов в той же кодировке (чаще всего она устанавливается автоматически).

Не нужно специально прописывать кодировку для таблиц стилей: им по умолчанию присваивается UTF-8. Это универсальная кодировка, её поддерживают все современные браузеры и приложения.

## Используйте только строчные символы

Весь код (как в HTML, так и в CSS) должен быть написан строчными символами.

Это касается имён HTML-элементов, атрибутов и их значений (за исключением text/CDATA), CSS-селекторов, свойств и значений свойств (за исключением строк, например, в значениях атрибутов `alt` и `title`).

school".

```
<!-- Не рекомендуется -->
```

```
<A HREF="/">Home</A>
```

```
<!-- Рекомендуется -->
```

```

```

```
/* Не рекомендуется */  
color: #E5E5E5;
```

```
/* Рекомендуется */  
color: #e5e5e5;
```

## Протокол

Если для получения внедрённых в документ элементов (медиафайлов, таблиц стилей и скриптов) доступен протокол HTTPS — используйте именно его. Когда страница, полученная по безопасному протоколу HTTPS, подгружает элементы по незащищенному протоколу HTTP, браузер сообщает пользователю о нарушении безопасности. Осторожный пользователь такую страницу покинет.

```
<!-- Не рекомендуется: протокол не указан -->
```

```
<script src="//ajax.googleapis.com/ajax/libs/jquery/3.4.0/jquery.min.js":
```

```
<!--Не рекомендуется: используется протокол HTTP при доступном HTTPS -->
```

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/3.4.0/jquery.mi
```

```
<!-- Рекомендуется -->
```

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.0/jquery.m
```



```
/* Не рекомендуется: протокол не указан */
```

```
@import url('//fonts.googleapis.com/css?family=Open+Sans');
```

*/\* Рекомендуется \*/*

```
@import url('https://fonts.googleapis.com/css?family=Open+Sans');
```

## Не вызывайте ненужное

Не загружайте неиспользуемые скрипты и стили. Это увеличивает время загрузки страницы.

*<!-- Не рекомендуется -->*

```
<!DOCTYPE html>
<html>
<head>
  <title>На этой странице нет формы</title>
  <link rel="stylesheet" href="form.css" media="screen">
  <link rel="stylesheet" href="default.css" media="screen">
  <script src="form.js"></script>
  <script src="default.js"></script>
</head>
<body>
  <h1>На этой странице нет формы</h1>
  <p>Но зачем-то сюда загружаются два лишних файла, управляющие отображе
</body>
</html>
```

*<!-- Рекомендуется -->*

```
<!DOCTYPE html>
<html>
<head>
  <title>На этой странице нет формы</title>
  <link rel="stylesheet" href="default.css">
  <script src="default.js"></script>
</head>
<body>
  <h1>На этой странице нет формы</h1>
  <p>И нет ничего лишнего в списке подгружаемых файлов</p>
</body>
</html>
```



## Удаляйте пробелы в конце строк

Пробелы в конце строк не нужны и могут усложнить операции с кодом.

*<!-- Не рекомендуется -->*

<p>Спасибо, не нужно.</p>

## Комментируйте свой код

Комментарии предназначены для разъяснения или структуризации кода. С вашим кодом будут работать другие программисты, они должны понимать ход вашей мысли.

Описывайте назначение элементов, отмечайте начало и конец логических блоков — это сделает код удобным для чтения и дальнейшей работы.

Комментарии можно писать как латиницей, так и кириллицей.

## Ведите список задач по коду

Вопросы, требующие принятия решения, отмечайте комментарием `TODO`.

Не используйте другие варианты, например: `@@`. Чтобы добавить контакты человека, который должен принять решение по конкретному вопросу, используйте формат `TODO(user@mail.ltd)`, в скобках указывайте имя пользователя или адрес электронной почты.

## ② HTML

### Используйте HTML5

При отсутствии особых требований для HTML-документов лучше использовать HTML5. Объявление HTML5 в коде выглядит так: `<!DOCTYPE html>`.

Рекомендуется использовать HTML. Не используйте XHTML: он хуже поддерживается браузерами и прочей инфраструктурой, предлагает меньше возможностей для оптимизации.

### Валидируйте свой HTML код

Проверяйте свой код на валидность, для этого есть специальные инструменты, например – [W3C HTML validator](#).

Валидация поможет вам понять технические требования и ограничения при написании кода, обнаружить и исправить структурные и синтаксические ошибки.

### Грамотно используйте семантические элементы

качество автоматического разбора страницы.

```
<!-- Не рекомендуется -->
```

```
<div onclick="goToRecommendations();">All recommendations</div>
```

```
<!-- Рекомендуется -->
```

```
<a href="/recommendations">All recommendations</a>
```

## Устанавливайте альтернативный текст для мультимедиа

Описывайте текстом мультимедийные элементы (изображения, видео, *canvas*), используйте атрибуты `alt` и `title`.

Для изображений пишите содержательный альтернативный текст (*alt*), для видео и аудио создавайте расшифровку и субтитры, если это возможно.

Альтернативный контент важен для доступности ваших веб-страниц: незрячий пользователь не поймёт содержание изображения без значения атрибута *alt*. Некоторые пользователи не смогут разобраться в смысле аудио или видео без текстового разъяснения.

Для изображений, которым не нужен альтернативный текст (например, декоративных), используйте пустой атрибут *alt*: `alt=""`.

```
<!-- Не рекомендуется -->
```

```

```

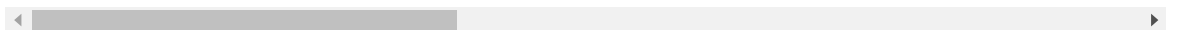
```
<!-- Рекомендуется -->
```

```
*

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML – шляпа</title>
  <link rel="stylesheet" href="default.css" media="screen">
</head>
<body>
  <h1 style="font-size: 1em;">HTML – шляпа</h1>
  <p>Я уже читал об этом на форумах, но теперь мне окончательно ясно:</p>
  <u>HTML – глупый язык!!1</u>
  <center>Поверить не могу, что я не могу изменить внешний вид своего са
</body>
</html>
```

*<!-- Рекомендуется -->*

```
<!DOCTYPE html>
<html>
<head>
  <title>Мой первый редизайн, основанный только на CSS</title>
  <link rel="stylesheet" href="default.css">
</head>
<body>
  <h1>Мой первый редизайн, основанный только на CSS</h1>
  <p>Раньше я только читал об этом на форумах, но теперь я наконец взялс
  <p>Как же это круто!</p>
</body>
</html>
```



## При использовании UTF-8 не используйте ссылки на HTML-сущности

Если работающая над проектом команда во всех файлах и редакторах использует одну и ту же кодировку UTF-8, не нужно ссылок на сущности вроде `& m d a s h;`, `& r d q u o;`, или `& # x 2 6 3 a`.

Исключения — символы с особым значением для HTML. В первую очередь < как `& l t;`, > как `& g t;`, & как `& a m p;` ) и неразрывные пробелы `& n b s p;`.

*<!-- Не рекомендуется -->*

Символ валюты Евро — `& l d q u o ; & e u r ; & r d q u o ;`

## Не используйте атрибут type для связанных файлов CSS и JS

Прописывать атрибуты *type* для связанных таблиц стилей и файлов JS нет необходимости, так как HTML5 присваивает типы *text/css* и *text/javascript* по умолчанию. Это работает даже в старых версиях браузеров.

*<!-- Не рекомендуется: использован атрибут type -->*

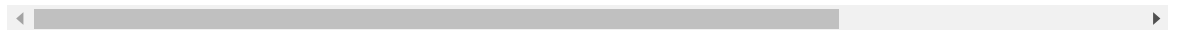
```
<link rel="stylesheet" href="https://www.google.com/css/maia.css" type="text/css">  
<!-- Рекомендуется -->
```

```
<link rel="stylesheet" href="https://www.google.com/css/maia.css">
```

*<!-- Не рекомендуется: использован атрибут type -->*

```
<script src="https://www.google.com/js/gweb/analytics/autotrack.js" type="text/javascript">  
<!-- Рекомендуется -->
```

```
<script src="https://www.google.com/js/gweb/analytics/autotrack.js"></script>
```



## Пишите с новой строки и с отступами

Пишите с новой строки каждый новый элемент: блок, список или таблицу, делайте отступ перед каждым новым дочерним элементом. Это упростит чтение кода

## Переносите длинные строки в HTML

Хотя в языке HTML нет рекомендации, ограничивающей длину строк, длинные строки стоит разбивать — такой код станет проще читать.

При переносе строк их продолжения должны иметь отступ минимум в два пробела относительно начальной строки.

```
<p>При переносе строк их продолжения должны иметь отступ минимум в два пробела относительно начальной строки.</p>
```

```
<a  
  href="https://yandex.ru"  
  class="my-wonderful-link"  
  target="_blank"  
  id="myId">
```



## Используйте в HTML двойные кавычки

Заклю­чайте значения атрибутов в двойные (""), а не одинарные (') кавычки. Это упростит взаимодействие с CSS и JS, улучшит читаемость кода. Когда в HTML-разметке кавычки двойные, а в коде CSS и скриптов одинарные, легче различать области разных языков.

*<!-- Не рекомендуется: использованы одинарные кавычки -->*

```
<a class='maia-button maia-button-secondary'>Войти </a>
```

*<!-- Рекомендуется -->*

```
<a class="maia-button maia-button-secondary">Войти</a>
```

## Не закрывайте одиночные теги

Пишите `<br>` и `<img ...>`, а не `<br />` или `<img ... />` Технической необходимости в этом нет.

## ③ CSS

### Валидируйте CSS-код

Пишите валидный CSS-код, если это возможно. Для проверки валидности используйте инструмент [W3C CSS validator](#).

Валидность гарантирует предсказуемое и корректное отображение страницы в браузере.

### Давайте классам и идентификаторам осмысленные названия

Используйте имена, отражающие назначение блоков и элементов. Это делает код более читаемым и понятным. Для работы с CSS используйте селекторы классов.

Пользуйтесь методологией [БЭМ](#). Она позволяет повторно использовать целые блоки кода.

*/\* Не рекомендуется: бессмысленное имя \*/*

```
.abyrvalg {}
```

*/\* Рекомендуется: конкретное описательное имя \*/*

```
.gallery {}
```

## Не назначайте стили элементам и ID, используйте классы

Не задавайте стили для имён элементов HTML или сочетаний этих имён с идентификаторами и классами. Это излишне расширяет или излишне ограничивает область применения стилей.

*/\* Не рекомендуется (установлен стиль для элемента): \*/*

```
div {}
```

*/\* Не рекомендуется (установлен стиль для элемента с классом): \*/*

```
div.error {}
```

*/\* Рекомендуется: \*/*

```
.error {}
```

Без исключительной необходимости не назначайте стили элементам по ID, избегайте конструкции `#id_name{}`, используйте

```
.class_name{}
```

## Не ставьте единицы измерения для нулевых значений

Не указывайте единицы измерения после нулевых значений, кроме случаев, когда это важно для кроссбраузерности:

*/\* Не рекомендуется: можно написать короче \*/*

```
margin: 0px;  
padding: 0px;
```

*/\* Рекомендуется \*/*

```
margin: 0;  
padding: 0;
```

## Не ставьте «ноль целых» в десятичных дробях

Не ставьте 0 перед значениями в диапазоне от -1 до 1.

```
font-size: 0.8em;
```

```
/* Рекомендуется */
```

```
font-size: .8em;
```

## Сокращайте запись шестнадцатеричных чисел

По возможности используйте трёхсимвольную шестнадцатеричную систему для указания цветовых кодов — это лаконичней.

```
/* Не рекомендуется: можно написать короче */
```

```
color: #eebbcc;
```

```
/* Рекомендуется */
```

```
color: #ebc;
```

## Делайте отступ перед блочным контентом

Перед любым блочным контентом (правилами внутри правил, объявлениями) делайте отступы. Это улучшит читаемость кода.

```
@media screen, projection {  
  html {  
    background: #fff;  
    color: #444;  
  }  
}
```

## Соблюдайте пунктуацию

После каждого объявления ставьте точку с запятой. Если что, валидатор вас поправит. Отсутствие пунктуации — нарушение стандартов W3.org, техническая ошибка.

```
/* Не рекомендуется: забыли точку с запятой */
```

```
.test {  
  display: block;  
  height: 100px
```

```
.test {  
  display: block;  
  height: 100px;  
}
```

## Разделяйте пробелом свойство и значение

Всегда ставьте одиночный пробел между свойством и значением, но не между свойством и двоеточием.

*/\* Не рекомендуется: нет пробела после двоеточия \*/*

```
h3 {  
  font-weight:bold;  
}
```

*/\* Рекомендуется \*/*

```
h3 {  
  font-weight: bold;  
}
```

## Ставьте пробел перед открывающей фигурной скобкой

Всегда ставьте пробел между селектором и открывающей фигурной скобкой перед блоком объявления.

В каждом правиле открывающая фигурная скобка должна стоять на той же строке, что и селектор

*/\* Не рекомендуется: не хватает пробела \*/*

```
.video{  
  margin-top: 1em;  
}
```

*/\* Рекомендуется \*/*

```
.video {  
  margin-top: 1em;  
}
```

## Отделяйте объявления стилей

Каждое объявление стилей пишите на отдельной строке:

```
a:tocus {
    position: relative; top: 1px;
}

/* Рекомендуется */

h1 {
    font-weight: normal;
    line-height: 1.2;
}
```

## Разделение правила

Всегда оставляйте пустую строку (двойной переход на новую строку) между правилами.

```
html {
    background: #fff;
}

body {
    margin: auto;
    width: 50%;
}
```

## Используйте в CSS одинарные кавычки

Значения атрибутов и свойств, в том числе в селекторах, пишите с одинарными кавычками ('). Исключение — правило `@charset`. Для него применяйте двойные кавычки — одинарные недопустимы. URL указывайте без кавычек.

```
/* Не рекомендуется: кавычки в URL */

@import url("https://www.google.com/css/maia.css");

/* Не рекомендуется: двойные кавычки */

html {
    font-family: "open sans", arial, sans-serif;
}

/* Рекомендуется */

@import url(https://www.google.com/css/maia.css);

html {
    font-family: 'open sans', arial, sans-serif;
```

## Разделяйте код на тематические разделы

Разбивайте код на разделы, в комментариях задавайте разделам осмысленные заголовки. Отделяйте разделы пустыми строками.

```
/* Header */
```

```
.header {  
}
```

```
/* Gallery */
```

```
.gallery {  
}
```

```
/* Footer */
```

```
.footer {  
}
```

## Отбросьте «костыли»

Избегайте определения типа браузера или клиентского приложения (user agent detection), а также прочих «костылей» — CSS “hacks”. Ищите другие методы решения задач кроссбраузерности.

Методы определения клиентского приложения, специальные CSS-фильтры и прочие уловки стоит считать крайней мерой и ограничивать их применение, если вы хотите создать эффективный и управляемый код.

Использование «хаков» вызывает привыкание. Ваш код становится складом костылей, теряет управляемость и предсказуемость.

## Послесловие

### Обязательность приведенных выше правил

Описанные правила — не технические ограничения и не стандарты. Их нарушение не ведёт к ошибке интерпретации. Это наши внутренние правила оформления кода.

## Работа с чужим кодом

Сохраняйте стиль исходного кода.

Перед редактированием определите формат кода, с которым вам предстоит работать.

Изучите принципы, которыми руководствовался автор, и следуйте им, даже если они расходятся с вашими. Так вы сохраните логику и читаемость кода.

Ваш код не должен значительно отличаться от оригинального по форматированию. Различия усложнят понимание кода для любого, кто будет читать его после вас.

Мы написали правила форматирования кода для того, чтобы наши разработчики и студенты говорили на одном диалекте языка HTML. Стандарты других команд могут значительно отличаться от наших.