



Kirill Cherkashin @z6Dabrata
Пользователь

89,5 карма
0,0 рейтинг

Профиль	7 Публикации	98 Комментарии	1 Избранное	12 Подписчики
---------	--------------	----------------	-------------	---------------

8 мая 2012 в 08:35

Руководство по оформлению HTML/CSS кода от Google перевод

Веб-разработка*, Веб-дизайн*, CSS*

От переводчика

С удовольствием ознакомился с этими рекомендациями и теперь предлагаю вам перевод.

Введение

Это руководство описывает правила для оформления и форматирования HTML и CSS кода. Его цель — повысить качество кода и облегчить совместную работу и поддержку инфраструктуры.

Это относится к рабочим версиям файлов использующих **HTML**, **CSS** и **GSS**

Разрешается использовать любые инструменты для минификации компиляции или обфускации кода, при условии, что общее качество кода будет сохранено.

Общие правила оформления

Протокол

Не указывайте протокол при включении ресурсов на страницу.

Опускайте название протокола (**http:**, **https:**) в ссылках на картинки или другие медиа-ресурсы, файлы стилей или скрипты, конечно, если эти файлы доступны по обоим протоколам.

Отсутствие протокола делает ссылку относительной, что предотвращает смешивание ресурсов из разных протоколов и незначительно уменьшает размер файлов.

Не рекомендуется:

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

Рекомендуется:

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

Не рекомендуется:

```
.example {  
  background: url(http://www.google.com/images/example);  
}
```

Рекомендуется:

```
.example {  
  background: url(//www.google.com/images/example);  
}
```

Общее форматирование

Отступы

Всегда используйте для отступа два пробела.

Не используйте табуляцию и не смешивайте табуляцию с пробелами.

Рекомендуется:

```
<ul>
  <li>Пять
  <li>Погулять
</ul>
```

Рекомендуется:

```
.example {
  color: blue;
}
```

Регистр
Всегда пишите в нижнем регистре.

Весь код должен быть написан в нижнем регистре: Это относится к названиям элементов, названиям атрибутов, значениям атрибутов (кроме текста/**CDATA**), селекторам, свойствам и их значениям (кроме текста).

Не рекомендуется:

```
<A HREF="/">Домой</A>
```

Рекомендуется:

```

```

Пробелы в конце строки
Убирайте пробелы в конце строки.

Пробелы в конце строк не обязательны и усложняют использование diff.

Не рекомендуется:

```
<p>Что?_
```

Рекомендуется:

```
<p>Вот так.
```

Общие мета правила

Кодировка
Используйте UTF-8 (без BOM).

Убедитесь, что ваш редактор использует кодировку UTF-8 без метки порядка байтов (BOM).

Указывайте кодировку в HTML шаблонах и документах с помощью **<meta charset="utf-8">**. Опускайте кодировку для css-файлов: для них UTF-8 задана по умолчанию.

(Вы можете узнать больше о кодировках, и о том, как их использовать, по этой ссылке: [Наборы символов и кодировки в XHTML, HTML CSS \(англ.\)](#).)

Комментарии

По возможности поясняйте свой код, где это необходимо.

Используйте комментарии, чтобы пояснить свой код: что он делает, за что отвечает, и почему используется выбранное решение.

(Этот пункт не обязателен, потому что нет смысла ожидать, что код всегда будет хорошо задокументирован. Полезность комментирования зависит от сложности проекта и может различаться для HTML и CSS кода.)

Задачи

Отмечайте задачи для списка дел с помощью `TODO`.

Отмечайте задачи с помощью ключевого слова `TODO`. не используйте другие часто встречающиеся форматы, такие как `@@`.

Заклучайте контакты (имя пользователя или список адресатов) в круглые скобки: `TODO(контакт)`.

Описывайте задачу после двоеточия, например: `TODO: Задача`.

Рекомендуется:

```
{# TODO(Ivan Ivanov): Разобраться с центровкой #}  
<center>Тест</center>
```

Рекомендуется:

```
<!-- TODO: Убрать необязательные теги -->  
<ul>  
  <li>Огурцы</li>  
  <li>Помидоры</li>  
</ul>
```

Правила оформления HTML

Тип документа

Используйте HTML5.

HTML5 (HTML синтаксис) рекомендуется для всех html-документов: `<!DOCTYPE html>`.

(Рекомендуется использовать HTML с типом контента `text/html`. Не используйте XHTML, так как `application/xhtml+xml` (англ.), хуже поддерживается браузерами и ограничивает возможность оптимизации.)

Валидность HTML

По возможности используйте валидный HTML.

Используйте валидный HTML код, кроме случаев, когда использование не позволяет достичь размера файла, необходимого для нужного уровня производительности.

Используйте такие инструменты как [W3C HTML validator](#) (англ.) чтобы проверить валидность кода.

Валидность — это важное и при этом измеряемое качество кода. Написание валидного HTML способствует изучению технических требований и ограничений и обеспечивает правильное использование HTML.

Не рекомендуется:

```
<title>Проверка</title>  
<article>Просто проверка
```

Рекомендуется:

```
<!DOCTYPE html>  
<meta charset="utf-8">
```

```
<title>Проверка</title>
<article>Просто проверка.</article>
```

Семантика

Используйте HTML так, как это было задумано.

Используйте элементы (Иногда неверно называемые “тегами”) по назначению: заголовки для заголовков, **p** для абзацев, **a** для ссылок и т.д.

Это облегчает чтение, редактирование и поддержку кода.

Не рекомендуется:

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

Рекомендуется:

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

Альтернатива для мультимедиа

Всегда указывайте альтернативное содержимое для мультимедиа.

Постарайтесь указать альтернативное содержимое для мультимедиа: например для картинок, видео или анимаций, заданных с помощью **canvas**. Для картинок это осмысленный альтернативный текст (**alt**), а для видео и аудио расшифровки текста и подписи если это возможно.

Альтернативное содержимое может помочь людям с ограниченными возможностями. Например человеку со слабым зрением сложно понять, что на картинке если для нее не задан **@alt**. Другим людям может быть тяжело понять о чем говорится в видео или аудио записи.

(Если для картинки **alt** избыточен, или она используется только в декоративных целях в местах, где нельзя использовать CSS, используйте пустой альтернативный текст **alt=""**)

Не рекомендуется:

```

```

Рекомендуется:

```

```

Разделение ответственности

Разделяйте структуру, оформление и поведение.

Держите структуру (разметка), оформление (стили) и поведение (скрипты) отдельно и постарайтесь свести взаимодействие между ними к минимуму.

Убедитесь, что документы и шаблоны содержат только HTML, и что HTML служит только для задания структуры документа. Весь код, отвечающий за оформление, перенесите в файлы стилей, а код отвечающий за поведение — в скрипты.

Старайтесь сократить их пересечения к минимуму, включая в шаблоны минимальное количество файлов стилей и скриптов.

Отделение структуры от представления и поведения помогает облегчить поддержку кода. Изменение шаблонов и HTML-документов всегда занимает больше времени чем изменение файлов стилей или скриптов.

Не рекомендуется:

```
<!DOCTYPE html>
<title>HTML sucks</title>
<link rel="stylesheet" href="base.css" media="screen">
<link rel="stylesheet" href="grid.css" media="screen">
<link rel="stylesheet" href="print.css" media="print">
<h1 style="font-size: 1em;">HTML Отстой</h1>
<p>Я об этом и раньше где-то читал, но теперь точно все ясно:
    <u>HTML – полная фигня!!1</u>
<center>Не могу поверить, что для того чтобы изменить оформление,
нужно каждый раз все переделывать заново. </center>
```

Рекомендуется:

```
<!DOCTYPE html>
<title>My first CSS-only redesign</title>
<link rel="stylesheet" href="default.css">
<h1>Мой новый CSS дизайн</h1>
<p>Я читал об этом и раньше, но наконец-то сделал сам:
    Используя принцип разделения ответственности и не пихаю оформление в HTML
<p>Как круто!
```

Ссылки-мнемоники
Не используйте ссылки-мнемоники.

Нет смысла использовать ссылки-мнемоники, такие как —, ”, или ☺, когда все команды в файлах, редакторах используют одну кодировку (UTF-8)

Единственное исключение из этого правила — служебные символы HTML (например < и &) а так же вспомогательные и “невидимые” символы (например неразрывный пробел).

Не рекомендуется:

Валютный знак евро: “&eur;”.

Рекомендуется:

Валютный знак евро: “€”.

Необязательные теги
Не используйте необязательные теги. (не обязательно)

Для уменьшения размера файлов и лучшей читаемости кода можно опускать необязательные теги. В спецификации HTML5 (англ.) есть список необязательных тегов.

(Может потребоваться некоторое время для того, чтобы этот подход начал использоваться повсеместно, потому что это сильно отличается от того, чему обычно учат веб-разработчиков. С точки зрения, согласованности, и простоты кода лучше всего опускать все необязательные теги, а не некоторые из них).

Не рекомендуется:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Тратим байты – тратим деньги.</title>
  </head>
  <body>
    <p>Вот.</p>
  </body>
</html>
```

Рекомендуется:

```
<!DOCTYPE html>
<title>Байты-деньги!</title>
<p>Так-то
```

Атрибут 'type'

Не указывайте атрибут **type** при подключении стилей и скриптов в документ.

Не используйте атрибут **type** при подключении стилей (кроме вариантов когда используется что-то кроме CSS) и скриптов (кроме вариантов когда это не JavaScript).

Указывать атрибут **type** в данном случае не обязательно потому что HTML5 использует [text/css \(англ.\)](#) и [text/javascript \(англ.\)](#) по умолчанию. Это будет работать даже в старых браузерах.

Не рекомендуется:

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

Рекомендуется:

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

Не рекомендуется:

```
<script src="//www.google.com/js/gweb/analytics/autotrack.js"
type="text/javascript"></script>
```

Рекомендуется:

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

Правила форматирования HTML

Форматирование

Выделяйте новую строку для каждого блочного, табличного или списочного элемента и ставьте отступы для каждого дочернего элемента.

Независимо от стилей заданных для элемента (CSS позволяет изменить поведение элемента с помощью свойства **display**), переносите каждый блочный или табличный элемент на новую строку.

Также ставьте отступы для всех элементов вложенных в блочный или табличный элемент.

(Если у вас возникнут сложности из-за пробельных символов между списочными элементами, допускается поместить все **li** элементы в одну строку. Линту [утилита для проверки качества кода прим. пер.] рекомендуется в данном случае выдавать предупреждение вместо ошибки.

Рекомендуется:

```
<blockquote>
  <p><em>Space</em>, the final frontier.</p>
</blockquote>
```

Рекомендуется:

```
<ul>
  <li>Маша
  <li>Глаша
  <li>Чебураша
```


Рекомендуется:

```
<table>
  <thead>
    <tr>
      <th scope="col">Прибыль
      <th scope="col">Налоги
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>$ 5.00
      <td>$ 4.50
    </tr>
  </tbody>
</table>
```

Правила оформления CSS

Валидность CSS

По возможности используйте валидный CSS-код.

Кроме случаев, где необходим браузеро-зависимый код, или ошибок валидатора, используйте валидный CSS код.

Используйте такие инструменты как [W3C CSS Валидатор \(англ.\)](#) для проверки своего кода.

Валидность — это важное и при этом измеряемое качество кода. Написание валидного CSS помогает избавиться от избыточного кода и обеспечивает правильное использование таблиц стилей...

Идентификаторы и названия классов

Используйте шаблонные или имеющие смысл имена классов и идентификаторы.

Вместо использования шифров, или описания внешнего вида элемента, попробуйте в имени класса или идентификатора выразить смысл его создания, или дайте ему шаблонное имя...

рекомендуется выбирать имена, отражающие сущность класса, потому что их проще понять и, скорее всего, не понадобится менять в будущем.

Шаблонные имена — это просто вариант названия для элементов, у которых нет специального предназначения или которые не отличаются от своих братьев и сестер. Обычно они необходимы в качестве “Помощников.”

Использование функциональных или шаблонных имен уменьшает необходимость ненужных изменений в документа или шаблонах.

Не рекомендуется:

```
/* Не рекомендуется: не имеет смысла */
#yee-1901 {}

/* Не рекомендуется: описание внешнего вида */
.button-green {}
.clear {}
```

Рекомендуется:

```
/* Рекомендуется: точно и по делу */
#gallery {}
#login {}
.video {}

/* Рекомендуется: шаблонное имя */
.aux {}
.alt {}
```


Названия идентификаторов и классов

Для идентификаторов и классов используйте настолько длинные имена, насколько нужно, но настолько короткие, насколько возможно.

Попробуйте сформулировать, что именно должен делать данный элемент, при этом будьте кратки насколько возможно.

Такое использование классов и идентификаторов вносит свой вклад в облегчение понимания и увеличение эффективности кода.

Не рекомендуется:

```
/* Не рекомендуется */  
#navigation {}  
.atr {}
```

Рекомендуется:

```
/* Рекомендуется */  
#nav {}  
.author {}
```

Селекторы типа

Избегайте использование имен классов или идентификаторов с селекторами типа (тега) элемента.

Кроме случаев когда это не обходимо (например с классами-помощниками), не используйте названия элементов с именами классов или идентификаторами.

Это повышает [Производительность \(англ.\)](#).

Не рекомендуется:

```
/* Не рекомендуется */  
ul#example {}  
div.error {}
```

Рекомендуется:

```
/* Рекомендуется */  
#example {}  
.error {}
```

Сокращенные формы записи свойств

Используйте сокращенные формы записи свойств, где возможно.

CSS предлагает множество различных [сокращенных \(англ.\)](#) форм записи (например **font**), которые рекомендуется использовать везде где это возможно, даже если задается только одно из значений.

Использование сокращенной записи свойств полезно для большей эффективности и лучшего понимания кода.

Не рекомендуется:

```
/* Не рекомендуется */  
border-top-style: none;  
font-family: palatino, georgia, serif;  
font-size: 100%;  
line-height: 1.6;  
padding-bottom: 2em;
```



```
padding-left: 1em;
padding-right: 1em;
padding-top: 0;
```

Рекомендуется:

```
/* Рекомендуется */
border-top: 0;
font: 100%/1.6 palatino, georgia, serif;
padding: 0 1em 2em;
```

0 и единицы измерения

Не указывайте единицы измерения для нулевых значений

Не указывайте единицы измерения для нулевых значений если на это нет причины.

Рекомендуется:

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

0 в целой части дроби

Не ставьте “0” в целой части дробных чисел.

Не ставьте **0** в целой части в значениях между -1 и 1.

Рекомендуется:

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

Кавычки в ссылках

Не используйте кавычки в ссылках

Не используйте кавычки ("" , ") с **url()**.

Рекомендуется:

```
@import url(//www.google.com/css/go.css);
```

Шестнадцатеричные названия цветов

Используйте трехсимвольную шестнадцатеричную запись где это возможно.

Трехсимвольная шестнадцатеричная запись для цветов короче и занимает меньше места.

Не рекомендуется:

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

Рекомендуется:

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

Префиксы

Предваряйте селекторы уникальными для текущего приложения префиксами. (не обязательно)

В больших проектах, а так же в коде, который будет использоваться для других проектов или в других сайтах, используйте префиксы (в качестве пространств имен) для идентификаторов и имен классов. Используйте короткие уникальные названия с последующим дефисом.

Использование пространств имен позволяет предотвратить конфликты имен и может облегчить обслуживание сайта. Например при поиске и замене.

Рекомендуется:

```
.adw-help {} /* AdWords */
#maia-note {} /* Maia */
```

Разделители в классах и идентификаторах

Разделяйте слова в идентификаторах и именах классов с помощью дефиса.

Не используйте ничего, кроме дефиса, для соединения слов и сокращений в селекторах, чтобы повысить удобство чтения и легкость понимания кода.

Не рекомендуется:

```
/* Не рекомендуется: слова “demo” и “image” не разделены */
.demoimage {}

/* Не рекомендуется: используется подчеркивание вместо дефиса */
.error_status {}
```

Рекомендуется:

```
/* Рекомендуется */
#video-id {}
.ads-sample {}
```

Хаки

Избегайте использования информации о версии браузеров, или CSS “хаков”— сперва попробуйте другие способы.

Кажется заманчивым бороться с различиями в работе разных браузеров с помощью CSS-фильтров, хаков или прочих обходных путей. Все эти подходы могут быть рассмотрены лишь в качестве последнего средства, если вы хотите получить эффективную и легко поддерживаемую кодовую базу. Проще говоря, допущение хаков и определения браузера повредит проекту в долгосрочной перспективе, так как это означает, что проект идет по пути наименьшего сопротивления. Что облегчает использование хаков и позволяет использовать их все чаще и чаще, что в результате приведет к слишком частому их использованию.

Правила форматирования CSS

Упорядочивание объявлений

Сортируйте объявления по алфавиту.

Задавайте объявления в алфавитном порядке, чтобы получить согласованный код, с которым легко работать.

При сортировке игнорируйте браузерные префиксы. При этом, если для одного свойства используются несколько браузерных префиксов, они также должны быть отсортированы (например **-moz** должен быть перед **--webkit**)

Рекомендуется:

```
background: fuchsia;
border: 1px solid;
-moz-border-radius: 4px;
-webkit-border-radius: 4px;
border-radius: 4px;
color: black;
text-align: center;
text-indent: 2em;
```

Отступы в блоках.

Всегда ставьте отступы для содержимого блоков.

Всегда ставьте отступы для любого [блочного содержимого \(англ.\)](#), Например для правил внутри правил или объявлений, чтобы отобразить иерархию и облегчить понимание кода.

Рекомендуется:

```
@media screen, projection {

    html {
        background: #fff;
        color: #444;
    }

}
```

После объявлений

Ставьте точку с запятой после каждого объявления.

После каждого объявления ставьте точку с запятой для согласованности кода и облегчения добавления новых свойств.

Не рекомендуется:

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

Рекомендуется:

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

После названий свойств

Используйте пробелы после двоеточий в объявлениях.

Всегда используйте один пробел после двоеточия (но не до) в объявлениях, для порядка в коде.

Не рекомендуется:

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

Рекомендуется:

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

Отделение селектора и объявления

Отделяйте селекторы и объявления переносом строки.

Начинайте каждый селектор или объявление с новой строки.

Не рекомендуется:

```
a:focus, a:active {  
  position: relative; top: 1px;  
}
```

Рекомендуется:

```
h1,  
h2,  
h3 {  
  font-weight: normal;  
  line-height: 1.2;  
}
```

Разделение правил

Разделяйте правила переносом строки.

Всегда ставьте перенос строки между правилами.

Рекомендуется:

```
html {  
  background: #fff;  
}  
  
body {  
  margin: auto;  
  width: 50%;  
}
```

Мета правила CSS

Группировка правил

Группируйте правила и обозначайте группы комментарием. (не обязательно)

По возможности объединяйте правила в группы. Обозначайте группы комментариями и разделяйте переносом строки.

Рекомендуется:

```
/* Header */  
  
#adw-header {}  
  
/* Footer */  
  
#adw-footer {}
```

```
/* Gallery */

.adw-gallery {}
```

Заключение

Будьте последовательны

Если вы редактируете код, потратьте несколько минут, чтобы разобраться в том, как он написан. Если математические операторы обособлены пробелами, делайте то же самое. Если комментарии окружены скобочками или черточками, сделайте то же со своими комментариями.

Идея этого руководства в том, чтобы создать общий словарь, который позволил бы разработчикам сконцентрироваться на том **что** они хотят выразить, а не на том, **как**.

Мы предлагаем единые правила оформления позволяющие писать код в одном стиле, но стиль кода, уже используемый в проекте, также важен.

Если ваш код будет сильно отличаться от существующего, это может сбить читающего с ритма и затруднить чтение. Постарайтесь этого избежать.

Примечание от переводчика

Хочется еще отметить, что Google ориентируется в первую очередь на большие высоконагруженные проекты, где каждый байт дорог, поэтому стоит учитывать, что если они рекомендуют начинать каждый селектор с новой строки, или использовать пробелы вместо табов, то это в первую очередь подразумевает, что код будет обязательно минифицирован и сжат до использования на сайте.

Спасибо всем кто дочитал до этого места.

 css, html, styleguides

↑

+277





↓

👁

150k


★

3153



↔

Перевод: **Jens O. Meiert** / Google



Kirill Cherkashin

@z6Dabrata

карма

рейтинг

89,5

0,0

Реклама

Похожие публикации

+37

Подборка интересных CSS рецептов «Голые пятницы #3»

👁 43,4k

★ 674

💬 14

+4

Разработка CSS в GitHub

👁 8,5k

★ 82

💬 1

+42


Подборка занимательных CSS рецептов «Голые пятницы #2»

👁 45,2k

★ 650

💬 39

Комментарии (168)



armed

8 мая 2012 в 09:03

#

+10

↑

↓

Полезно, спасибо.

Дочитать не получилось, но такой статье самое место в избранном, однозначно.

 **h0use** 8 мая 2012 в 09:07 #

+16 ↑ ↓

Интересная статья, но вот с пунктом «Атрибут 'type'» не совсем согласен, наш сайт весьма глючил в старых браузерах, когда у

 **Amadeusck** 8 мая 2012 в 14:25 # h ↑

+3 ↑ ↓

Интересно у вас комментарий оборвался.

 **h0use** 8 мая 2012 в 14:27 # h ↑


+2 ↑ ↓

Что-то сглючило видать :) я имел ввиду, что браузеры не полдключали скрипты если был не указан тип, кажись IE6-7 этим баловались, точно не помню, давно было.

 **yurik417** 16 мая 2012 в 10:26 # h ↑

0 ↑ ↓

Внезапно стук в дверь

 **putnik** 8 мая 2012 в 09:26 #

+22 ↑ ↓

Важно понимать, что это именно советы от Гугла, т. е. они заточены под то, что а) над кодом может работать куча людей, б) дорог каждый байт.

Именно поэтому появляются советы делать вот так (omg!):

```
<!DOCTYPE html>
<title>Байты-деньги!</title>
<p>Так-то
```

или сортировать правила по алфавиту, а не группировать по смыслу (ибо смысл субъективен, а алфавит у всех один).

Хотя большая часть советов универсальна, бездумно их применять всё-таки не стоит.

 **z6Dabrata** 8 мая 2012 в 09:44 # h ↑

+1 ↑ ↓

Вот это и интересно в данном случае.
Мне кажется, что дело не в попытке сократить место, потому что сейчас любой код перед использованием проходит пре-обработку, во время которой можно легко вырезать все необязательные теги.

Возможно просто попытка сэкономить время разработчиков?

 **IonDen** 8 мая 2012 в 10:00 # h ↑

+3 ↑ ↓

Не встречал пока что преобработки html-кода. Максимум что делают — сливают все css, js в один файл, убирают все пробелы. У html-кода тоже убирают все пробелы. Но вырезать теги, нет, я бы не позволил, можно и все порушить.

В данном случае это действительно делается для маниакальной задачи уменьшения веса страниц. С точки зрения разработки такой код читать гораздо сложнее.

Ну а уж время разработчиков это вовсе не экономит. Если вы ведете разработку в профессиональном IDE, то там для вас будут все инструменты для быстрой вставки кода, начиная с автокомплита и заканчивая сниппетами. Либо вообще многие могут использовать zen-coding для сверх быстрого набора кода.

 **z6Dabrata** 8 мая 2012 в 10:16 # h ↑

0 ↑ ↓

Вот хорошая статья Kangax'a на эту тему, к сожалению сайт лежит, так что пока версия из кеша:
webcache.googleusercontent.com/search?q=cache:http://perfectionkills.com/experimenting-with-html-minifier/
Штука пре-обращивает html включая вырезание необязательных тегов.

Думаю, что сейчас это не очень популярно, потому что неплохо загружает сервер, но в ближайшем будущем думаю ситуация поменяется.


Вот еще пара линков на программки, слышал о них, но не работал:
code.google.com/p/htmlcompressor/#How_it_Works
code.google.com/p/page-speed/wiki/MinifyHtml

Насчет автокомплита и zen-coding — абсолютно согласен, но быстрее — не написать тег вообще, чем на писать его быстро.

 **IonDen** 8 мая 2012 в 10:29 # h ↑

+2 ↑ ↓

В том то и дело, что обычно автокомплит ставит закрывающие теги автоматически, так же как и zen-coding. Так что разницы по времени нет.

 **z6Dabrata** 8 мая 2012 в 10:39 # h ↑ 0 ↑ ↓

Я имел в виду head, body, tbody и т.д. хотя они тоже наверное есть в шаблонах и проставляются атоматически.

 **Source** 9 мая 2012 в 01:10 # h ↑ +2 ↑ ↓

Для тех, кто не любит писать лишние символы, есть [Hamli](#), на нём и пишется быстрее и читается он гораздо лучше, чем HTML без закрывающих тегов.

 **reiser** 10 мая 2012 в 13:45 # h ↑ 0 ↑ ↓

А для тех, кто ещё больше не любит лишние символы, есть [Slim](#) :)
Хотя у них обоих проблем хватает

 **Source** 16 мая 2012 в 19:41 # h ↑ 0 ↑ ↓

и в чём же у них проблемы для верстальщика? если заказчик требует HTML, то сгенерировать их из Hamli-файлов дело пары секунд.

 **reiser** 16 мая 2012 в 20:52 # h ↑ 0 ↑ ↓

Вот тут обсуждали: juick.com/Mendor/1876034#7
В кратце: inline-оформление и javascript

 **Source** 16 мая 2012 в 21:07 # h ↑ 0 ↑ ↓

ну это скорее проблема в верстальщике, если он inline-стили и inline-javascript в разметку суёт

 **vsb** 8 мая 2012 в 12:27 # h ↑ +2 ↑ ↓

Мне проще читать, когда меньше текста и избыточной информации. Хотя при этом требуется держать контекст в голове, ну и если много кода, можно и запутаться. Хотя это субъективно, кто к чему привык.

НЛО прилетело и опубликовало эту надпись здесь

 **neuotq** 8 мая 2012 в 10:22 # h ↑ +2 ↑ ↓

А почему бы и нет, мне идея понравилась. я даже как то не задумывался об этом и по правде говоря не знал о необязательных тегах в html5.

 **RAMAZAN** 8 мая 2012 в 09:37 # +1 ↑ ↓

Спасибо за перевод! Прочитал с удовольствием.

 **morello** 8 мая 2012 в 09:45 # 0 ↑ ↓


Однозначно — полезное. И в одном месте. В избранное без раздумий. Спасибо

 **miguelle74** 8 мая 2012 в 10:08 # +1 ↑ ↓

Хороший перевод нужной статьи. Бездумно применять, как писалось выше — не рекомендуется, но в целом очень полезно!

 **SergeiStartsev** 8 мая 2012 в 10:15 # +2 ↑ ↓

Этакий Code Convention, его действительно не хватало по html и css, с еще большим интересом ознакомился бы с аналогичным по js.

 **z6Dabrata** 8 мая 2012 в 10:22 # h ↑ +4 ↑ ↓

Есть и для Javascript, не переведенный правда:
google-styleguide.googlecode.com/svn/trunk/javascriptguide.xml

Вообще гугл все свои руководства по оформлению складывает в одном месте:
code.google.com/p/google-styleguide/

 **SergeiStartsev** 8 мая 2012 в 10:30 # h ↑ +1 ↑ ↓

О, спасибо!

Необязательные теги

Не используйте необязательные теги. (не обязательно)

Взорвало мозг сначала :)

**yktoo** 8 мая 2012 в 13:47 <#> [h](#) [↑](#)

+11 [↑](#) [↓](#)

Не обязательно не использовать необязательные теги, что непонятного-то :-)

**Richard_Ferlow** 8 мая 2012 в 10:37 <#>

+1 [↑](#) [↓](#)

А можно разве так?

Пять

Погулять

т.е. не закрывая тэги? еще в примерах там есть с td и th

**z6Dabrata** 8 мая 2012 в 11:03 <#> [h](#) [↑](#)

0 [↑](#) [↓](#)

Можно, см. пункт «Необязательные теги»

**deerua** 8 мая 2012 в 10:43 <#>

+24 [↑](#) [↓](#)

Рекомендует гугл у которого упор на скорость сайтов, а по их правилам пробелы будут весить больше правил

Рекомендует гугл у которого вместо инпутов/текстзон стоят дивы с контентедит

Рекомендует гугл у которого ни одного валидного сайта

Рекомендует гугл у которого куча сайта сверстанных на тейблах

зы: правила ок, только многие это просто ИМХО, мне удобнее табы, мне нравится писать тип цсс и джс и т.д.

**AlexoLive** 8 мая 2012 в 11:13 <#> [h](#) [↑](#)

+22 [↑](#) [↓](#)

Тот самый гугл, который зарабатывает больше тех у кого сайты валидны :)

**Perez** 8 мая 2012 в 14:03 <#> [h](#) [↑](#)

+4 [↑](#) [↓](#)

зелёное с мягким не надо путать. Что можно льву, то смерть оленю. :)

**MaGic2laNTern** 8 мая 2012 в 16:03 <#> [h](#) [↑](#)

+6 [↑](#) [↓](#)

Ну как-то так, да.

Меня ещё удивило, что в примере ниже (который рекомендуется как валидный HTML) есть пробелы в конце строк, хотя чуть выше рекомендуется так не делать.

```
<!DOCTYPE html>
<meta charset="utf-8">
<title>Проверка</title>
<article>Просто проверка.</article>
```

Но, впрочем, я проверил в оригинале — там пробелов не было.

Но главное не в этом, а в том, что действительно type=«text/css» и type=«text/javascript» выглядят, как мне кажется, неплохо, и убирать их особого смысла нет (или, например, убирать type из).

Да и опускать тэги html, head и body — тоже как-то не очень приятно. Можно, но я бы не сказал, что эта экономия нескольких байт так уж хороша. Я помню, некоторое время назад любители CP1251 и KOI8-R «козыряли» тем, что кириллические символы на их страницах занимают вдвое меньше места, и это очень ценно.

Помню недавно работал с верстальщиками, и написал им длинные guidelines, где довольно чётко было прописано, как именно требуется сделать. Они не выполнили, по-моему, и половину оттуда. Попытались сначала вообще невалидно сверстать (ссылаясь на какую-то статью на Хабре), потом сверстали валидно, но почему-то с XHTML 1.0 (26 января 2000), и не указав alt (хотя бы "") ни у одной картинки — поэтому получилась куча ошибок валидации. Потом я им указал на это, и они поставили alt (но абсолютно везде alt="", даже там, где имеет смысл написать что-то конкретное). Ещё у них там были прямо в HTML указаны JS-события для инпутов, для того, чтобы туда подставлялись placeholder'ы (хотя, казалось бы, можно просто добавить атрибут placeholder, а для старых браузеров подключить [jquery.placeholder](#)). Когда я им об этом написал, и указал, как лучше сделать, они поставили там нужные атрибуты, но так и не поменяли doctype на HTML5 (то есть снова ошибки валидации).

Ещё у них там зачем-то было такое:

```
<meta http-equiv="X-UA-Compatible" content="e;IE=EmulateIE7" />
<meta http-equiv="X-UA-Compatible" content="e;IE=7" />
```

Наверно, это для того, чтобы если более новые и адекватные версии IE уже могли нормально поддерживать какие-либо современные тэги, атрибуты и свойства, то они этого не делали.

В общем, многие люди делают много очень странных вещей. Аналогично, я не понимаю, когда люди указывают у каких-либо элементов (например, у текстовых полей) только цвет фона или только цвет текста. Потому что на практике это означает, что у любого пользователя, у которого цвет фона по умолчанию не белый и цвет текста по умолчанию не чёрный (а для этого достаточно сменить тему) будет наблюдать, например, очень тёмный текст на очень тёмном фоне (автор-то думал, что цвет фона у всех будет белый, у него-то белый отображается). Или если, например, задан светлый цвет текста, а сзади загружается тёмная картинка. Но тёмный цвет фона не задан. Поэтому если картинка по какой-то причине не загрузится, то текст будет невозможно прочитать.

Ещё очень часто у картинок не заданы width и height, и это приводит к тому, что содержимое вокруг картинки «скачет» в разные стороны, пока картинки загружаются.

Но лидирует в направлении «Самая странная вёрстка» пока что всё же белорусский сайт beltechnolift.com (на Хабре про него было). Вот это я понимаю — тут не то, что ни о каких гайдлайнах речи не идёт, а тут у авторов вообще совершенно альтернативное понимание всего. :)

В общем, это всё к чему. Советы от Google неплохи (да и, что уж там — эта компания как правило понимает, о чём говорит), но ни в коем случае не нужно руководствоваться их советами в том случае, если это противоречит здравому смыслу. При разработке разных сайтов задачи совершенно разные, и в каких-то случаях бывает так, что об оптимизации размера содержимого, выдаваемого пользователям, думают в последнюю очередь (бывает так, что оно должно быть наоборот как можно более читаемое для людей, и так далее).

Поэтому тут надо делать не по принципу «сделаю так, потому что знающий человек так сказал», а по принципу «сделаю так, потому что есть такие-то причины для того, чтобы это решение работало лучше, чем все остальные возможные».

 **MaGIc2laNTern** 8 мая 2012 в 16:12 # h ↑ +1 ↑ ↓

Вернее так:


Но главное не в этом, а в том, что действительно type="text/css" и type="text/javascript" выглядят, как мне кажется, неплохо, и убирать их особого смысла нет (или, например, убирать type из <link rel="icon" type="image/png" href="/favicon.png">).

Странно, кстати, что на Хабре нельзя отключить автоматическую замену «"» на кавычки-ёлочки. Потому что если я имею в виду кавычки-ёлочки, то я нажимаю на Compose key и ввожу два раза << или >>. А если ввожу «"», то и имею в виду «"». А так приходится писать ".

Да, кстати, вспомнилось — ещё часто забывают «&» в ссылках заменить на &. Браузер эту ошибку умеет обрабатывать, но валидность от этого уже теряется. Так что лучше заменять, чтобы было по стандартам.

 **z6Dabrata** 8 мая 2012 в 17:38 # h ↑ +1 ↑ ↓

Прошу прощения за пробелы в примерах — боролся с парсером.

 **MaGIc2laNTern** 8 мая 2012 в 18:56 # h ↑ +1 ↑ ↓

Понятно. Ну, я примерно так и понял. :)

Жалко, что у Хабра нету режима, в котором он вообще не меняет вводимый текст (ну, или только убирает из него потенциально опасные для пользователей конструкции).

 **AYShestakov** 11 мая 2012 в 11:57 # h ↑ 0 ↑ ↓

```
<!doctype html>
<!--[if lt IE 7]>
<html class="no-js lt-ie9 lt-ie8 lt-ie7" lang="en"> <![endif]-->
<!--[if IE 7]>
<html class="no-js lt-ie9 lt-ie8" lang="en"> <![endif]-->
<!--[if IE 8]>
<html class="no-js lt-ie9" lang="en"> <![endif]-->
<!--[if gt IE 8]><!-->
<html class="no-js" lang="en"> <!--<![endif]-->
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
```

Пример из [Boilerplate](#)



NSA 8 мая 2012 в 17:43 # h ↑

+4 ↑ ↓

Рекомендует гугл у которого вместо инпутов/текстзон стоят дивы с контентедит

В инпутах и textarea невозможно форматировать вводимый текст.



potomushto 8 мая 2012 в 10:52 #

+4 ↑ ↓

Гитхабовские соглашения по написанию кода для [javascript](#) и [css](#)



Alexufo 8 мая 2012 в 11:35 # h ↑

+1 ↑ ↓

ну раз пошла такая пьянка - [codex.wordpress / CSS_Coding_Standards](#)



potomushto 8 мая 2012 в 12:00 # h ↑

+1 ↑ ↓

Добавлю по Twitter. Собственно их Bootstrap сам по себе несёт стандарты, косвенно в коде и напрямую в документе [js philosophy](#), а для поддержания css-кода есть ещё неплохой проект [Twitter Recess](#)



ShpuntiK 8 мая 2012 в 11:21 #

+23 ↑ ↓

А я всё равно tab люблю.



fatum 8 мая 2012 в 11:33 #

+7 ↑ ↓

Меня заботит одно — никто в таких сборниках правил не пишет почему так нужно делать, почему пришли к тому или иному решению. Только несколько пунктов имеют причину.



Shedal 8 мая 2012 в 11:48 # h ↑

+1 ↑ ↓

На мой взгляд, большинство правил обоснованы одной из двух причин: либо стремлением к осмысленной краткости, где только возможно, либо к стилю, который будет привычен для большинства разработчиков.



monolithed 8 мая 2012 в 11:38 #

+2 ↑ ↓

Относительно мнемоник не совсем корректный пример, т.к. знак евро отсутствует в стандартной раскладке, а кавычки и так все пишут без мнемоник.



TNK 8 мая 2012 в 14:55 # h ↑

+2 ↑ ↓

В примере не знаки дюйма, а именно ковычки, которые тоже отсутствуют в стандартной раскладке.



Shedal 8 мая 2012 в 11:46 #

+3 ↑ ↓

Просто какой-то поток незамутнённого здравого смысла :) Отличные правила!

Единственное правило, с которым я не согласен — сортировка правил css по алфавиту. Мне кажется, это ничего не даёт с точки зрения поддерживаемости и читабельности.

Я, например, сортирую правила по степени важности, причём, в начале идут правила расположения (position, float, top, и т.д.), а потом уже правила отображения (font, color, background-image, и т.п.).



jakobz 8 мая 2012 в 12:21 # h ↑

+1 ↑ ↓

Сортировка упрощает жизнь разработчикам. Вот в твой такой код залезешь и будешь думать как там расставить правила, чтобы они по важности были, чтобы ты потом не ругался. Можно, конечно, и такой порядок в документ описать, но получится слишком сложно, не все запомнят, и соблюдаться такое вряд ли будет.



Shedal 8 мая 2012 в 12:28 # h ↑

+2 ↑ ↓

Сортировка упрощает жизнь только если какой-то стиль содержит несколько десятков строк — тогда в них проще ориентироваться, если они отсортированы. Но с другой стороны, если в стиле несколько десятков строк, то это явный признак того, что его можно отрефакорить — разделить правила на несколько семантически осмысленных селекторов.

И, по-моему, правило «*сначала структурно-позиционные правила, а затем влияющие на внутреннее отображение*» не сильно сложнее правила «*сортировать по алфавиту, но вставлять правила с префиксами после одноимённых безпрефиксных правил*».




LXE 2 августа 2012 в 00:11 # h ↑

0 ↑ ↓

Сортировка (в любом детерминированном порядке) — это необходимое условие осмысленного функционирования differencing/merging tools.

Это как в Java-проектах правила сортировки-группировки импортов. Или они есть и общие для команды, или кому-то придется вручную вправлять кишки при каждом мерже — что портит жизнь при стратегии merge often не хуже, чем частые светофоры на хайвее.

 **dominik** 8 мая 2012 в 11:57 #

+1 ↑ ↓

С некоторыми пунктами можно поспорить, но в целом полезно. Хотя и местами по-кэпски.

Насчет валидности и CSS-хаков. Я двумя руками за валидность кода настолько, насколько это возможно. Но бессмысленно превращать её в самоцель. Я через это проходил. Но сейчас уже пришел к выводу, что при разумном использовании хаки зачастую более удобный и надежный инструмент, чем фанатичная борьба за абсолютную валидность.

 **sdevalex** 8 мая 2012 в 12:28 #

+1 ↑ ↓

А интересно, гугл использует вещи типа HAML, SASS, CoffeeScript? Используя это сложнее нарушать проловина правил из списка.

 **sarjick** 8 мая 2012 в 13:06 #

+1 ↑ ↓

Прочитал статью, глянул на главную страницу google.com.ua и сразу в глаза по первой рекомендации нарушение — `<script type="https://..."`

не говоря уже о таких записях `<body bgcolor="#fff"`

 **lukaville** 8 мая 2012 в 13:38 # h ↑

+1 ↑ ↓

И кстати, `<!doctype html>` (как на главной гугла и по правилу «писать всё в нижнем регистре» или `<!DOCTYPE html>` (как в примерах в статье)?

 **MTonly** 8 мая 2012 в 15:47 # h ↑

+2 ↑ ↓

В HTML регистр `DOCTYPE` не имеет значения, но в XML-нотации ключевое слово пишется в верхнем регистре, поэтому и в HTML-нотации имеет смысл делать так же.

НЛО прилетело и опубликовало эту надпись здесь

 **MTonly** 8 мая 2012 в 17:02 # h ↑

+1 ↑ ↓

К вопросу регистра это отношения не имеет. ;-)

НЛО прилетело и опубликовало эту надпись здесь

 **reiser** 8 мая 2012 в 13:07 #


+8 ↑ ↓

Откуда у всех такая любовь к отступам из пробелов? Чем народу табы не угодили?

 **limon_spb** 8 мая 2012 в 13:26 # h ↑

+1 ↑ ↓

Согласен! и с параноидальной точки зрения экономии трафика табы эффективнее. Не понятно.

 **vsb** 8 мая 2012 в 13:34 # h ↑

+3 ↑ ↓

С параноидальной точки зрения экономии трафика эффективнее минифицировать всё специальными утилитами. Не будет ни пробелов ни табов, все довольны :)

 **mr_avi** 8 мая 2012 в 16:09 # h ↑

+2 ↑ ↓

Если мой, например, нетбинс настроен на `tab = 4` «отступа», а чей-то еще нетбинс настроен на `tab=2` «отступа», и у меня в коде намешаны табы и пробелы, то у второго весь код полезет.

 **reiser** 8 мая 2012 в 16:31 # h ↑

+5 ↑ ↓

я и не предлагаю мешать табы и пробелы, мне непонятна эта тенденция с использованием двух пробелов вместо табов

 **E_STRICT** 8 мая 2012 в 19:58 # h ↑

+2 ↑ ↓

Длина одного таба зависит от текстового редактора в котором просматривается данный файл. Пробелы это более универсальное решение.

 **reiser** 8 мая 2012 в 21:07 # h ↑

+1 ↑ ↓

да, зависит, в этом и смысл, что каждый может настроить отображение отступов так, как ему нравится



Athari 8 мая 2012 в 21:51 # h ↑

+3 ↑ ↓

Это в теории, которая очень редко соотносится с реальностью. Реальность — это код вроде этого:

```
class Foo
{
    void Bar (int a,
              int b, int c)
    {
        int x = 1,
            y = 2;
    }
}
```

Единственный способ сделать отображение такого кода корректным — это очень аккуратно смешать табы и пробелы. Это совершенно непрактично, никто этим заниматься не будет. Отказываться от переносов строк — это тоже не вариант.



Shedal 8 мая 2012 в 22:20 # h ↑

+4 ↑ ↓

Да, это точно. Я бы даже сформулировал так:

При использовании пробелов табы не нужны. А вот при использовании табов иногда приходится примешивать пробелы, что плохо.



putnik 9 мая 2012 в 03:33 # h ↑

0 ↑ ↓

Какие у вас интересные конструкции в CSS :)

По поводу обычного кода понятно, что ничего не понятно. На Хабре одно время был холивар из пяти, что ли, топиков. Но тут же речь о CSS, в котором только один (изредка — два) уровень вложенности. Почему бы не пользоваться табами?



Athari 9 мая 2012 в 04:13 # h ↑

+1 ↑ ↓

В рекомендациях речь идёт и про HTML, и про CSS. В HTML случается, что у элемента атрибутов или много, или они длинные, поэтому переносить придётся. Или вы предлагаете использовать в проекте 4 и больше языка, а CSS объявить особенным и форматировать только его на табах? :)

И вообще, в современном CSS без переносов строк тоже не обойтись...

background:

```
radial-gradient(60% 43%, closest-side circle, #b03 26%, rgba(187,0,51,0) 27%),
radial-gradient(40% 43%, closest-side circle, #b03 26%, rgba(187,0,51,0) 27%),
radial-gradient(40% 22%, closest-side circle, #d35 45%, rgba(221,51,85,0) 46%),
radial-gradient(60% 22%, closest-side circle, #d35 45%, rgba(221,51,85,0) 46%),
radial-gradient(50% 35%, closest-side circle, #d35 30%, rgba(221,51,85,0) 31%),

radial-gradient(60% 43%, closest-side circle, #b03 26%, rgba(187,0,51,0) 27%) 50px 50px,
radial-gradient(40% 43%, closest-side circle, #b03 26%, rgba(187,0,51,0) 27%) 50px 50px,
radial-gradient(40% 22%, closest-side circle, #d35 45%, rgba(221,51,85,0) 46%) 50px 50px,
radial-gradient(60% 22%, closest-side circle, #d35 45%, rgba(221,51,85,0) 46%) 50px 50px,
radial-gradient(50% 35%, closest-side circle, #d35 30%, rgba(221,51,85,0) 31%) 50px 50px;
background-color:#b03;
background-size:100px 100px;
```

Источник: lea.verou.me/css3patterns/#hearts (там ещё много таких градиентов в стиле «как такое вообще возможно»)



ScratchBoom 8 мая 2012 в 13:19 #

+1 ↑ ↓

Всегда интересовало, почему атрибут alt рекомендуют проставлять даже когда он пустой?



SKeaPer 8 мая 2012 в 13:46 # h ↑

+1 ↑ ↓

Валидатор указывает ошибку/предупреждение когда у картинки вообще отсутствует alt. Вполне может быть потому.

НЛО прилетело и опубликовало эту надпись здесь



chaliy 8 мая 2012 в 14:29 # h ↑

+3 ↑ ↓

Скринридеры если alt нема, будет читать урлу картинки. Мало кому это понравится.



DaleMartinWatson 8 мая 2012 в 14:03 #

+1 ↑ ↓

А если я привык делать отступ в 4 пробела? Это очень плохо?



Kicker 8 мая 2012 в 14:18 # h ↑

+4 ↑ ↓

Да, в гугл не возьмут...



mekegi 11 мая 2012 в 12:56 # h ↑

0 ↑ ↓

Скорее всего это правило ввели изза обилия вложенных элементов в html. И читабельность кода где нибудь на 15 уровне вложенности где код начинается с позиции 50 очень низка — вот они решили сократить вместо привычных 4 до 2. Таким образом код не так быстро уезжает от левого края



maxpain 8 мая 2012 в 14:43 #

+13 ↑ ↓

Сейчас меня, конечно, забросают помидорами, но эта статья — куча противоречивых и непоследовательных, а местами просто вредных советов. То он предлагает экономить один байтик в HTML, то вдруг вспоминает про читаемость и советует в CSS после двоеточия ставить пробел. То W3C валидатор используйте, то незакрытый тег center вне body. Если для проекта актуально экономить на байтах, можно наладить автоматическую обфускацию/минификацию кода перед деплойментом. Потому что вот это:

```
<!DOCTYPE html>
<title>Байты-деньги!</title>
<p>Так-то
```

Просто безобразие.

НЛО прилетело и опубликовало эту надпись здесь



Shedal 8 мая 2012 в 16:44 # h ↑

+1 ↑ ↓

Иногда их полезно не закрывать. Например, для решения [проблемы лишних отступов между inline-элементами](#) (см. заголовок «Skip the closing tag»). На Хабре по этому поводу тоже была большая статья, но сейчас не могу найти.

В PHP, например, тоже считается хорошим тоном не писать закрывающий тег ">".

Вообще говоря, html — не xml. Заккрытие тегов во многих случаях является просто делом привычки.



AndrewStephanoff 8 мая 2012 в 20:25 # h ↑

+3 ↑ ↓

В PHP закрывающий тег не рекомендуют по причине того, что любой пробел после него попадает в output и если вы не заворачиваете output в буфер, то у вас потенциально может быть ситуация, когда контент — пробелы — уже отправился, а заголовки — ещё нет. Как это соотносится с HTML — непонятно.



Shedal 8 мая 2012 в 21:50 # h ↑

+1 ↑ ↓

Мало как соотносится, это был пример «по аналогии». Хотя по ссылке о лишних отступах между li, которую я привёл чуть выше, наблюдается точно та же проблема — whitespace'ы между несколькими элементами li попадают в документ. Если не писать закрывающий тег, этого не происходит. Так что аналогия не такая уж и далёкая.

НЛО прилетело и опубликовало эту надпись здесь



Shedal 8 мая 2012 в 22:13 # h ↑

+1 ↑ ↓

Да, я имел в виду «падают во **внешний** документ» — во внешний по отношению к inline-элементу.



maxpain 8 мая 2012 в 21:36 # h ↑

+1 ↑ ↓

>> Иногда их полезно не закрывать.



maxpain 8 мая 2012 в 21:42 # h ↑

+1 ↑ ↓

Прошу прощения, случайно отправилось недописанное.

>> Иногда их полезно не закрывать.

Таких случаев нет. Незакрытые теги порождают неопределённое поведение и ломают работу в IDE. Проблема с inline-block лечится нулевым размером шрифта либо комментированием строки.

>> Вообще говоря, html — не xml. Заккрытие тегов во многих случаях является просто делом привычки.

А в JS можно не ставить точку с запятой в конце строки. Ничего хорошего в подобной практике нет.



Athari 8 мая 2012 в 21:55 # h ↑

+2 ↑ ↓

Незакрытые теги порождают неопределённое поведение и ломают работу в IDE.

Незакрытые теги полностью соответствуют стандартам, поведение определено чётко. Они работают во всех браузерах. Если какая-то среда разработки некорректно с ними работает, то это баг.



maxpain

8 мая 2012 в 22:06

h ↑

+2 ↑ ↓

>> Если какая-то среда разработки некорректно с ними работает, то это баг.

Незакрытый <p> покалечит folding и автотабуляцию в NetBeans, Komodo, WebStorm/PhpStorm.

>> Незакрытые теги полностью соответствуют стандартам, поведение определено чётко.

habrahabr.ru/post/143452/#comment_4809102 — здесь человек описал, как неоднозначно это будет выглядеть в шаблонах.

НЛО прилетело и опубликовало эту надпись здесь



Spaceoditty

26 августа 2013 в 04:13 (комментарий был изменён)

h ↑

0 ↑ ↓

Стоп, Илья! Я не понял этой чехарды с незакрытыми тэгами.

В частности с <p> и .

Если я буду размещать вложенные абзацы и списки не закрывая тэги — как браузеры это отрендерят?

И пофиг что абзац в абзаце это несеманлично. Если html5 такой умный, что не разделяет элементы на блочные и инлайновые — то пусть и абзацы по умолчанию воспринимает как инлайновые.

НЛО прилетело и опубликовало эту надпись здесь



Athari

8 мая 2012 в 22:18

h ↑

+3 ↑ ↓

Это я и написал. :)

Проверил в PhpStorm 4 код с тремя tr в table. До сих пор не починили. :(А вот с p всё в порядке, сворачивание работает.

НЛО прилетело и опубликовало эту надпись здесь

НЛО прилетело и опубликовало эту надпись здесь



Shedal

8 мая 2012 в 22:07

h ↑

+2 ↑ ↓

Незакрытые теги порождают неопределённое поведение

Неправда. Поведение чётко описано в стандарте HTML. Обычно оно сводится к тому, что тег закрывается в тот момент, когда начинается другой, одноимённый, или же оканчивается родительский тег.

Между прочим, закрытие тегов не всегда помогает «определить» поведение. Например:

```
<p>Текст1 <p>Текст2</p> Текст3</p>
```

Этот пример будет воспринят парсером вот так:

```
<p>Текст1 </p><p>Текст2</p> Текст3
```

А всё потому, что, по стандарту, тег **p** не может содержать в себе блочных элементов.

ломают работу в IDE

Думаю, в этом случае проблемным местом является IDE, которая не поддерживает стандарт HTML.

Посмотрите и с другой стороны на теги, которые явно не закрыты. Следующая разметка:

```
<ul>
  <li>Текст 1
  <li>Текст 2
</ul>
```

Отрендерится так:

- Текст 1

- Текст 2

Одной точке соответствует один элемент li :) Как по мне, логично.

То же самое с параграфами. Обычный человек задумывается о том, что нужно перевести строку, только в начале параграфа, но не в конце.

Проблемы с таким подходом возникают только в одном случае: если пытаться воспринимать HTML как подмножество XML. Как на уровне кода, так и на уровне собственного понимания.



maxpain

8 мая 2012 в 22:24

h ↑

+2



>> Между прочим, закрытие тегов не всегда помогает «определить» поведение. Посмотрите и с другой стороны на теги, которые явно не закрыты.

Начнётся чехарда — <a> закрываем, <p> не закрываем, <div> закрываем, не закрываем. Оно вам надо? В вёрстке и так головной боли хватает с браузероспецифичными хаками.

>> Проблемы с таким подходом возникают только в одном случае: если пытаться воспринимать HTML как подмножество XML. Как на уровне кода, так и на уровне собственного понимания.

Если хочется визуальной простоты, можно использовать HAML — он сделает за вас всю грязную работу по преобразованию в HTML.

НЛО прилетело и опубликовало эту надпись здесь



maxpain

8 мая 2012 в 23:25

h ↑

+2



>> Начнем с того, что есть теги, которые закрывать нельзя: , <input>, <meta>, <hr>,
.

>> Так что отмахнуться от того, что «теги» (точнее, элементы) разные (по предназначению, типичной роли, модели содержимого, DOM-интерфейсу и т.п.), попросту не получится.

Назначение тега никак не должно пересекаться с синтаксисом. Точно так же. как в языках программирования у классов разное предназначение, но синтаксис создания объектов одинаковый:

```
var a = new Database();
var b = new URLRequest();
var c = new DateTime();
```

>> Это не «чехарда», это документированная специфика технологии.

В спецификации HTML ещё сказано, что можно значение атрибутов без кавычек писать в некоторых случаях. W3C предпочли не бороться с бардаком, а узаконить его.

>> И кто не может ее осилить (хотя не так уж она сложна, особенно если абстрагироваться от «тегов» и начать мыслить структурой страницы, как ее видит и показывает браузер) — надо ли лезть в эту воду? ;)

Кажется мы подошли к моменту, когда начинаются завуалированные упрёки в профнепригодности. На эту тему я говорить, пожалуй, не хочу.

НЛО прилетело и опубликовало эту надпись здесь



maxpain

9 мая 2012 в 00:17

h ↑

0



>> Виноват, приношу извинения! Действительно, получилось похоже на личный выпад, хотя, честное слово, и в мыслях не было, я имел в виду исключительно «порог вхождения» для новичков (который часто считают «неприлично низким»).

Окей, без проблем :) Что касается порога входа, так он ведь действительно низкий.

>> Но тем не менее пересекается — это факт (разница между пустыми и непустыми элементами в т.ч. синтаксическая). В ЯП тоже бывает разный синтаксис для создания разных объектов, и в этом нет никакой трагедии.

Трагедии, безусловно, нет. Однако все потребности HTML вполне покрываются стандартным XML-синтаксисом.

>> Узаконить бардак (точнее, исторически сложившуюся объективную реальность) решили производители браузеров, по очевидным прагматическим соображениям (совместимость с существующим контентом). «Бороться» с этим можно примерно с тем же успехом, как законодательно отменять иррациональность числа «пи» (как в каком-то американском штате:).

Бороться с бардаком можно и нужно. То, что поддерживается исключительно ради обратной

совместимости, объявлять в спецификациях новых версий HTML и CSS как obsolete. Публиковать не только спецификации, но и гайдлайны по грамотному использованию возможностей языка. Прислушиваться к тому, что говорят девелоперы, а не отклонять просьбу о введении background-position-x/-y, потому что мудрецы из комитета не видят теоретической необходимости. Перестать сношать вола на темы типа «color или colour?». Но от W3C не дождёшься :(

НЛО прилетело и опубликовало эту надпись здесь

НЛО прилетело и опубликовало эту надпись здесь

A

mr_avi

8 мая 2012 в 16:13

h ↑

-1 ↑ ↓

CSS-ки минифицируются перед выходом в продакшен.

👤

KidsKilla

8 мая 2012 в 23:09

h ↑

0 ↑ ↓

А на мой взгляд отсутствие хеда и боди есть гуд

1. этот вариант (мне) читается легче — содержимое хеда и боди абсолютно разные и не пересекаются, плюс они почти всегда пустые, без атрибутов
2. текста меньше и отступы слева не нужны

в итоге кол-во «кпд» информации выше

байты — деньги, но вот тут они ни при чём

НЛО прилетело и опубликовало эту надпись здесь

0000
0101
010

Aidsoid

8 мая 2012 в 15:46

#

-1 ↑ ↓

Вобщем-то гугл делает также как и я на протяжении многих лет, единственный момент — я не заморачиваюсь по поводу пробелов и табов.

👤

stardust_kid

8 мая 2012 в 15:47

#

+1 ↑ ↓

Согласен со всем, кроме

> Разделяйте слова в идентификаторах и именах классов с помощью дефиса.

Нижний слеш более читабельный, как мне кажется.

👤

NoN

14 мая 2012 в 19:06

h ↑

0 ↑ ↓

Согласен.

Если тут это хотя бы возможно, но большинство языков программирования (начиная с JS) не позволят так задавать названия.

Нижнее подчёркивание удобно и тем, что его можно использовать везде.

👤

Unhandled_Exception

8 мая 2012 в 16:35

#

+1 ↑ ↓

Посмотрел исходник gmail-вской страничке, и там сразу:

```
<!DOCTYPE html> <html lang="ru"> <head> <meta content="text/html; charset=utf-8" http-equiv="Content-Type"> <title>
Gmail</title>
```

:)

Если серьезно, то гугл — умничка, что популяризирует стандартизацию вот таких вещей для программистов. К примеру, мы пишем на C++, и для проверки стиля кода мы взяли их валидатор и немного подпилили. Работать стало очень удобно. Так что, валидацию — в массы!

👤

BReal

8 мая 2012 в 16:38

#

0 ↑ ↓

А на сколько эти методы близки к яндекс роботу?

И есть ли какой-нибудь софт, правящий табуляцию и пробелы в коде — а то больно много строк, руками править устану. Помню как-то в VS делал давным давно — но там ведь под HTML и CSS не заточено, верно?

🇩🇪

Shedal

8 мая 2012 в 16:47

h ↑

0 ↑ ↓

В Visual Studio можно настроить правила отступов и затем применить их ко всему документу (*Ctrl+K*, *Ctrl+D*). В том числе, можно форматировать и JS, и CSS, и HTML. Рекомендую.

👤

Squier

8 мая 2012 в 17:11

#

0 ↑ ↓

Теперь я понял, что личные законы нужно вычеркивать и полагаться на советы большого гиганта. Ему же индексировать, в первую очередь :D

Как бы все было бы хорошо, если бы не одно «но».

Собственно, оно заключается в том, что Корпорация Добра не слишком ли много одеяла начала на себя тянуть?

Правила для оформления и форматирования должны быть, да, но поисковому гиганту нужно заниматься своим делом, то есть поиском, а с правилами уж как-нить без них разберутся.

С таким успехом следующим шагом будут рекомендации от Гугла как правильно писать письма, чтобы они быстрее доходили и не попадали в спам, а также в какой цвет красить крыши домов и как равнять рельеф, чтобы на Мапсах изображение было более качественным.

Это их внутренние рекомендации, не?

Для уменьшения размера файлов и лучшей читаемости кода можно опускать необязательные теги.

Ага. Опустить. А потом выяснится, что в их же гугловском Webmaster Tools нужно что-то поместить именно в head, который внезапно становится обязательным. В целом, смысла опускать html, head, body не вижу.

Плюс немало HTML-редакторов клинит при незакрытых тегах. Автоматически переформатируешь таблицу с незакрытыми тегами в PhpStorm — от увиденной лесенки волосы дыбом встанут. Иногда незакрытые теги в принципе мешают корректно отформатировать код, например, как правильно?

```
<p>text
<? include(...) ?>
```

или

```
<p>text
  <? include(...) ?>
```

Зависит от того, что внутри включаемого файла — блок или инлайн. Если же тег <p> закрыть, неоднозначность пропадает.

Я, конечно, за минимализм, но он далеко не всегда уместен.

Конечно же, опускание тегов не всегда уместно. Поэтому в приведенной вами цитате и фигурирует слово «можно».

НЛО прилетело и опубликовало эту надпись здесь

Я бы еще добавил — в css удобнее сначала указывать свойства, влияющие на положение элемента (display,width,height,float,clear,margin), а затем свойства, описывающие его внешний вид. Таким образом за длинным описанием бэграунда и шрифта не потеряется какой-нибудь clear:right.

Есть мнение, что такие вещи лучше вообще по разным классам разносить, и назначать оба класса нужному элементу. Повышается вероятность повторного использования классов.

читаю как верстальщик и просто голова кипит :) как с такими согласится... просто многие вещи то не поддерживаются еще :)) это больше мечтание на будущее чем рекомендация у использованию

хотя конечно многие предметы конечно использую

PS но вот два пробела вместо табуляции :) как религия не позволяет делать

На первый взгляд, всё перечисленное прекрасно поддерживается. С чем именно проблемы?

например не законченные парные тэги

- текст

разве все браузеры такое держут?


 **sogologo** 8 мая 2012 в 20:45 # h ↑

0 ↑ ↓

ой извините :) коменты съели код

вот такая конструкция

```
<ul>
  <li>пункт
  <li>еще
</ul>
```


 **KidsKilla** 8 мая 2012 в 22:59 # h ↑

0 ↑ ↓

это даже ие 4 и хтмл 3 «поддерживали»

НЛО прилетело и опубликовало эту надпись здесь

НЛО прилетело и опубликовало эту надпись здесь

 **sogologo** 9 мая 2012 в 07:35 # h ↑

0 ↑ ↓

вот не знал :) наверное у меня другая школа :)


 **alekciy** 8 мая 2012 в 18:58 #

0 ↑ ↓

>*Опускайте кодировку для css-файлов*

Поскольку css линкуется с текущей страницей через *link* тег, то кодировка должна указывается в *charset* атрибуте.

В случае же единого проекта (в смысле, что разработчик имеет доступ и к коду и к конфигурации сервера) наиболее разумно использовать абсолютно везде utf-8 и отдавать этот тип кодировки в HTTP заголовке.

 **alekciy** 8 мая 2012 в 19:22 #

0 ↑ ↓

>*Отмечайте задачи для списка дел с помощью TODO*

Я бы рекомендовал немного другой формат. По коду должно быть сразу видно, когда был оставлен этот комментарий без необходимости подъема истории в репозитории. А формат такой:

[*примечание*] YYYY-MM-DD *никнейм_автора*: комментарий,


где *примечание* — тип комметария (TODO, FIXME, NOTE);

YYYY-MM-DD — дата с заполнителем 0;

никнейм_автора — ник автора каменты, все ники в рамках проекта на протяжении все его истории должны быть уникальны.

Пример:

```
// [TODO] 2012-05-08 alekciy: Съешь ещё этих мягких французских булок.
```

 **bizikov** 8 мая 2012 в 20:04 #

0 ↑ ↓

Интересное руководство, но думаю не стоит принимать все буквально. Некоторые вещи, можно взять на вооружение. Но избавляться от,, и т.д... не буду. Это все равно что фалангу пальца отрезать

 **bizikov** 8 мая 2012 в 20:27 # h ↑

0 ↑ ↓

извиняюсь, надо было в код взять: «избавиться от,, ,»

 **AndrewStephanoff** 8 мая 2012 в 20:28 #

0 ↑ ↓

А зачем указывать ? Я это делаю в заголовке: Content-Type: text/html; charset=UTF-8

 **AndrewStephanoff** 8 мая 2012 в 20:29 #


0 ↑ ↓

Имел ввиду meta тег с content-type

 **dasty** 8 мая 2012 в 21:10 #

0 ↑ ↓

Незакрытые теги li и tr режут глаз, но выглядят довольно интересно.


 **crx** 8 мая 2012 в 22:06 #

0 ↑ ↓

Ирония в том, что не хватило сил соблюсти свои же правила в руководстве по стилю:

<STYLEPOINT title="Section comments">


<SUMMARY>
Group sections by a section comment (optional).
</SUMMARY>
<BODY>

 **z6Dabrata** 8 мая 2012 в 22:15 # h ↑ 0 ↑ ↓

В данном случае все в порядке: руководство описывает рекомендации по HTML и CSS, но выложено в xml+xslt, для этого у них совсем другие гайдлайны (тоже есть на сайте)

 **webknjaz** 9 мая 2012 в 00:22 # +2 ↑ ↓

Не закрывать теги? Убило мозг.

 **mrjj** 9 мая 2012 в 01:22 # 0 ↑ ↓


Этот документ наводит на ряд размышлений:

1. Либо квалификация специалистов в google, как и в любой другой организации, неоднородная, либо это открытый саботаж браузеров от сторонних производителей.
2. Это не конвенции. Конвенции должны учитывать многие варианты и плясать от существующего положения вещей, тут же свод советов от чего нужно отказываться, в своем роде видение компанией будущих стандартов.
3. Незакрытые теги, как самый спорный момент, неприемлемы несколько по иной причине нежели «ой как непривычно». Многие языки разметки вполне неплохо живут без явных открытий и закрытий форматирования. НО.

Какой тег сможет автоматически закрыться каким не прописано в стандарте html и вообще нигде, движки могут понимать это очень по-разному. Требовать этого от верстальщика, не предлагая ни ему ни разработчикам браузеров никакого стандарта нельзя. Так что скорее я все-таки склоняюсь к мысли о саботаже.

Про экономию байтов. Да не смешите мои тапочки. Содержимое html кода это не то, на чем имеет смысл экономить, так как это одна из худших форм оптимизации. Если они так этого хотят от индексируемых сайтов пусть первые подадут пример, вместо того ужаса, который представляет из себя верстка их сервисов. (С переходом на новые интерфейсы еще и очень глючного ужаса)


НЛО прилетело и опубликовало эту надпись здесь

 **mrjj** 9 мая 2012 в 02:43 # h ↑ 0 ↑ ↓

Текущие стандарты палка о многих концах, особенно whatwg, например в соответствии с ним вообще нет никакого html5. Это все драфты с часто непонятным статусом реализации, включать их в рекомендации и конвенции мягко говоря преждевременно, тестирование на валидное создание дерева тоже нет.

И в целом это не очень здоровая практика, так как изменение содержимого тега может повлечь за собой изменение его статуса как закрытого или открытого, что, в свою очередь, приведет к увеличению количества ошибок.

НЛО прилетело и опубликовало эту надпись здесь

 **Shedal** 9 мая 2012 в 14:34 # h ↑ +1 ↑ ↓

изменение содержимого тега может повлечь за собой изменение его статуса как закрытого или открытого


Я выше уже [приводил пример](#), из которого видно, что изменение содержимого может форсировать «досрочное» закрытие тега вне зависимости от того, был ли он явно закрыт разработчиком:

```
<p>Текст1 <p>Текст2</p> Текст3</p>
```


Этот пример будет воспринят парсером вот так:

```
<p>Текст1 </p><p>Текст2</p> Текст3
```

А всё потому, что, по стандарту, тег **p** не может содержать в себе блочных элементов.

 **mrjj** 9 мая 2012 в 14:48 # h ↑ 0 ↑ ↓

Да, то есть преобразование в DOM идет по неочевидному правилу, в коде присутствует вложенность, в DOM линейка с последний выпавшим элементом. Мне вообще кажется что html сильно страдает от подобной двойственности. Руки чешутся делать нечто вроде Текст1 Текст2 Текст3 но это некорректно с точки зрения индексации, как и, например, таблицы div-ами :(

 **Shedal** 9 мая 2012 в 14:59 # h ↑ +2 ↑ ↓

Это правило неочевидно только в том случае, если вы воспринимаете HTML как XML. В XML'е вложенность и

сохранение изначальной структуры — это святое. В HTML же есть много нюансов.

Дело в том, что и HTML, и XML были основаны на SGML, отсюда и теговый синтакс. Но SGML не определял, является ли логическая вложенность следствием структурной вложенности. XML в некоторой степени более жёсток в этом отношении, но не HTML.

НЛО прилетело и опубликовало эту надпись здесь



Shedal

9 мая 2012 в 23:50



0



Ну, я как раз имел в виду, что парсеры HTML в браузерах очень «мягки» — что бы разработчик ни написал, парсер попытается угадать семантику и построит DOM хотя бы в каком-то виде.

В XML же, per se, вообще нет семантики. Семантика задаётся структурой конкретного XML-based формата, и там уже, обычно, ни от семантики, ни от структуры, нельзя отойти ни на шаг (хотя, конечно, тоже depends).

В общем, мы с вами говорим об одном и том же, но разными словами :)

НЛО прилетело и опубликовало эту надпись здесь



StrangeAttractor

9 мая 2012 в 05:36



0



Я бы ещё добавил «используйте HTML5, но пишите так, чтобы он был валидным XML» (закрывайте теги, б-ть!). Увы, некоторые примеры в статье демонстрируют обратное, и это огорчает.



monolithed

9 мая 2012 в 12:49



0



На самом деле не совсем понятно кому адресованы эти рекомендации, но использовать XML-сериализацию (XHTML5) в HTML5 можно:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title> Title </title>
    <meta charset="UTF-8" />
  </head>
  <body>
    <input type="text" placeholder="input text" />
  </body>
</html>
```

PS:

- первая строка обязательна только для документов имеющих кодировку отличную от UTF-8.
- вторая строка опциональна.
- третья обязательна

НЛО прилетело и опубликовало эту надпись здесь



АНТОН

9 мая 2012 в 13:08



0



Замечательное руководство. Только не совсем понятно, что будет тем, кто ему будут следовать?

Улучшение видимости сайта в Гугле? — Сомнительно.

Большая производительность Хрома на сайте? — Да он и так неплохо работает.

Гугл решил взять на себя руководство над верстальщиками и вебмастерами? — Фигу ему с маслом.

Зачем же нужно это руководство?



Shedal

9 мая 2012 в 14:36



+1



Это руководство для программистов и верстальщиков, которые работают в Гугле. Оно теперь просто расшарено на паблик, чтобы и сторонние люди могли пользоваться этими стандартами (если они им подходят).



vsb

9 мая 2012 в 19:20



0



У языка Java есть одно интересное свойство. Существует официальное руководство от Sun по оформлению исходного кода. Вплоть до мелочей. И этому оформлению следуют подавляющее большинство программистов. В итоге очень большой процент кода на Java выглядит однообразно. И это помогает, когда приходится смотреть чужой код. Если HTML/CSS придёт к этому, будет хорошо.



shoorick

9 мая 2012 в 20:23



0



Недавно, изучая статистику посещений сайта, обнаружил, что некоторые браузеры ссылку вида `//host.name/path` воспринимают всё-таки как путь от корня текущего сайта, а не как *текущий_протокол*://host.name/path и в результате шлют запросы не к тому серверу, куда задумано, а к текущему, то есть на *текущий_протокол*://текущий_сайт//host.name/path.

Получается, что надо либо в ссылках указывать протокол явно (тратить на это аж целых пять байт, а то и шесть!), либо настраивать перенаправление на своём сайте (в апаче — `mod_alias` или `mod_rewrite`).



fatal

9 мая 2012 в 23:25

h ↑

+1

↑

↓

Какие это были браузеры, если не секрет?



zBit

9 мая 2012 в 23:05

#

0

↑

↓

Ух ё. Был в шоке от

Маша

честно. Не знал, что так можно. И с аналогично.

И ещё я был сильно удивлён когда проверил bootstrap.css в jigsaw.w3.org/css-validator/
К сожалению, мы обнаружили следующие ошибки (835)



fatal

9 мая 2012 в 23:23

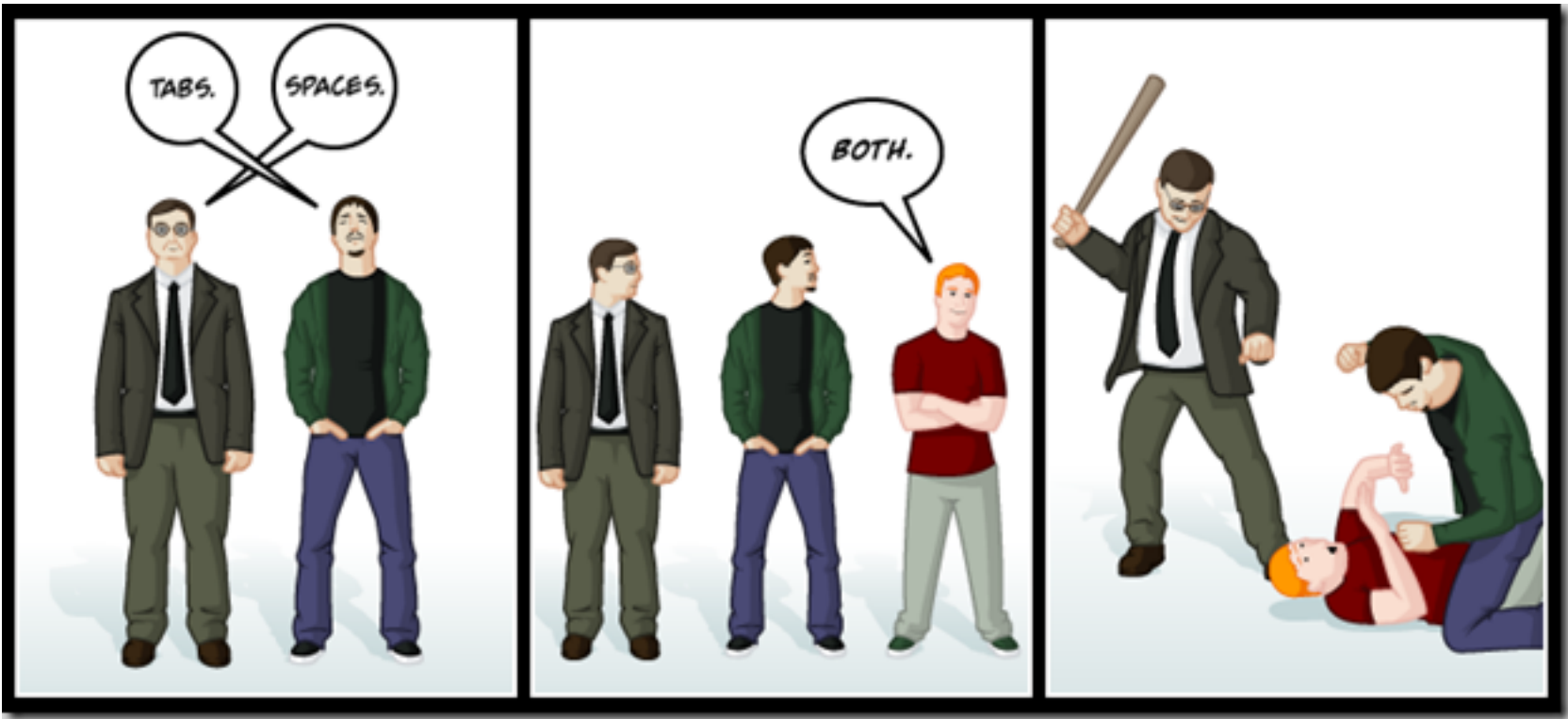
#

+1


↑

↓

Согласен со всем, кроме использования пробелов вместо табуляции :P



lea.verou.me/2012/01/why-tabs-are-clearly-superior/



negodnik

9 мая 2012 в 23:49


#

0

↑

↓

Насчет мнемоников спорно, т.к. в редакторе не всегда можно замыленным глазом отличить
-, -, -



Athari

10 мая 2012 в 18:40

h ↑

0

↑

↓

За различия между «-» и «—» вас не расстреляют. Более того, это никто никогда не заметит. Это же надо знать длину в пикселях минуса и короткого тире для всех шрифтов всех размеров, и при этом обладать глазомером, который различит разницу в один пиксель.



negodnik

10 мая 2012 в 18:48

h ↑

0

↑

↓

Ну а что насчет длинного тире?

И бтв

Пользователь [Athari](#) ответил на ваш комментарий к хабратопику "Руководство по различиям между «-» и «—» вас не расстреляют. Более того, это



Athari

10 мая 2012 в 19:23

h ↑

0

↑

↓

Длинное отличить легко.

Хе, у меня другой шрифт. :) 1-2+3-4+5-6

Длинное отличить легко.

Хе, у меня другой шрифт. :) 1-2+3-4+5-6

> background: url(//www.google.com/images/example);

выглядит бредовато...

про мнемоники — спорно, про неиспользование необязательных тегов (html, head, body) — тоже.

> HTML5 (HTML синтаксис) рекомендуется для всех html-документов: <!DOCTYPE html>.

Тогда уж так:

<!doctype html>

как это на (http://www.google.ru/). Или на //www.google.ru/...? :)

Отступы

Всегда используйте для отступа два пробела.

Не используйте табуляцию и не смешивайте табуляцию с пробелами.

то есть 2 пробела = 2 байта — круче чем 1 табуляция = 1 байт?

меня просто бесит когда я вижу в качестве отступов пробелы. это конечно сугубо мои проблемы, но начать педалить дальше пока не приведу код к нормальному виду с отступами в виде табуляций и т.п. просто не получается
есть тут у меня проект в котором товарищи на back-end очень любят пробелы (при том что server-side) на Java там 100500 пустых линий и переносов (тримать пустые линии не хотят бояться что что-то поломается)
Сделал исследование маленькое — главную страницу сохранил заменил пробелы на табуляцию — получил — 25kb веса страницы. убрал пустые линии, пробелы и отступы не значащие при сохранении отступов в коде для «читабельности» — еще минус 10kb
все я вам пожаловался :(

получил «минус» 25kb (парсер поставил тире)

НЛО прилетело и опубликовало эту надпись здесь

Shift + Tab лучше чем 2 backspace


java-криптопроцессор, используется в Symantec Network Security для ЭЦП логов.

/* Не рекомендуется: используется подчеркивание вместо дефиса */ .error_status {}
Не согласен. Два раза клацнул и выделил, а так нужно мышкой проводить, экономит время.

Только зарегистрированные пользователи могут оставлять комментарии. [Войдите](#), пожалуйста.

Интересные публикации



 РНР-Дайджест № 82 – интересные новости, материалы и инструменты (14 – 27 марта 2016) 0

 Японский рентгеновский телескоп распался на орбите 12

- Н

Русский нейросетевой чатбот

5
- СТ

Из-за мутации в X-хромосоме некоторые женщины различают в 100 раз больше цветов, чем обычные люди

36
- Н

Casual-заготовки под Swift

6
- М

Бизнесмен-философ: основатель и гендиректор Slack Стюарт Баттерфилд – рубрика «Бизнес-персона»

0
- СТ

Джули Рубикон. Признание бывшего сотрудника Facebook

35
- Н

Опыт запуска ANCI в VxWorks653

7
- М

Этот 25-летний специалист по информационным технологиям помогает разрушить привычный для нас мир финансов

5
- СТ

Новая тактика Роскомнадзора: блокирование целого сайта, а не отдельных страниц

23

Вакансии

Мой круг

Senior Android Developer

Москва • Полный рабочий день • от 200 000 руб.

Full Stack разработчик (Ruby on Rails разработчик)

Москва • Полный рабочий день

Front-end разработчик (Разработчик интерфейсов)

Москва • Полный рабочий день

Системный администратор (junior level)

Полный рабочий день

Rails разработчик

Москва • Полный рабочий день • от 130 000 до 150 000 руб.

Создать резюме

Разместить вакансию

Заказы

Фрилансим

Запустить OpenVPN через ICMP

27.03.2016 • 0 откликов • Цена договорная

Django-программирование

27.03.2016 • 0 откликов • Цена договорная

Таргетолог на проект(ы)

27.03.2016 • 0 откликов • Цена договорная

Скрипт аплоада и редактирования видео на js

27.03.2016 • 3 отклика

Правки верстки на бутстрапе

27.03.2016 • 4 отклика

Зарегистрироваться

Разместить заказ