



24,00

Рейтинг

Uprock

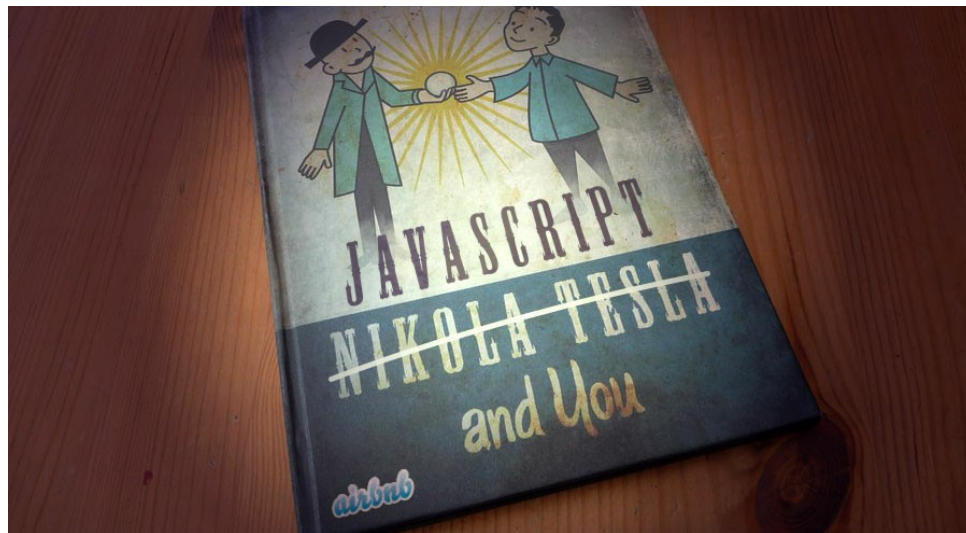
Компания



Jabher 4 декабря 2013 в 19:23

Перевод AirBnB Style Guide

JavaScript, Блог компании Uprock



У нас, как и у всех, изначально в разработке использовались стандарты написания кода, но на деле они ограничивались примерно таким:

— В кэмелкейсе пишем?
— Да, как обычно, в кэмелкейсе
... прошло две недели...
— Мы ж в кэмелкейсе договаривались!!!

В нашем случае не было разве, что последней фразы.

Уже давно хотелось как-то это систематизировать, но никак не доходили руки до выбора между стилями от jQuery, Google, ideomatic.js и Crockford.

Когда месяц с небольшим назад в trending гитхаба попало руководство от AirBnB, оно тут же попало на глаза...

А неделю назад **мы его перевели**, в первую очередь для собственных нужд, но не поделиться с сообществом не могли.

- Оригинал: <https://github.com/airbnb/javascript>
- Перевод: <https://github.com/uprock/javascript>

Прочитать крайне советую как минимум всем начинающим js-разработчикам.

Чем оно так отличается от других руководств, и чем оно так приглянулось нам?

Дело в том, что оно не столько указывает на то, как стилистически оформлять код, сколько рассказывает, как нужно грамотно писать на JavaScript. Это не только список указаний для JSLint, но платформа для того, чтобы начать писать хорошо: этот текст, который можно вдумчиво прочитать за час, дает стажерам больше понимания JavaScript, чем день собственных проб и ошибок и вопросов старшим товарищам.

Хардкорная конфа по C++. Мы приглашаем только профи.

Приходите

Реклама

ИНФОРМАЦИЯ

Локация	Санкт-Петербург Россия
Сайт	uprock.pro
Численность	Неизвестно
Дата регистрации	17 сентября 2013 г.

Относительно оформления — в этом руководстве используются «популярные» выборы.

Примеры:

табуляция в 2 пробела используется, так же в Google, npm, Node.js, Idiomatic

Всегда использовать точку с запятой, как в Google, Node.js, Crockford

Объявление переменных с «висячими» запятыми в одном var, как в Idiomatic, jQuery

Что интересно тут указывается даже подобный момент:

Объявляйте переменные, которым не присваивается значение, в конце. Это удобно, когда вам необходимо задать значение одной из этих переменных на базе уже присвоенных значений.

camelCase для переменных — как в Google, npm, Node, Idiomatic

При этом в этом гайде явно указывается:

Используйте camelCase для именования объектов, функций и переменных.
Используйте PascalCase для именования конструкторов классов.

Понятно и ясно объясняются некоторые моменты, например:

Дополнительная запятая в конце объектов: Нет. Она способна вызвать проблемы с IE6/7 и IE9 в режиме совместимости. В некоторых реализациях ES3 запятая в конце массива увеличивает его длину на 1, что может вызвать проблемы. Этот вопрос был прояснен только в ES5 (оригинал):

Редакция ECMAScript 5 однозначно устанавливает факт, что запятая в конце ArrayInitialiser не должна увеличивать длину массива. Это несемантическое изменение от Редакции ECMAScript 3, но некоторые реализации до этого некорректно разрешали этот вопрос.

Наглядно объясняются механизмы работы «всплытия» объявлений переменных и функций внутри области видимости.

Объявление анонимной функции поднимает к верху области видимости саму переменную, но не ее значение.
Именованные функции поднимают на верх области видимости переменную, но не ее значение. Имя функции при этом недоступно в области видимости переменной и доступно только изнутри.
Объявления функции поднимают на верх текущей области видимости и имя, и свое значение.

Каждый пример понятно и наглядно иллюстрирован тестовым блоком кода, очень многие вещи, выбранные из соображений быстродействия или особенностей работы, сопровождаются ссылками на jsPerf, спецификацию ECMAScript (для тех недоверчивых зануд, кому недостаточно простых объяснений) или отдельные весьма интересные статьи по данной тематике.

В общем — как минимум это весьма интересное чтение, и если вы давно собирались формализовать стилистику написания кода — сейчас самое время.

В комплекте идет набор ссылок на другие основные гайды, список команд, которые уже используют данный стиль написания кода, сайты, которые стоит почитать и еще много полезных ссылок, и самое главное — конфигурация для JSLint в Sublime.

Вот краткий пересказ содержимого для самых занятых:

- Всегда использовать краткий синтаксис для создания объектов и массивов;
- Для строк использовать одиночные кавычки, максимальная длина строки — 80 символов, перенос через +;
- Все объявления переменных (и функций) в начале блока, через запятую и с одной var на всю функции. Объявление функции (не присваивание) недопустимо в условных блоках;

- Переменные без присваивания значения идут в конце объявления переменных;
- Доступ к свойствам и методам объекта всегда по возможности через точку, запрос вроде `user['get']` под запретом (если это не требуется для чего-то особого в коде);
- Всегда проверять строгое равенство и использовать приведение типов;
- Фигурные скобки только для многострочных блоков кода. Если идет один оператор с новой строки — его тоже нужно обернуть в `{}`;
- Многострочные комментарии всегда через `/** */`, использовать JSDoc;
- Однострочные комментарии только через `//`, всегда предваряют описываемое, перед ними ставится одна пустая строка;
- Табуляция — два пробела, перез открывающей скобкой всегда один пробел;
- Использовать логические отступы в цепочках вызовов, каждый вызов на новой строке. Так, в jQuery, например, отступ ставится перед и после каждой смены целевого блока (`.find`, `.closest`, `.children` и так далее);
- Точки с запятой — всегда, где это необходимо;
- Приведение типов по возможности в начале операции, для чисел всегда использовать `parseFloat` или `Number`. Побитовые операции для приведения допустимы ради быстродействия, каждую такую операцию нужно комментировать;
- Все функции именуются, `camelCase` для всех переменных и функций, `PascalCase` для конструкторов. У приватных методов и свойств ставим вначале `"_ "` Для ссылки на `this` используем `_this`;
- Универсальные ассесоры не используются, геттеры и сеттеры назначаются для каждого свойства или делаются общие `.get` и `.set`. Замена прототипа — зло, прототипы можно только расширять;
- В событие всегда передается в качестве объект, чтобы его можно было расширить в процессе всплытия;
- jQuery-переменным задается префикс `$`;


От себя мы добавили буквально несколько разъясняющих блоков и внесли одно изменение: в оригинале предлагалось использовать комментарии `//FIXME` и `//TODO`, но IDE и редакторы кода обычно подсвечивают только `//TODO`, так что вместо `//FIXME` предлагается использовать `//TODO FIXME` для более простой навигации по автоматически сгенерированному индексу.

На всякий случай дублирую ссылки:

- Оригинал: <https://github.com/airbnb/javascript>
- Перевод: <https://github.com/uprock/javascript>

Теги: [javascript](#), [airbnb](#), [style guide](#), [styleguide](#), [guideline](#)


+55	385	37,5k	92
-----	-----	-------	----



Uprock

Компания

24,00



58,2

Карма

0,0

Рейтинг

28

Подписчики

Сева Родионов

@Jabher

Пользователь

Поделиться публикацией



ПОХОЖИЕ ПУБЛИКАЦИИ

23 апреля 2014 в 15:13

Прошел GTA – запили плеер!

+10 23,1k 79 21

8 октября 2013 в 18:26

CornerJS, или директивы «как в AngularJS», только лучше

+37 12,3k 125 24

Комментарии 92



spmbt 4 декабря 2013 в 19:51

-1

По 6.

Зачем всегда строгое равенство для выражений, выдающих всегда один тип данных и сравнивающих с этим же типом? Например, `if(s.indexOf('a') == 1)...`?



Jabher 4 декабря 2013 в 20:07

+7

Все банально — единообразие. *Ordnung macht frei* и все такое.

Совсем недавно начал понимать, зачем на самом деле в гайдах всегда прописывается точное количество пробелов, кавычки и так далее... Все оказалось просто, это нужно для быстрого поиска по огромным проектам. Одно дело, когда ты знаешь, что нужно найти что-то вроде `$('#', а другое — что надо еще учесть варианты $('#, $("# и так далее.`

Тут то же самое — вполне может быть ситуация, что надо найти, например, `indexOf(name) === 5`, например, и в этой ситуации расписывать еще и `= {2,3}` регекспом каждый раз печально. Недавно пришлось разгребать один проект — так там из-за нестыковок в нотации у разработчиков несколько раз реально 10-15 минут искал определенное обращение. К счастью, с ним пришлось работать совсем недолго.

К подобной практике приучивают и препроцессоры, тот же самый `coffeeScript`, например, в принципе не умеет делать `==`. Если написать `if a==b`, он превратит это в `if (a===b)`. Когда начинаешь пользоваться точным равенством `===`, какое-то время использование сравнение через `==` становится, гхм, непривычным. Просто попробуйте для сравнения.



Aingis 5 декабря 2013 в 15:56

-1

На самом деле случаев, когда действительно требуется тройное равенство (неравенство), крайне мало. И у Ангуса Кролла, на которого ссылаются в статье об этом написано. Если важен тип переменной, то он обычно проверяется загодя. В результате получается совершенно лишняя избыточность единообразного кода, а писать по три равно очень быстро надоедает.



inook 5 декабря 2013 в 00:57

-4

Обычно пишу так `if (!!~s.indexOf('a'))`, с первого взгляда покажется бредом. Но после таза привыкают к такому выражению.



Jabher 5 декабря 2013 в 10:58

-1

А я-то думал, что самое извращенное что может быть это

```
if (s.indexOf('a')+1)
```



Mithgol 5 декабря 2013 в 11:11

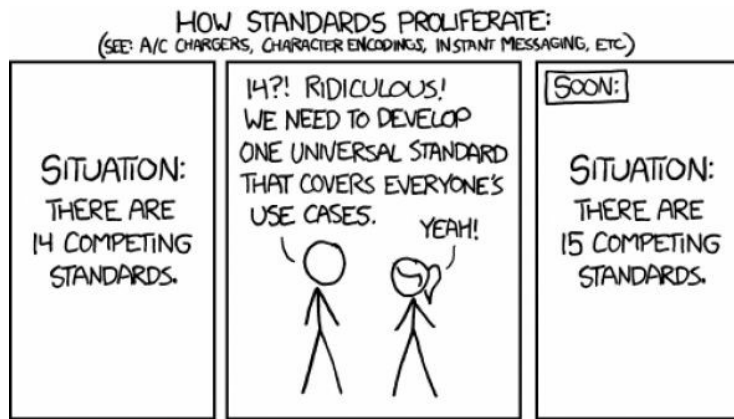
+2

Простого `~s.indexOf('a')` было бы достаточно, потому что «!!» внутри «if» подразумевается.



Talyutin 4 декабря 2013 в 19:53

+2



Agent_J 4 декабря 2013 в 20:00

+10

PNG vs JPEG

achekalin 5 декабря 2013 в 09:36

+7

Иногда хочется, чтобы на Хабре была страница со всеми цитируемыми картинками: двумя выше, с буханкой из тролейбуса, с еще десятком, которые постоянно, к месту и нет, упоминаются, хотя бы по одной-две, в подобных постах — и именно на эту страницу все комментаторы, считающие графику свежим, неизбитым ответом, ссылались бы. Чес-слово, ребят, ну уже даже не смешно.

Jabher 4 декабря 2013 в 20:12

-2

тут скорее «14 стандартов? чуваки, а давайте посмотрим решения, которые использует большинство, и возьмем их, а заодно опишем почему так дёпять — хорошо».

Talyutin 4 декабря 2013 в 20:38

+1

Ну, как мне показалось, не совсем.
Было «от jQuery, Google, ideomatic.js и Crockford».
Из них взяли все, что посчитали самым удобным и появился еще один Style Guide от AirBnB.

Jabher 4 декабря 2013 в 20:57

0

на самом деле есть еще nodeguide.com/style.html, npmjs.org/doc/coding-style.html и github.com/Seravo/js-winning-style. Но это уже мелочи.

Agent_J 4 декабря 2013 в 21:23

+4

Спасибо вашей команде за перевод, но я не заметил, чтобы там описывалось «почему», просто установка «Используйте табуляцию из двух пробелов». А почему? Почему не 4 или 8?

Jabher 4 декабря 2013 в 21:40

+2

Просто потому что это популярное решение, вот и все. Большинство придерживается того же стиля, и им будет проще перейти: это плюс как для новых разработчиков, так и для опенсорс-сообщества, например.

Agent_J 4 декабря 2013 в 21:49

+3

Ну во-первых предъявите статистику, прежде чем утверждать, что этого способа придерживается большинство (я не про пробелы, а про все утверждения).
А во-вторых «так делает большинство» это не причина. Или вы сторонник мнения «миллиард китайцев не может ошибаться»?
Просто вы навязываете ничем не обоснованный подход. Либо объясняйте почему именно так, а не иначе, либо называйте это «принятые в нашей команде стандарты»

Jabher 4 декабря 2013 в 21:57

0

хорошо, я повторяюсь.
Руководства по написанию JavaScript — Google, npm, Node.js, Idiomatic — придерживаются софтверных табов в 2 пробела.
Четыре пробела предлагается только крокфордской нотацией.

Первые четыре руководства используются большинством веб-студий, которые в принципе используют какие-либо гайды. Когда мы выбирали подходящую стилистику — я опросил не один десяток веб-студий и агентств, и если там и использовались стили, то использовались Google или Idiomatic, была всего одна команда, которая сослалась на Крокфорд как основу.

В IntelliJ по умолчанию используются софтверные отступы в 2 таба — а я думаю, вы не станете спорить, что это на данный момент наиболее популярная среда разработки.

Относительно «это не причина». Это причина: большинство людей пришло к именно двум табам, поскольку это удобно. Это субъективно удобно для большинства, и если предстоит работать с большинством — сделайте так, чтобы большинству было удобно. Если оно вам нужно, конечно.



Regis 4 декабря 2013 в 22:18

+6

В IntelliJ по умолчанию используются softwerные отступы в 2 таба — а я думаю, вы не станете спорить, что это на данный момент наиболее популярная среда разработки.

В IntelliJ по умолчанию 4 пробела.



azproduction 4 декабря 2013 в 22:21

+2

`.editorconfig` + `EditorConfig` и конец этому занудному холивару про табы и пробелы. Сколько уж можно!



Regis 4 декабря 2013 в 22:22

+8

большинство людей пришло к именно двум табам

100% проектов, на которых я был за 7 лет работы использовали исключительно 4 пробела (я не считаю сторонние библиотеки). Но я не буду утверждать, что «большинство» пришло к 4 табам. Чтобы вообще заикаться про «большинство» нужно предоставлять какую-то конкретную, достаточно большую выборку, а не полагаться на личный опыт.



Agent_J 4 декабря 2013 в 23:42

+1

я не про пробелы, а про все утверждения

Раз уж вы так активно ринулись доказывать, то пройдитесь по всему списку, а не одному пункту.

Лично я не против стандартов, я поддерживаю многое из того, что описано в руководстве. Но вы говорите о нём не как об очередном стандарте, а как о неопровержимой истине, да ещё и смеее утверждать, что там сказано «почему так делать хорошо». Или я слеп и не увидел «почему», или вы врётё.

... и выше вам уже сказали «В IntelliJ по умолчанию 4 пробела» (хотя, подозреваю, это может зависеть от конкретного продукта и версии)



azproduction 4 декабря 2013 в 21:58

+4

Есть статистика DailyJS за 2012 без учета количества пробелов

Indentation with spaces 1421 53%

Indentation with tabs 1038 39%

dailyjs.com/files/2012-survey-summary.pdf

Сейчас идет голосование 2013 года dailyjs.com/2013/11/28/javascript-survey/ там набор вопросов больше.



AleksDesker 5 декабря 2013 в 02:41

0

Какое-то странное голосование, для библиотек куча неизвестных названий и нет JQuery, для источников reusable code аналогично, нет stackoverflow.com, у меня оттуда 99%.



azproduction 5 декабря 2013 в 11:13

0

Наверно именно поэтому, что 99% их и нет. А то от этого голосования не было бы смысла: 99 vs 1.



AleksDesker 5 декабря 2013 в 11:17

0

Надеюсь они воспользуются вашей логикой на следующих выборах президента России.



m_z 5 декабря 2013 в 11:31

+1

Просто потому что это популярное решение

Интересно, почему у JS разработчиков это более популярное решение? В популярных языках (Java, C#, PHP, Python) везде 4 пробела, а здесь два.



anotherpit 5 декабря 2013 в 12:35

0

потому что в js куда сильнее, чем в перечисленных языках, распространена передача колбэков

в аргументы вызовов. а от вложенных колбэков indentation пухнет



aTei 6 декабря 2013 в 02:45

0

2 пробела у Ruby и CoffeeScript.

Раньше, сложнее было понять, где отступ, потому 4 пробела было удобней, чем 2. Сейчас с дополнительной подсветкой (IDEA) надобность в 4 пробелах отпадает.



Regis 4 декабря 2013 в 22:16

0

Комбинация из частей «стандартов» — новый «стандарт». Другое дело, если бы был выбран какой-то один существующий и просто объявлен наилучшим.



zag2art 4 декабря 2013 в 19:57

0

Отличные ребята, недавно делал перевод их статьи: [Изоморфный JavaScript — будущее веб-приложений](#)



Jabher 4 декабря 2013 в 20:10

+1

О, а я практически по ней для одного из проектов с нуля сделал fullstack фреймворк в три сотни строк :) Спасибо за тот перевод, он очень помог.



zag2art 4 декабря 2013 в 20:21

0

Используя «изоморфный туториал»?



Jabher 4 декабря 2013 в 20:30

+1

Ну, вы скорее подкрепили мои намерения сделать его, но как минимум мне очень сильно помогло упоминание browserify, о котором я до этого не слышал, до этого использовал такой костыль (подсмотренный у одного товарища):

```
if (typeof module !== 'undefined') {
  module.exports = App;
} else if (typeof exports !== 'undefined') {
  exports.App = App;
} else if (!('WeakMap' in global)) {
  global.App = App;
}
```

Если интересно — для рутинга на клиенте использую очень похожий на роутер express page.js, шаблоны — jade с рендерингом в js-модуль. Но это я потом расскажу, в одном из следующих постов.



azproduction 4 декабря 2013 в 21:52

+1

Посмотрите еще в сторону [LMD](#).



zag2art 5 декабря 2013 в 12:19

0

Прикольно. Ждем поста.



hell0w0rd 4 декабря 2013 в 20:19

+4

```
// плохо
var items =.getItems();
var goSportsTeam = true;
var dragonball = 'z';

// хорошо
var items = getItems(),
    goSportsTeam = true,
    dragonball = 'z';
```

Да почему же вот так? Что в этом хорошего?

Если все переменные объявить через var — я любую могу закомментировать/удалить без лишних телодвижений, а какие преимущества у такого синтаксиса?



Jabher 4 декабря 2013 в 20:25

-4

Первое.

Не знаю как у вас, а у меня в IntelliJ любая строка интеллектуально(с учетом окружения) комментируется по ctrl+/,

Второе.

Появляется возможность сразу же визуальнo отдепить блок, где нахoдятся переменные, и блок, где нахoдится основной код. Соответственно — видно, куда можно добавить еще одну переменную. А мы помним, что их надо добавлять в начало окружения, так как все же неправильно доверяться компилятору в переносе в рамках окружения — есть риск самому запутаться, если объявить переменную где-либо в конце. Соответственно добавлять переменные нужно именно в начале. Да и сразу можно увидеть список всех переменных окружения.

Третье.

Я не помню на моей памяти ни одного случая, когда мне реально нужно было комментировать переменную. Куски кода — да, но переменную — ни разу, максимум изменять значение или переименовывать: переменные при инициализации начинают хранить данные, а не совершать действия, и эти данные либо могут понадобиться (так что оставляем переменную), либо точно не понадобятся (удаляем). Сценарий комментирования переменной мне с трудом представляется, единственное, что приходит в голову — это нужно для заигрываний с глобальным скопом.



hell0w0rd 4 декабря 2013 в 20:45

+6

Первое, третье — ок, второе — не понимаю, как несколько var мешают читабельности. Ну отделите блок переменных пустой строкой и все.



azproduction 4 декабря 2013 в 21:24

+5

ALL code is ugly. Yours, mine, everyone's. Code Is Ugly. Just face it. When someone says "beautiful code", hear "beautiful rotten entrails". The only beautiful code, the most clean code, is no code at all. If it's there, it's ugly. The unfortunate fact of software development is that, in order to create beautiful software, we must use ugly code.

—
[A better coding convention for lists and object literals in JavaScript](#)

В общем, у каждого подхода есть свои особенности — нельзя говорить, что так плохо. Это стиль кодирования который нужно просто принять придя работать в компанию. AirBnB выложили свое видение.

var ни чуть ни хуже визуальнo отбивает блок и при добавлении или удалении новой переменной в коммит попадает +1 или -1 строка, а не -1 строка и +2 строки. В общем то на каждое ваше утверждение можно найти противовес. Ну и оправдания вида «не приходилось» — эт не спортивно ;)

PS я использую 2 подход т.к. вся моя команда приняла решение использовать его.



malroc 5 декабря 2013 в 01:37

0

Не знаю как у вас, а у меня в IntelliJ любая строка интеллектуально(с учетом окружения) комментируется по ctrl+/
—

Не знаю, как там комментирует IntelliJ, но скажем у вас три приведённых переменных, вам надо закомментировать вторую. А потом раскомментировать. Мы получим в итоге 3 исходных строки? Честно сказать, не верю, с комментированием всё более-менее понятно, а в момент раскомментирования будет неоднозначность, это неформализуемая процедура в общем случае.

И потом, получается стандарт привязывает к определённой IDE? В общем, более чем сомнительный момент.



glamcoder 4 декабря 2013 в 20:26

+1

В читаемости. Вы читаете код гораздо чаще, чем его комментируете/удаляете. И когда в коде встречается строка, и в ней var вынесен в начало блока, а далее с отступом идут строки объявления переменных, то сразу понятно, где начало блока, а где конец.

Когда переменные объявляются каждая на своей строчке, то глазу труднее выделить, где что находится.



Regis 4 декабря 2013 в 22:05

+1

В читаемости. Вы читаете код гораздо чаще, чем его комментируете/удаляете.

С этим никто и не спорит.

И когда в коде встречается строка, и в ней var вынесен в начало блока, а далее с отступом идут строки объявления переменных, то сразу понятно, где начало блока, а где конец.

Проблема в том, что это вам нужно видеть начало и конце «блока переменных». У других таких блоков, возможно, в принципе нет. В итоге получаем решение проблемы которую сами и создали :(



IonDen 4 декабря 2013 в 20:41

+1

Ну некоторые вещи сомнительны, непонятно почему:

— 4 пробела плохо, 2 хорошо,

— "name" плохо, 'name' хорошо,

— self, that плохо, _this хорошо


 **hell0w0rd** 4 декабря 2013 в 20:56

0

Добавлю:


— «Оставляйте новую строку в конце файла.»

— `dragon.age()` ; `dragon.age(25)` ; , в `jquery` такое повсеместно используется и пока никто не жаловался

 **cyberface** 4 декабря 2013 в 21:21

+1

stackoverflow.com/questions/729692/why-should-files-end-with-a-newline

 **Jabher** 4 декабря 2013 в 21:00

-1

— 4 пробела плохо, 2 хорошо

наиболее популярное решение, нужно для читаемости при передаче проектов просто: разработчики смущаются, видя разницу в отступах в два раза. Этим пользуются почти все, так что «придерживаемся большинства».

— «name» плохо, 'name' хорошо

Опять же «придерживаемся большинства». А единообразие в кавычках необходимо, т.к. очень важно для поиска. Я выше пример писал.

— `self`, `that` плохо, `_this` хорошо

`this` обычно подсвечивается в IDE как спецслово, что помогает его визуально отличать. префикс `_` тоже помогает визуально отделять ее от обычных переменных. `Self` и `that` на первый взгляд кажутся тоже обычными переменными.

 **ishaba** 4 декабря 2013 в 21:11

+2

«name» плохо, 'name' хорошо

потому что

```
'<div class="item">'
```

`self`, `that` плохо, `_this` хорошо

`_` визуально искать гораздо проще, `self` не отличить от остального кода

Добавлю:

— «Оставляйте новую строку в конце файла.»

очевидно для того чтобы при склеивании, последняя строка файла `n` не спуталась с первой строкой файла `n+1`

 **azproduction** 4 декабря 2013 в 21:28

0

'name' хорошо

1) При поиске строки 'name' нужно искать только один раз.

2) Одиночная кавычка легче на вид.

3) Во многих ЯП двойная кавычка используется для инлайн шаблонов — будет меньше путаницы если использовать одинарную.

 **hell0w0rd** 4 декабря 2013 в 21:30

+3


Очевидно что при склеивании, склейщик должен сам об этом позаботиться.

Спасибо @cyberface, частенько вызывая `cat` для разных файлов наткнулся на отсутствие перевода строки перед `$`, добавлю в настройки редакторов

 **azproduction** 4 декабря 2013 в 21:33

+5

Опять же нельзя однозначно утверждать про `self` vs `_this`. `self` используется в ряде языков как ссылка на лексический контекст (и тем кто пишет на том языке и на JS проще понять такой код). Плюс подчеркивание у `_this` визуально зашумляет код. Да и есть еще 3 подход, когда запрещается замыкать псевдо-лексический контекст. Те всегда использовать `.bind(this)` или другим способом прокидывать контекст.

 **Agent_J** 4 декабря 2013 в 21:41

0

потому что

```
'<div class="item">'
```

Значит я должен так писать?

```
alert('Error'+"\n"+'У вас ошибка!');
```

_ визуально искать гораздо проще

... особенно среди прочих псевдо-приватных переменных



PSDCoder 5 декабря 2013 в 00:10

+3

Значит я должен так писать?

```
alert('Error'+"\n"+'У вас ошибка!');
```

Насколько я помню одинарные и двойные кавычки в js равнозначны, это Вам не PHP =)

Т.е. вполне достаточно написать просто:

```
alert('Error\nУ вас ошибка!');
```

без конкатенаций...



Agent_J 5 декабря 2013 в 12:10

+1

Совершенно верно. Посыпаю голову пеплом)



malroc 5 декабря 2013 в 01:25

+1

потому что

```
'<div class=«item»>'
```

Да, многие используют одинарные кавычки в JS именно по этой причине. В своё время тоже так делал. Была такая система: в PHP (когда ещё писал на PHP) использовал двойные кавычки, в джаваскрипте одинарные, в HTML двойные. В итоге, можно было в PHP без проблем генерировать блоки на JS, а в JS на HTML (проблемы только в случае двойной вложенности — т.е. HTML внутри JS внутри PHP).

Но дело в том, что такой стиль — это уже немного прошлое десятилетие. Во времена повсеместного использования шаблонизаторов писать HTML-блок внутри строки — дурной тон. Поэтому уже года два использую только двойные кавычки везде, для унификации.

Кстати, если на то пошло, то никто не мешает использовать в HTML одинарные кавычки (это допускается стандартом и прекрасно понимается всеми браузерами), т.е. пример на самом деле легко переворачивается.



ishaba 4 декабря 2013 в 21:11

0

спасибо за перевод, и особенно за краткий пересказ)

aretmy 4 декабря 2013 в 21:13

+1

А в чем необходимость ставить; всегда? Это для минификации?



Jabher 4 декабря 2013 в 21:15

-1

ИМЕННО



hell0w0rd 4 декабря 2013 в 21:35

+1

Не правда. github.com/twbs/bootstrap/blob/master/js/affix.js

Это опять же просто общепризнанный стиль.

Программист не должен заботиться о склейщиках, минификаторах и прочих инструментах — они сами должны заботиться о программисте. Они для того и созданы



Jabher 4 декабря 2013 в 21:44

0

По опыту скажу — в стороннем проекте видел, что js-файлы собирались не через grunt-contrib-uglify, а через grunt-contrib-concat, separator: ".

Так что это скорее защита от дурака, как и добавление еще одной строки в конце.

Ну и да, общепризнанный стиль тоже — чтобы всегда можно было продолжить писать код в файле, не проверяя, есть ли там в конце точка с запятой.



azproduction 4 декабря 2013 в 21:36

0

Это одно из мест где можно избежать ошибки при [auto semicolon insertion](#).



Aingis 5 декабря 2013 в 16:04

0

Самая распространённая ошибка: с return — и в этом случае «;» не поможет. А если код нормально, по тем же руководствам, то других ошибок возникать не должно. В итоге, если не знать про автоматическое расстановку точек с запятой, ошибки всё равно будут. А если знать, то никакой нужды в правиле нет.

azproduction 4 декабря 2013 в 21:49

+3

В событие всегда передается в качестве объект, чтобы его можно было расширить в процессе всплытия;

Это, я не побоюсь этого слова, крайне не верный подход использования событий. Обработчики событий не должны знать о существовании друг друга и соответственно должны воспринимать переданные им данные как свои личные (в режиме read only). События вне DOM не гарантируют порядок выполнения обработчиков. И не стоит завязываться на то, что кто первый подписался тот первый получит уведомление.

Вот вы представьте только: у вас 2 модуля оба подписаны на событие и один из них пропатчивает data — второй не ожидает такого поведения и все ломается. В DOM и в тех же jQuery.Event каждому обработчику передается уникальный event.

Jabher 4 декабря 2013 в 22:01

0

Для одновременных слушателей элемента это некорректно. Однако для всплытия — это вполне корректное поведение: клик всплывает к body однозначно позже того, как он сработал на a.

Это не лучшая практика, однако в некоторых условиях она допустима, и лучше предусмотреть этот сценарий, хотя использовать его в качестве основного не стоит.

Regis 4 декабря 2013 в 22:14

+1

jQuery-переменным задается префикс \$;

Опять же, спорное правило. Если на проекте jQuery выбран в качестве основы для всего и вся — писать каждый раз этот \$ становится глупо, так как все DOM-элементы всюду в коде уже заведомо обернуты в jQuery.

azproduction 4 декабря 2013 в 22:19

0

\$ визуально отделяет DOM(jQuery) от JS-a.

Regis 4 декабря 2013 в 22:23

+1

Каким образом? И зачем?

azproduction 4 декабря 2013 в 22:29

0

Первый вопрос странный или я его не понял — отвечу наличием \$ в начале имени переменной. Проще читать код и работать с ним, проще искать переменные, из имени переменной уже понятно как с ней работать, не нужно писать вербозных суффиксов: \$name vs name vs nameElement. Вот Angular все испортил со своим \$scope :)

Regis 4 декабря 2013 в 22:39

0

Я говорю про случай, когда в проекте nameElement нет. Вообще. Только jQuery-вские объекты.

Да и зачем искать переменные? IDE же их и так выделяет.

azproduction 4 декабря 2013 в 22:42

+1

Хорошо. jQuery это просто обертка над DOM те фактически DOM просто пофиксенный. Остается \$name vs name в этом случае первая переменная расскажет больше о своем содержании чем вторая (придется догадаться, почитать код).

Regis 4 декабря 2013 в 22:57

0

Чем jQuery-объекты такие особенные? В 99% случаев уже по названию переменной либо по ближайшему окружению будет ясно, что там элемент, а не что-то еще. Или вы также предлагаете под каждый тип данных заводить свой префикс?

Префикс \$ очень полезен, когда в одном коде у вас смешаны raw dom objects и jQuery обертки. Когда действительно у имен button и \$button будет разный смысл — \$ устраняет неоднозначность.

Но если в коде неоднозначности нет — чем \$button, \$mainTable, \$sidePanel будут лучше button, mainTable и sidePanel?

return 5 декабря 2013 в 01:29

+1

Разве обычных переменных в коде не бывает, только jQuery-обертки, да dom-ноды? :)

Regis 5 декабря 2013 в 02:35


-1

Можете привести пример реального кода, где по названию переменной будет не ясно, лежит ли в ней кнопка или, скажем, число?

 **k12th** 5 декабря 2013 в 11:09

-1

Потому что надо называть переменную `peopleCountDisplay` или `peopleCountBtn`. Псевдо-венгерская псевдо-нотация утомила уже.

 **k12th** 12 декабря 2013 в 15:08

0

Я просто оставлю это здесь: www.joelonsoftware.com/articles/Wrong.html

 **Regis** 5 декабря 2013 в 15:24

-1

В вашем примере вы используете `$` как замену более длинному суффиксу «Element» (или его вариациям).

Сравните:

```
var peopleCountElement = $('#people_count'), peopleCount = peopleCountElement.val();
```

Читаемость пострадала? Нет.

Хочу обратить внимание, что в данной ситуации это не тот «Element», который используется для выделения raw DOM elements в коде с лишь частичным использованием jQuery.

Вместо «Element» могут быть вариации на тему «Holder», «Div», «Span» и т.д. Зависит от контекста.

Пока не убедили.

 **azproduction** 5 декабря 2013 в 16:09

+2

Когда я читаю `peopleCountElement` я сразу вижу DOM элемент; `$peopleCount` — jQuery; `Element Div` и `Ко` будут сбивать с толку если в проекте есть jQuery.

 **Regis** 5 декабря 2013 в 16:22

-1

Еще раз — речь о проекте, где **только** jQuery. В подавляющем большинстве случаев у вас нет конфликта между названиями переменных как в примере выше.

НЛО прилетело и опубликовало эту надпись здесь

 **Regis** 7 декабря 2013 в 03:15

0

И что мешает конкретно в этом случае вам взять и, если ничего другого не приходит в голову, конкретно в этом случае воспользоваться префиксом `$`? Или ради этого обязательно захламлять весь остальной код префиксами?

НЛО прилетело и опубликовало эту надпись здесь

 **Regis** 7 декабря 2013 в 08:19

0

Ну не знаю. Для меня это спор о вкусах фломастеров. За последние несколько лет дай бог чтобы 4-5 раз возникала ситуация, где было бы реально нужно работать с DOM напрямую и префикс магически бы решал все проблемы :)

По примеру выше, буквально недавно нужно было порисовать стрелочки с канвасом. Код выглядел примерно вот так:

```
function getContext() {
    return $('#canvas').get(0).getContext('2d');
}
```

Собственно всё. Никакой драмы.

НЛО прилетело и опубликовало эту надпись здесь

 **Regis** 7 декабря 2013 в 19:22

0

А кто говорит, что он больше одного раза вызывается?) Что мешает закешировать результат?)

Код дробят на отдельные функции не только чтобы выделить дублирующуюся логику, но и чтобы просто выделить логически изолированные блоки.



merk 4 декабря 2013 в 22:17

+1

Благодаря вам узнал о github.com/trending, спасибо!

the_ghost 4 декабря 2013 в 22:58

+7

Прям всем и каждому хочется возразить, к каждому пункту можно придраться. Но знаете что скажу? Главное — чтобы все придерживались единого стандарта, выбранного потом, кровью и криками.



DarkPark 5 декабря 2013 в 01:20

0

может кому-то будет интересна собственная компиляция многих популярных руководств github.com/DarkPark/jscs



shcoderAlex 5 декабря 2013 в 06:31

0

Я хочу сказать Вам большое спасибо, за проделанную работу.



Mithgol 5 декабря 2013 в 10:04

0

Изложенная в руководстве AirBnB идея использовать массивы и метод join для конкатенации строк представляется мне слабоприемлемой (затрудняя постижение смысла кода), вне зависимости от её влияния на производительности джаваскрипта в IE.



paunch 5 декабря 2013 в 10:12

0

По поводу перевода

```
// TODO: должна быть возможность изменять значение через параметр функции
this.total = 0;
```

Мне кажется не верным переводить или писать комментарии на русском языке, даже в проектах, где все его знают.



zag2art 5 декабря 2013 в 12:29

-2

Почему, я вот наоборот считаю? Русскоязычная команда — комментарии должны быть русские. Зачем костыли? Другое дело, если ты делаешь свой «мега-фреймворк», который должен тебе принести межгалактическую известность...



bubuq 5 декабря 2013 в 14:24

+2

А завтра в вашу русскоязычную команду войдёт сотрудник из Пакистана.

aplic

5 декабря 2013 в 12:47

+4

Еще один. В большинстве конечно согласен, тем более что ничего нового они не придумали, но в топку такие tutoriales, которые несовместимы с большинством IDE и добавляют нераберихи в давние войны тупоконечников и остроконечников. Это я про табы в два пробела.

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

САМОЕ ЧИТАЕМОЕ

Сутки

Неделя

Месяц

Про одну девушку

+32

30,8k

125

90

Стальные ликвидаторы			
+64	12,8k	40	16
«Почта России» переходит с MS Office на отечественный офисный пакет. Стоимость перехода — 352 млн рублей			
+18	11,7k	1	65
DockerHub взломан			
+39	26,5k	27	13
Про одного парня			
+206	142k	777	246