

Раздел

[Качество кода](#)

Навигация по уроку

Краткость – сестра таланта!

Однобуквенные переменные

Используйте сокращения

Будьте абстрактны при выборе имени.

Проверка внимания

Русские слова и сокращения

Хитрые синонимы

Повторно используйте имена

Добавляйте подчёркивания

Покажите вашу любовь к разработке

Перекрывайте внешние переменные


Внимание... Сюр-при-из!

Мощные функции!

Итого

Комментарии

Поделиться

[Редактировать на GitHub](#)[🏠 → Язык программирования JavaScript](#)
[→ Качество кода](#) 13-го января 2020

Ниндзя-код

Предлагаю вашему вниманию советы мастеров древности.

Программисты прошлого использовали их, чтобы заострить разум тех, кто после них будет поддерживать код.

Гуру разработки при найме старательно ищут их применение в тестовых заданиях.

Новички иногда используют их ещё лучше, чем матёрые ниндзя.

Прочитайте их и решите, кто вы: ниндзя, новичок или, может быть, гуру?

Осторожно, ирония!

Многие пытались пройти по пути ниндзя. Мало, кто преуспел.

Краткость – сестра таланта!

Пишите «как короче», а не как понятнее. Покажите, насколько вы умны!

«Меньше букв» – уважительная причина для нарушения любых соглашений. Ваш верный помощник – возможности языка, использованные неочевидным образом.

Обратите внимание на оператор вопросительный знак ' ? ', например:

```
1 // код из jQuery
2 i = i ? i < 0 ? Math.max(0, len + i) : i : 0;
```

Разработчик, встретивший эту строку и попытавшийся понять, чему же всё-таки равно `i`, скорее всего, придёт к вам за разъяснениями. Смело скажите ему, что короче – это всегда лучше. Посвятите и его в пути ниндзя. Не забудьте вручить [Дао дэ цзин](#).

Однобуквенные переменные

Кто знает — не говорит. Кто говорит — не знает.

“ Лао-цзы

Ещё один способ писать быстрее – использовать короткие имена переменных. Называйте их `a`, `b` или `c`.

Короткая переменная прячется в коде лучше, чем ниндзя в лесу. Никто не сможет найти её, используя функцию «Поиск» текстового редактора. Более того, даже найдя – никто не сможет «расшифровать» её и догадаться, что она означает.

...Но есть одно исключение. В тех местах, где однобуквенные переменные общеприняты, например, в счётчике цикла – ни в коем случае не используйте стандартные названия `i`, `j`, `k`. Где угодно, только не здесь!

Остановите свой взыскательный взгляд на чём-нибудь более экзотическом. Например, `x` или `y`.

Эффективность этого подхода особенно заметна, если тело цикла занимает одну-две страницы (чем длиннее – тем лучше).

В этом случае заметить, что переменная – счётчик цикла, без пролистывания вверх, невозможно.

Раздел

Качество кода

Навигация по уроку

Краткость – сестра таланта!

Однобуквенные переменные

Используйте сокращения

Будьте абстрактны при выборе имени.

Проверка внимания

Русские слова и сокращения

Хитрые синонимы

Повторно используйте имена

Добавляйте подчёркивания

Покажите вашу любовь к разработке

Перекрывайте внешние переменные

Внимание... Сюр-при-из!

Мощные функции!

Итого

Комментарии

Поделиться



Редактировать на GitHub



Используйте сокращения

Если правила, принятые в вашей команде, запрещают использовать абстрактные имена или имена из одной буквы – сокращайте их.

Например:

- `list` → `lst`.
- `userAgent` → `ua`.
- `browser` → `brsr`.
- ...и т.д.

Только коллеги с хорошо развитой интуицией поймут такие имена. Вообще, старайтесь сокращать всё. Только одарённые интуицией люди достойны заниматься поддержкой вашего кода.

Будьте абстрактны при выборе имени.

*Лучший кувшин лепят всю жизнь,
Высокая музыка неподвластна слуху,
Великий образ не имеет формы.*

“ Лао-цзы

При выборе имени старайтесь применить максимально абстрактное слово, например `obj`, `data`, `value`, `item`, `elem` и т.п.

- **Идеальное имя для переменной: `data`**. Используйте это имя везде, где можно. В конце концов, каждая переменная содержит *данные*, не правда ли?

...Но что делать, если имя `data` уже занято? Попробуйте `value`, оно не менее универсально. Ведь каждая переменная содержит *значение*.

- **Называйте переменную по типу данных, которые она хранит: `str`, `num` ...**

Попробуйте! Сделают ли такие имена интереснее разработку? Как ни странно, да и намного!

Казалось бы, название переменной содержит информацию, говорит о том, что в переменной – число, объект или массив... С другой стороны, **когда непосвящённый будет разбирать этот код – он с удивлением обнаружит, что информации нет!**

Ведь как раз тип легко понять, запустив отладчик и посмотрев, что внутри. Но в чём смысл этой переменной? Что за массив/объект/число в ней хранится? Без долгой медитации над кодом тут не обойтись!

- **...Но что делать, если и эти имена закончились?** Просто добавьте цифру: `data1`, `item2`, `elem5` ...

Проверка внимания

Только истинно внимательный программист достоин понять ваш код. Но как проверить, достоин ли читающий?

Один из способов – использовать похожие имена переменных, например, `date` и `data`.

Бегло прочитать такой код почти невозможно. А уж заметить опечатку и поправить её... Ммммм... Мы здесь надолго, время попить чайку.

Русские слова и сокращения

Если вам *приходится* использовать длинные, понятные имена переменных – что поделаться... Но и здесь есть простор для творчества!

Назовите переменные «калькой» с русского языка или как-то «улучшите» английское слово.

В одном месте напишите `var ssilka`, в другом `var ssylka`, в третьем `var link`, в четвёртом – `var lnk` ... Это действительно великолепно работает и очень креативно!

Количество ошибок при поддержке такого кода увеличивается во много раз.

Хитрые синонимы

Очень трудно найти чёрную кошку в тёмной комнате, особенно, когда её там нет.

“ Конфуций

Чтобы было не скучно – используйте *похожие* названия для обозначения *одинаковых* действий.

Например, если метод показывает что-то на экране – начните его название с `display..` (скажем, `displayElement`), а в другом месте объявите аналогичный метод как `show..` (`showFrame`).

Как бы намекиньте этим, что существует тонкое различие между способами показа в этих методах, хотя на самом деле его нет.

По возможности, договоритесь с членами своей команды. Если Вася в своих классах использует `display..`, то Валера – обязательно `render..`, а Петя – `paint..`

...И напротив, если есть две функции с важными отличиями – используйте одно и то же слово для их описания! Например, с `print..` можно начать метод печати на принтере `printPage`, а также – метод добавления текста на страницу `printText`.

А теперь пусть читающий код думает: «Куда же выводит сообщение `printMessage?`». Особый шик – добавить элемент неожиданности. Пусть `printMessage` выводит не туда, куда все, а в новое окно!

Повторно используйте имена

*Когда целое разделено, его частям нужны имена.
Уже достаточно имён.
Нужно знать, когда остановиться.*

“ Лао-цзы

По возможности, повторно используйте имена переменных, функций и свойств. Просто записывайте в них новые значения.

Добавляйте новое имя, только если это абсолютно необходимо. В функции старайтесь обойтись только теми переменными, которые были переданы как параметры.

Это не только затруднит идентификацию того, что *сейчас* находится в переменной, но и сделает почти невозможным поиск места, в котором конкретное значение было присвоено.

Цель – развить интуицию и память читающего код программиста. Ну, а пока интуиция слаба, он может построчно анализировать код и конспектировать изменения переменных для каждой ветки исполнения.

Продвинутый вариант этого подхода – незаметно (!) подменить переменную на нечто похожее, например:

```
1 function ninjaFunction(elem) {
2   // 20 строк кода, работающего с elem
3
4   elem = clone(elem);
5
6   // ещё 20 строк кода, работающего с elem!
7 }
```

Программист, пожелавший добавить действия с `elem` во вторую часть функции, будет удивлён. Лишь во время отладки, посмотрев весь код, он с удивлением обнаружит, что, оказывается, имел дело с клоном!

Регулярные встречи с этим приёмом на практике говорят: защититься невозможно. Эффективно даже против опытного ниндзи.

Раздел

Качество кода

Навигация по уроку

Краткость – сестра таланта!

Однобуквенные переменные

Используйте сокращения

Будьте абстрактны при выборе имени.

Проверка внимания

Русские слова и сокращения

Хитрые синонимы

Повторно используйте имена

Добавляйте подчёркивания

Покажите вашу любовь к разработке

Перекрывайте внешние переменные

Внимание... Сюр-при-из!

Мощные функции!

Итого

Комментарии

Поделиться



Редактировать на GitHub

Раздел

Качество кода

Навигация по уроку

Краткость – сестра таланта!

Однобуквенные переменные

Используйте сокращения

Будьте абстрактны при выборе имени.

Проверка внимания

Русские слова и сокращения

Хитрые синонимы

Повторно используйте имена

Добавляйте подчёркивания

Покажите вашу любовь к разработке

Перекрывайте внешние переменные

Внимание... Сюр-при-из!

Мощные функции!

Итого

Комментарии

Поделиться



Редактировать на GitHub



Добавляйте подчёркивания

Добавляйте подчёркивания `_` и `__` к именам переменных. Например, `_name` или `__value`. Желательно, чтобы их смысл был известен только вам, а лучше – вообще без явной причины.

Этим вы достигните двух целей. Во-первых, код станет длиннее и менее читаемым, а во-вторых, другой программист будет долго искать смысл в подчёркиваниях. Особенно хорошо сработает и внесёт сумятицу в его мысли, если в некоторых частях проекта подчёркивания будут, а в некоторых – нет.

В процессе развития кода вы, скорее всего, будете путаться и смешивать стили: добавлять имена с подчёркиваниями там, где обычно подчёркиваний нет, и наоборот. Это нормально и полностью соответствует третьей цели – увеличить количество ошибок при внесении исправлений.

Покажите вашу любовь к разработке

Пусть все видят, какими замечательными сущностями вы оперируете! Имена `superElement`, `megaFrame` и `niceItem` при благоприятном положении звёзд могут привести к просветлению читающего.

Действительно, с одной стороны, кое-что написано: `super..`, `mega..`, `nice..`. С другой – это не несёт никакой конкретики. Читающий может решить поискать в этом глубокий смысл и замедлит работу на часок-другой оплаченного рабочего времени.

Перекрывайте внешние переменные

*Находясь на свету, нельзя ничего
увидеть в темноте.
Пребывая же в темноте, увидишь все,
что находится на свету.*

“ Гуань Инь-цзы

Почему бы не использовать одинаковые переменные внутри и снаружи функции? Это просто и не требует придумывать новых имён.

```
1 let user = authenticateUser();
2
3 function render() {
4   let user = anotherValue();
5   ...
6   ...многобукв...
7   ...
8   ... // <-- программист захочет внести исправления сюда
9   ...
10 }
```

Зашедший в середину метода `render` программист, скорее всего, не заметит, что переменная `user` локально перекрыта и попытается работать с ней, полагая, что это – результат `authenticateUser()` ... Ловушка захлопнулась! Здравствуй, отладчик.

Внимание... Сюр-при-из!

Есть функции, название которых говорит о том, что они ничего не меняют. Например, `isReady()`, `checkPermission()`, `findTags()` ... Предполагается, что при вызове они произведут некие вычисления или найдут и возвратят полезные данные, но при этом их не изменят. В трактатах это называется «отсутствие сторонних эффектов».

По-настоящему красивый приём – делать в таких функциях что-нибудь полезное, заодно с процессом проверки. Что именно – совершенно неважно.

Удивление и ошеломление, которое возникнет у вашего коллеги, когда он увидит, что функция с названием на `is..`, `check..` или `find...` что-то меняет – несомненно, расширит его границы разумного!

Раздел

Качество кода

Навигация по уроку

Краткость – сестра таланта!

Однобуквенные переменные

Используйте сокращения

Будьте абстрактны при выборе имени.

Проверка внимания

Русские слова и сокращения

Хитрые синонимы

Повторно используйте имена

Добавляйте подчёркивания

Покажите вашу любовь к разработке

Перекрывайте внешние переменные

Внимание... Сюр-при-из!

Мощные функции!

Итого

Комментарии

Поделиться



Редактировать на GitHub



Ещё одна вариация такого подхода – возвращать нестандартное значение.

Ведь общеизвестно, что `is...` и `check...` обычно возвращают `true/false`. Продemonстрируйте оригинальное мышление. Пусть вызов `checkPermission` возвращает не результат `true/false`, а объект с результатами проверки! А что, полезно.

Те же разработчики, кто попытается написать проверку `if (checkPermission(. .))`, будут весьма удивлены результатом. Ответьте им: «Надо читать документацию!». И перешлите эту статью.

Мощные функции!

*Дао везде и во всём,
и справа, и слева.*

“ Лао-цзы

Не ограничивайте действия функции тем, что написано в её названии. Будьте шире.

Например, функция `validateEmail(email)` может, кроме проверки e-mail на правильность, выводить сообщение об ошибке и просить заново ввести e-mail.

Выберите хотя бы пару дополнительных действий, кроме основного назначения функции. Главное – они должны быть неочевидны из названия функции. Истинный ниндзя-разработчик сделает так, что они будут неочевидны и из кода тоже.

Объединение нескольких смежных действий в одну функцию защитит ваш код от повторного использования.

Представьте, что другому разработчику нужно только проверить адрес, а сообщение – не выводить. Ваша функция `validateEmail(email)`, которая делает и то и другое, ему не подойдёт. И он не прервёт вашу медитацию вопросами о ней.

Итого

Все советы выше пришли из реального кода... И в том числе, от разработчиков с большим опытом. Возможно, даже больше вашего, так что не судите опрометчиво ;)

- Следуйте нескольким из них – и ваш код станет полон сюрпризов.
- Следуйте многим – и ваш код станет истинно вашим, никто не захочет изменять его.
- Следуйте всем – и ваш код станет ценным уроком для молодых разработчиков, ищущих просветления.

Проводим [курсы по JavaScript и фреймворкам](#). ✕

Комментарии

перед тем как писать...