


 → [Язык JavaScript](#)
 → [Качество кода](#)


 13 января 2020 г.

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.

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 поделать... 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 намекайте 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, просмотрев 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.

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 глубинный смысл и замедлитесь 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, и...
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 побочных 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!

Ещё одна вариация такого подхода – возвращать нестандартное значение.

Ведь общеизвестно, что `is...` и `check...` обычно возвращают `true/false`. Продемонстрируйте 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.



Предыдущий урок

Следующий урок



Поделиться



Карта учебника

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



Комментарии

- Если вам кажется, что в статье что-то не так - вместо комментария напишите [на GitHub](#).
- Для одной строки кода используйте тег `<code>` , для нескольких строк кода — тег `<pre>` , если больше 10 строк — ссылку на песочницу ([plnkr](#), [JSBin](#), [codepen...](#))
- Если что-то непонятно в статье — пишите, что именно и с какого места.

Присоединиться к обсуждению...

ВОЙТИ С ПОМОЩЬЮ

ИЛИ ЧЕРЕЗ DISQUS ?

Имя

♡ 97

Поделиться

Лучшие

Новые

Старые

И

Ирина

месяц назад

— 

Очень смешно)) Но смешно не будет разбирать подобный код. Интересно для чего эти ниндзя пишут так? Чтобы их код никто кроме их самих не понял? Какая-то цель особая или по другому не умеют?

0 0 Ответить • Поделиться ›

I

Indubie 612

→ Ирина

12 дней назад

— 

У самураев нет цели, только путь

1 0 Ответить • Поделиться ›

A

Anatoli 7777

месяц назад

— 

Великолепно получилось совместить иронию с изречениями мудрецов древности

3 0 Ответить • Поделиться ›



Андрей007

месяц назад

— 

Давно так не смеялся, спасибо)))))

5 0 Ответить • Поделиться ›

O

OnTheLimit!

месяц назад

— 

Помню опытного разраба, который давал имена value и data переменным

3 0 Ответить • Поделиться ›

V

Vladimir Degtyarev

2 месяца назад

— 

Это пять!))

3 0 Ответить • Поделиться ›

4 Ответить • Поделиться ›

**Андрей007**

→ Paul Pumphut

месяц назад

Открой форточку

1 0 Ответить • Поделиться ›

A**Артем Яковлев**

2 месяца назад

понял + принял, однако называть функции латиницей как-то банально, лучше всего использовать китайские иероглифы, что бы маслёнок помимо прочего расширял свой кругозор

5 0 Ответить • Поделиться ›

A**Александр Оленев**

3 месяца назад

Похоже что наболело)

2 0 Ответить • Поделиться ›

N**Nikolay**

3 месяца назад

А я вот не понимаю, зачем навязывать кому-то стиль именования переменных, функций. Если я не собираюсь работать в команде разработчиков, то может всё-таки я сам буду решать, какой стиль программирования выбрать. Если мне удобно, то я буду называть переменные по типу "a, b, c", или "sum1, sum2, sum3". Зачем в каждом руководстве пытаются навязать эти правила. Может я свободный художник, а не офисный раб-клерк, который не имеет ни капли воли и самостоятельности, ни своего мнения, а должен подчиняться какому-то там "коллективу" таких же. Что они могут? Фреймворк подключить, библиотеку и считать себя "первоклассными специалистами", которые настолько несамостоятельные, что даже пробелы и отступы в коде и то за них среда разработки расставляет.

Для меня программирование это в первую очередь творчество, которое нельзя загонять в какие-то рамки. И я хозяин своего кода. И мне решать, как мне будет лучше и удобнее. Да, какие несчастные эти "командные разработчики". Никогда им не познать весь кайф свободного художника. Несчастные люди.

И это не ирония.

показать больше

1 9 Ответить • Поделиться ›

M**Матвей Диркс**

→ Nikolay

2 месяца назад

Никогда не работай в команде, если тебе будет нужен проект из 500 строк

Николай, даже если ты и будешь работать один, если у тебя будет крупный проект на 500 строк файл, а таких файлов от 3 до 7, то ты будешь думать только том, как ты хочешь втащить себе и желательно посильнее, но если твой код это только

```
let a = 5
```

```
let b = 3
```

```
alert(a+b)
```

то все нормально можешь не запариваться

3

0 Ответить • Поделиться ›

V

Vladimir Gonchar

→ Nikolay

3 месяца назад

— 🚩

Ты через год вспомнишь, что ты там понаписал и что там шифруется в твоих переменных типа "shcd"?)

3

0 Ответить • Поделиться ›

J

Johan

→ Nikolay

3 месяца назад

— 🚩

Николай, ты можешь быть хозяином своего кода. И твой коллега может им быть и в целом, команды классных спецов может быть хозяевами своего кода. Вот только когда ты будешь ревьюить, тебе придется разбирать синтаксис "свободного художника". Энтропия проекта вырастит и всё, что останется ПМ это. Уволить вас за профнепригодность. Даже в опенсорсе за кривой код бьют по рукам. Стандарты для того и пишут, что бы понимать о чем идет речь.

Ну и главное, за твоей свободой скрывается твоя же клетка.

1

0 Ответить • Поделиться ›

H

HFq 3141

→ Nikolay

3 месяца назад

— 🚩

Статья шуточная и ничего никому не навязывает. Твоя задача-прочитать, если тебе хочется, либо пропустить и забыть. Это первое. А как второе - статья передает жизненный опыт программиста, который, вероятно, выполнял заказ, работая над сайтом. За заказ ему либо не заплатили, либо его кинули - фиг знает что случилось. Но сайт написан так, что никто, кроме него не разберется, что там происходит. Вот и для этих случаев нужно писать коряво настолько, насколько возможно. Ни один другой программист за такой код не возьмется, и работодатель будет вынужден оплатить работу и платить так же за поддержку сайта. Это просто пример

1

0 Ответить • Поделиться ›



Alexander Mandrov

3 месяца назад

— 🚩

какой-то очень толстый рофл...

Обстоятельства и желания сподвигли меня на создание собственного канала. Там я рассказываю о жизни и работе, о том, как стал программистом в 20 лет, коротко и по делу.

ТГ @unsleeping706

0

2 Ответить • Поделиться ›

J

jeison jei

4 месяца назад

— 🚩

Когда года 4 назад только начинал заниматься программированием всерьёз читал эту статью и думал, что так на самом деле нужно делать, чтобы другие не поняли мой код. Сейчас смеюсь

6 0 Ответить • Поделиться ›

М

Михаил Виноградов

4 месяца назад edited



⚡ Ну чё, народ, погна́ли? ⚡

Реально ли **изучить javascript за 7 месяцев** и трудоустроиться?

Вот я решил проверить и веду свой блог **Джаваскриптизёр** на ютубе, где буду выкладывать видео каждую неделю на протяжении 7 месяцев.

Если тебе тоже интересен джаваскрипт, присоединяйся:

ТГ: @javascriptzerr

Ютуб: Джаваскриптизёр

🔥 Видос уже на канале 🔥

🔥 Удачи всем нам 🔥

0 3 Ответить • Поделиться ›



J

John

4 месяца назад

За юмор лайк! Поржал))) Главное чтобы это новички не прочитали, для них это будет как инструкция к исполнению!

10 0 Ответить • Поделиться ›



Ярослав

5 месяцев назад



ахах, хочу получить статус ninja developer

11 0 Ответить • Поделиться ›

Загрузить ещё комментарии

Подписаться

Privacy

Не продавайте мои данные

