# Параметры JSHint

# <https://jshint.com/docs/options/>

Содержимое этой страницы берется из [JSHint репозиторий проектов](https://github.com/jshint/jshint). Если вы заметили ошибку, пожалуйста[, откройте проблему](https://github.com/jshint/jshint/issues/new)или (еще лучше) [сделать а тяни запрос](https://github.com/jshint/jshint/compare)!

### [Применение опций](https://jshint.com/docs/options/" \l "enforcing-options)

Если задано значение true, эти параметры заставят JSHint выдавать больше предупреждений о вашем коде.

|  |  |
| --- | --- |
| [побитовый](https://jshint.com/docs/options/#bitwise) | Этот параметр запрещает использование побитовых операторов, таких как ^(XOR), |(Или) и другие. Побитовые операторы очень редки в JavaScript программы и довольно часто &это просто опечатка &&. |
| [верблюжья шкура](https://jshint.com/docs/options/#camelcase) | **Предупреждение**этот параметр был объявлен устаревшим и будет сохранен удален в следующем крупном выпуске JSHint.  JSHint ограничивает свою область применения вопросами корректности кода. Если вы хотите применить правила, относящиеся к стилю кода, проверьте [АО проект](https://github.com/jscs-dev/node-jscs).  Этот параметр позволяет принудительно использовать все имена переменных либо стиль camelCase или UPPER\_CASE с подчеркиванием. |
| [кудрявый](https://jshint.com/docs/options/#curly) | Этот параметр требует, чтобы вы всегда ставили фигурные скобки вокруг блоков в петли и условные обозначения. JavaScript позволяет опустить фигурные скобки, когда блок состоит только из одного оператора, например:  while (day)  shuffle();  Однако в некоторых случаях это может привести к ошибкам (можно подумать, что sleep()является частью цикла в то время как на самом деле это не так):  while (day)  shuffle();  sleep(); |
| [enforceall](https://jshint.com/docs/options/#enforceall) | **Предупреждение**этот параметр был объявлен устаревшим и будет сохранен удален в следующем крупном выпуске JSHint.  Эта опция не может быть сохранена без автоматического выбора пользователей для новых функций. Это может привести к неожиданным последствиям предупреждения/ошибки при обновлении между второстепенными версиями Джшинт.  Эта опция является короткой рукой для самой строгой конфигурации JSHint как доступно в JSHint версии 2.6.3. Он включает все принудительные опции и отключает все расслабляющие опции, определенные в этом выпуске. |
| [eqeqeq](https://jshint.com/docs/options/#eqeqeq) | Этот вариант запрещает использование ==и !=в пользу ===а !==. Первые пытаются принудить значения, прежде чем сравнивать их, которые могут приведите к некоторым неожиданным результатам. Последние не делают никакого принуждения, так что они, как правило, безопаснее. Если вы хотите узнать больше о типе принуждение в JavaScript, мы рекомендуем [истину, равенство и JavaScript](http://javascriptweblog.wordpress.com/2011/02/07/truth-equality-and-javascript/) автор-Ангус Кролл. |
| [es3](https://jshint.com/docs/options/#es3) | **Предупреждение**этот параметр был объявлен устаревшим и будет сохранен удален в следующем крупном выпуске JSHint.  Используйте esversion: 3вместо этого.  Этот параметр сообщает JSHint, что ваш код должен соответствовать ECMAScript 3 спецификация. Используйте эту опцию, если вам нужно, чтобы ваша программа была исполняемой в старых браузерах-таких как Internet Explorer 6/7/8/9—и других устаревших Среды JavaScript. |
| [es5](https://jshint.com/docs/options/#es5) | **Предупреждение**этот параметр был объявлен устаревшим и будет сохранен удален в следующем крупном выпуске JSHint.  Используйте esversion: 5вместо этого.  Этот параметр включает синтаксис, впервые определенный в [ECMAScript 5.1 спецификация](http://es5.github.io/). Это включает в себя разрешение зарезервировать ключевые слова как свойства объекта. |
| [эсверсия](https://jshint.com/docs/options/#esversion) | Этот параметр используется для указания версии ECMAScript, к которой относится код должны придерживаться. Он может принимать одно из следующих значений:   * 3 - Если вам нужно, чтобы ваша программа была исполняемой в старых браузерах-таких как Internet Explorer 6/7/8/9—и других устаревших Среды JavaScript * 5 - Чтобы включить синтаксис, впервые определенный в [ECMAScript 5.1 спецификация](http://www.ecma-international.org/ecma-262/5.1/index.html). Это включает в себя разрешение зарезервированных ключевых слов в качестве свойств объекта. * 6 - Чтобы сообщить JSHint, что ваш код использует [ECMAScript 6](http://www.ecma-international.org/ecma-262/6.0/index.html)специфика синтаксис. Обратите внимание, что не все браузеры реализуют их. * 7 - Включить языковые функции, введенные [ECMAScript 7](https://www.ecma-international.org/ecma-262/7.0/index.html). Примечательный дополнения: оператор возведения в степень. * 8 - Включить языковые функции, введенные [ECMAScript 8](https://www.ecma-international.org/ecma-262/8.0/index.html). Примечательный дополнения: асинхронные функции, общая память и атомика * 9 - Включить языковые функции, введенные [ECMAScript 9](https://www.ecma-international.org/ecma-262/9.0/index.html). Примечательный дополнения: асинхронная итерация, свойства rest/spread и различные Расширения регулярных выражений * 10 - Включить языковые функции, введенные ECMAScript 10]( <https://www.ecma-international.org/ecma-262/10.0/index.html)>. Заметные дополнения: опциональные привязки защелки. |
| [forin](https://jshint.com/docs/options/#forin) | Этот параметр требует, чтобы все for inциклы фильтровали элементы объекта. То for in statement позволяет циклически перебирать имена всех свойства объекта, в том числе унаследованные через прототип цепь. Такое поведение может привести к неожиданным элементам в вашем объекте, поэтому оно как правило, безопаснее всегда отфильтровывать унаследованные свойства, как показано в образец:  for (key in obj) {  if (obj.hasOwnProperty(key)) {  // We are sure that obj[key] belongs to the object and was not inherited.  }  }  Для более глубокого понимания for inциклов в JavaScript прочтите [Изучение JavaScript for-in петли](http://javascriptweblog.wordpress.com/2011/01/04/exploring-javascript-for-in-loops/) Ангус Кролл. |
| [замерзать](https://jshint.com/docs/options/#freeze) | Эта опция запрещает перезапись прототипов собственных объектов, таких как ArrayDateи так далее.  // jshint freeze:true  Array.prototype.count = function (value) { return 4; };  // -> Warning: Extending prototype of native object: 'Array'. |
| [futurehostile](https://jshint.com/docs/options/#futurehostile) | Этот параметр включает предупреждения об использовании следующих идентификаторов: определено в будущих версиях JavaScript. Хотя переписывать их имеет отсутствие эффекта в контекстах, где они не реализуются, эта практика может вызывают проблемы при переносе базовых кодов на более новые версии языка. |
| [глобальный](https://jshint.com/docs/options/#globals) | Этот параметр можно использовать для указания белого списка глобальных переменных, которые формально они не определены в исходном коде. Это наиболее полезно, когда в сочетании с undefопцией для подавления предупреждений для глобальные переменные для конкретного проекта.  Установка записи в trueзначение позволяет выполнять чтение и запись в эту переменную. Установка этого falseпараметра вызовет JSHint, чтобы рассмотреть эту переменную только для чтения.  Смотрите также раздел Параметры "окружающая среда": набор параметров, которые будут использоваться как короткие рука для включения глобальных переменных, определенных в общем JavaScript окружающая среда.  Для настройки globalsв отдельном файле см. [раздел Inline Конфигурация](http://jshint.com/docs/#inline-configuration). |
| [внутримышечно](https://jshint.com/docs/options/#immed) | **Предупреждение**этот параметр был объявлен устаревшим и будет сохранен удален в следующем крупном выпуске JSHint.  JSHint ограничивает свою область применения вопросами корректности кода. Если вы хотите применить правила, относящиеся к стилю кода, проверьте [АО проект](https://github.com/jscs-dev/node-jscs).  Этот параметр запрещает использование немедленных вызовов функций без завернув их в скобки. Обертывание скобок помогает читателям ваш код в понимании того, что выражение является результатом функция, а не сама функция. |
| [выемка](https://jshint.com/docs/options/#indent) | **Предупреждение**этот параметр был объявлен устаревшим и будет сохранен удален в следующем крупном выпуске JSHint.  JSHint ограничивает свою область применения вопросами корректности кода. Если вы хотите применить правила, относящиеся к стилю кода, установите флажок вышел [проект](https://github.com/jscs-dev/node-jscs)АКС .  Этот параметр задает определенную ширину вкладки для вашего кода. |
| [латедеф](https://jshint.com/docs/options/#latedef) | Этот параметр запрещает использование переменной до ее определения. JavaScript имеет только область действия функции и, кроме того, все переменные всегда перемещаются—или поднимаются-в верхнюю часть функции. Такое поведение может привести к некоторым очень неприятным ошибкам, и именно поэтому это безопаснее всегда использовать переменная только после того, как они были явно определены.  Установка этого параметра в значение "nofunc" позволит выполнять объявления функций игнорируется.  Для более глубокого понимания области применения и подъема в JavaScript, чтение [области видимости JavaScript и Подъемник](http://www.adequatelygood.com/2010/2/JavaScript-Scoping-and-Hoisting) автор-Бен черри. |
| [leanswitch](https://jshint.com/docs/options/#leanswitch) | Этот параметр запрещает ненужные предложения в switchоператорах, напр.  switch (x) {  case 1:  default:  z();  }  Хотя подобные положения технически верны, они не действуют поведение программы может указывать и на ошибочный рефакторинг. |
| [максимальная сложность](https://jshint.com/docs/options/#maxcomplexity) | Этот параметр позволяет управлять цикломатической сложностью всего кода. Цикломатическая сложность измеряет число линейно независимых путей через исходный код программы. Подробнее о [цикломатической сложности читайте на сайте Википедия](http://en.wikipedia.org/wiki/Cyclomatic_complexity). |
| [maxdepth](https://jshint.com/docs/options/#maxdepth) | Этот параметр позволяет управлять тем, насколько вложенными должны быть ваши блоки:  // jshint maxdepth:2  function main(meaning) {  var day = true;  if (meaning === 42) {  while (day) {  shuffle();  if (tired) { // JSHint: Blocks are nested too deeply (3).  sleep();  }  }  }  } |
| [maxerr](https://jshint.com/docs/options/#maxerr) | Эта опция позволяет задать максимальное количество предупреждений, которое будет выдавать JSHint произведите перед сдачей. Значение по умолчанию-50. |
| [макслен](https://jshint.com/docs/options/#maxlen) | **Предупреждение**этот параметр был объявлен устаревшим и будет сохранен удален в следующем крупном выпуске JSHint.  JSHint ограничивает свою область применения вопросами корректности кода. Если вы хотите применить правила, относящиеся к стилю кода, установите флажок вышел [проект](https://github.com/jscs-dev/node-jscs)АКС .  Этот параметр позволяет задать максимальную длину строки. |
| [maxparams](https://jshint.com/docs/options/#maxparams) | Этот параметр позволяет задать максимальное количество формальных параметров, разрешенных для каждого пользователя. функция:  // jshint maxparams:3  function login(request, onSuccess) {  // ...  }  // JSHint: Too many parameters per function (4).  function logout(request, isManual, whereAmI, onSuccess) {  // ...  } |
| [maxstatements](https://jshint.com/docs/options/#maxstatements) | Этот параметр позволяет задать максимальное количество операторов, разрешенных для каждой функции:  // jshint maxstatements:4  function main() {  var i = 0;  var j = 0;  // Function declarations count as one statement. Their bodies  // don't get taken into account for the outer function.  function inner() {  var i2 = 1;  var j2 = 1;  return i2 + j2;  }  j = i + j;  return j; // JSHint: Too many statements per function. (5)  } |
| [newcap](https://jshint.com/docs/options/#newcap) | **Предупреждение**этот параметр был объявлен устаревшим и будет сохранен удален в следующем крупном выпуске JSHint.  JSHint ограничивает свою область применения вопросами корректности кода. Если вы хотите применить правила, относящиеся к стилю кода, проверьте [АО проект](https://github.com/jscs-dev/node-jscs).  Этот параметр требует прописных имен функций-конструкторов. Прописные функции, предназначенные для использования с newоператором это просто соглашение, которое помогает программистам визуально различать конструктор функций из других типов функций, чтобы помочь определить ошибки при использовании this.  Если вы этого не сделаете, ваш код не будет нарушен в любых браузерах или средах, но это будет немного сложнее выяснить, прочитав код, если предполагалось, что функция будет использоваться с новым или без него. А это уже ... важно, потому что когда функция, которая была предназначена для использования с newиспользуется без него, thisбудет указывать на глобальный объект вместо этого о новом объекте. |
| [noarg](https://jshint.com/docs/options/#noarg) | Этот параметр запрещает использование arguments.callerи arguments.callee. Оба .callerи .calleeделают совсем немного оптимизация невозможна, поэтому они были устаревшими в будущих версиях язык JavaScript. Фактически, ECMAScript 5 запрещает использованиеarguments.callee в строгом режиме. |
| [nocomma](https://jshint.com/docs/options/#nocomma) | Этот параметр запрещает использование оператора запятой. При неправильном использовании, то оператор запятой может скрывать значение оператора и продвигать его некорректный код. |
| [нет пустоты](https://jshint.com/docs/options/#noempty) | **Предупреждение**этот параметр был объявлен устаревшим и будет сохранен удален в следующем крупном выпуске JSHint.  JSHint ограничивает свою область применения вопросами корректности кода. Если вы хотите применить правила, относящиеся к стилю кода, проверите вне [JSCS проект](https://github.com/jscs-dev/node-jscs).  Этот параметр предупреждает о наличии пустого блока в коде. JSLint был первоначально предупреждение для всех пустых блоков, и мы просто сделали его необязательным. Не было никаких исследований сообщающих о том что пустые блоки в JavaScript ломаются ваш код в любом случае. |
| [nonbsp](https://jshint.com/docs/options/#nonbsp) | Этот параметр предупреждает о" неразрывных пробелах " символов. Эти символы могут быть введены с помощью опции-пробел на компьютерах Mac и иметь потенциал взлома веб-страниц, отличных от UTF8. |
| [nonew](https://jshint.com/docs/options/#nonew) | Этот параметр запрещает использование функций конструктора для побочных эффектов. Некоторые люди любят вызывать функции конструктора, не назначая его результат для любой переменной:  new MyConstructor();  Нет никакого преимущества в этом подходе по сравнению с простым вызовом MyConstructorтак как объект, который newсоздает оператор, не является используется везде, так что вы обычно должны избегать конструкторов, подобных этому. |
| [noreturnawait](https://jshint.com/docs/options/#noreturnawait) | Асинхронные функции разрешаются по их возвращаемому значению. В большинстве случаев это делает возвращая результат выражения ожидания (которое само по себе является обещанием пример) ненужный. Для ясности часто предпочтительнее вернуть результат асинхронной работы напрямую. Заметным исключением является в рамках tryпредложения о свидании-Подробнее см. раздел " Ожидание vs return vs return await":  <https://jakearchibald.com/2017/await-vs-return-vs-return-await/> |
| [predef](https://jshint.com/docs/options/#predef) | Этот параметр позволяет управлять тем, какие переменные JSHint считает необходимыми неявно определяется в среде. Настройте его с помощью массива строковое значение. Префиксация имени переменной с символом дефиса ( - ) будет удалите это имя из коллекции предопределенных переменных.  JSHint будет считать переменные, объявленные таким способом, доступными только для чтения.  Эта опция не может быть указана в строке; она может быть использована только через JavaScript API или из внешнего файла конфигурации. |
| [quotmark](https://jshint.com/docs/options/#quotmark) | **Предупреждение**этот параметр был объявлен устаревшим и будет сохранен удален в следующем крупном выпуске JSHint.  JSHint ограничивает свою область применения вопросами корректности кода. Если вы хотите применить правила, относящиеся к стилю кода, установите флажок вышел [проект](https://github.com/jscs-dev/node-jscs)АКС .  Этот параметр обеспечивает согласованность кавычек, используемых повсюду ваш код. Он принимает три значения: trueесли вы не хотите применять один определенный стиль, но нужна некоторая последовательность, "single"Если вы хотите разрешить только одинарные кавычки и "double"если вы хотите разрешить только двойные цитаты. |
| [regexpu](https://jshint.com/docs/options/#regexpu) | Этот параметр включает предупреждения для регулярных выражений, которые этого не делают включите флаг "u". Флаг " u " расширяет поддержку Unicode, а также включает более строгие правила синтаксического анализа. JSHint будет применять эти правила даже если он выполняется в движке JavaScript, который не поддерживает "u" Отметить. |
| [тень](https://jshint.com/docs/options/#shadow) | Этот параметр подавляет предупреждения о затенении переменных, т. е. объявление a переменная, которая уже была объявлена где-то во внешней области видимости.   * "inner" - проверяет наличие переменных, определенных только в одной области видимости * "outer" - проверяет также переменные, определенные во внешних областях * ложь-то же, что и внутренняя * true-разрешить переменное затенение |
| [одиночные группы](https://jshint.com/docs/options/#singleGroups) | Этот параметр запрещает использование оператора группировки, если он не используется строго требуется. Такое употребление обычно отражает непонимание например, унарные операторы:  // jshint singleGroups: true  delete(obj.attr); // Warning: Unnecessary grouping operator. |
| [строгий](https://jshint.com/docs/options/#strict) | Этот параметр требует, чтобы код выполнялся в строгом режиме ECMAScript 5. [Строгий режим](https://developer.mozilla.org/en/JavaScript/Strict_mode) это способ выбрать ограниченный вариант JavaScript. Строгий режим устраняет некоторые ловушки JavaScript, которые не вызывали ошибок, изменяя они производят ошибки. Он также исправляет ошибки, которые сделали его трудным чтобы движки JavaScript выполняли определенные оптимизации.   * "глобальный" - должна быть "use strict";директива на глобальном уровне * "подразумевается" - Линт код, как будто есть "use strict";директива * false-отключить предупреждения о строгом режиме * true-на уровне функций должна быть "use strict";директива; * this is preferable for scripts intended to be loaded in web * browsers directly because enabling strict mode globally * could adversely effect other scripts running on the same * page |
| [trailingcomma](https://jshint.com/docs/options/#trailingcomma) | Этот параметр предупреждает, когда запятая не помещается после последнего элемента в поле литерал массива или объекта. Из-за ошибок в старых версиях IE, трейлинг раньше запятые не поощрялись, но поскольку ES5 их семантика была стандартизированный. (Видеть [#11.1.4](http://www.ecma-international.org/ecma-262/5.1/#sec-11.1.4)и [#11.1.5](http://www.ecma-international.org/ecma-262/5.1/#sec-11.1.5).) Теперь они помогают предотвратить то же [самое визуальное неясности](http://www.ecma-international.org/ecma-262/5.1/#sec-7.9.2) что строгое использование точек с запятой помогает предотвратить.  Например, этот код мог сработать в прошлый вторник:  [  b + c  ].forEach(print);  Но если кто-то добавляет элемент в массив и забывает компенсировать отсутствует запятая, синтаксическая ошибка не возникает, и Линтер не может определить если это было ошибкой или преднамеренным вызовом функции.  [  b + c  (d + e)  ].forEach(print);  Если всегда добавлять элемент списка через запятую, эта двусмысленность невозможна происходить:  [  b + c,  ].forEach(print);  [  b + c,  (d + e),  ].forEach(print); |
| [undef](https://jshint.com/docs/options/#undef) | Этот параметр запрещает использование явно необъявленных переменных. Этот опция очень полезна для обнаружения утечек и опечаток переменных.  // jshint undef:true  function test() {  var myVar = 'Hello, World';  console.log(myvar); // Oops, typoed here. JSHint with undef will complain  }  Если ваша переменная определена в другом файле, вы можете использоватьglobal директива рассказать об этом Джшинту. |
| [неиспользуемый](https://jshint.com/docs/options/#unused) | Этот параметр предупреждает, когда вы определяете и никогда не используете свои переменные. Это очень полезно для общей очистки кода, особенно при использовании в дополнение к undef.  // jshint unused:true  function test(a, b) {  var c, d = 2;  return a + d;  }  test(1, 2);  // Line 3: 'b' was defined but never used.  // Line 4: 'c' was defined but never used.  Кроме того, этот параметр будет предупреждать вас о неиспользуемых глобальных переменные, объявленные с помощью globalдирективы.  Если установлено значение true, неиспользуемые параметры, за которыми следует используемый параметр не будет выдавать предупреждений. Этот параметр можно установить varsв значение to проверяйте только переменные, а не параметры функций или strictдля проверки все переменные и параметры. |
| [varstmt](https://jshint.com/docs/options/#varstmt) | Если установлено значение true, использование VariableStatements запрещено. Например:  // jshint varstmt: true  var a; // Warning: `var` declarations are forbidden. Use `let` or `const` instead. |

### [Расслабляющие варианты](https://jshint.com/docs/options/" \l "relaxing-options)

Если задано значение true, эти параметры заставят JSHint выдавать меньше предупреждений о вашем коде.

|  |  |
| --- | --- |
| [asi](https://jshint.com/docs/options/#asi) | Этот параметр подавляет предупреждения об отсутствии точек с запятой. Там есть много чего конечно, фуд около точки с запятой распространился довольно много людей в сообществе. Распространенными мифами являются то, что точка с запятой требуется все время (они являются нет) и что они ненадежны. JavaScript имеет правила о точках с запятой которые следуют все браузеры, так что это до вас, чтобы решить следует или не следует использовать точки с запятой в коде.  Для получения дополнительной информации о точках с запятой в JavaScript прочтите [открытое письмо к лидерам JavaScript по поводу Точки с запятой](http://blog.izs.me/post/2353458699/an-open-letter-to-javascript-leaders-regarding) автор Исаак Шлютер и [JavaScript точка с запятой Вставка](http://inimino.org/~inimino/blog/javascript_semicolons). |
| [Босс](https://jshint.com/docs/options/#boss) | Этот параметр подавляет предупреждения об использовании назначений в случаях где ожидаются сравнения. Чаще всего подобный код if (a = 10) {}является опечаткой. Однако это может быть полезно в таких случаях, как этот:  for (var i = 0, person; person = people[i]; i++) {}  Вы можете отключить эту ошибку на основе каждого использования, окружив назначение с круглыми скобками, такими как:  for (var i = 0, person; (person = people[i]); i++) {} |
| [отлаживать](https://jshint.com/docs/options/#debug) | Этот параметр подавляет предупреждения об debuggerоператорах в вашем компьютере. код. |
| [elision](https://jshint.com/docs/options/#elision) | Этот параметр сообщает JSHint, что в коде используются элементы elision массива ES3, или пустые элементы (например, [1, , , 4, , , 7]). |
| [eqnull](https://jshint.com/docs/options/#eqnull) | Этот параметр подавляет предупреждения о == nullсравнениях. Такой сравнение часто полезно, когда вы хотите проверить, является ли переменная nullили undefined... |
| [esnext](https://jshint.com/docs/options/#esnext) | **Предупреждение**этот параметр был объявлен устаревшим и будет сохранен удален в следующем крупном выпуске JSHint.  Используйте esversion: 6вместо этого.  Эта опция сообщает JSHint, что ваш код использует ECMAScript 6 specific синтаксис. Обратите внимание, что не все браузеры реализуют эти функции.  Подробная информация:   * [Спецификация для ECMAScript 6](http://www.ecma-international.org/ecma-262/6.0/index.html) |
| [зло](https://jshint.com/docs/options/#evil) | Этот параметр подавляет предупреждения об использовании eval. Использование evalне рекомендуется, потому что это может сделать ваш код уязвимым для различные инъекционные атаки и это делает его трудным для JavaScript интерпретатор для выполнения определенных оптимизаций. |
| [выражение](https://jshint.com/docs/options/#expr) | Этот параметр подавляет предупреждения об использовании выражений, где обычно вы ожидаете увидеть назначения или вызовы функций. Большинство время, такой код является опечаткой. Тем не менее, это не запрещено спецификацией и именно поэтому это предупреждение является необязательным. |
| [funcscope](https://jshint.com/docs/options/#funcscope) | Этот параметр подавляет предупреждения об объявлении переменных внутри структуры управления пока достигающ их более поздно от снаружи. Несмотря на то, что идентификаторы, объявленные с varимеют две реальные области—глобальные и функция-такая практика приводит к путанице среди людей новых к язык и трудно отлаживаемые ошибки. Именно поэтому, по умолчанию, JSHint предупреждает о переменных, которые используются за пределами их предполагаемой области действия.  function test() {  if (true) {  var x = 0;  }  x += 1; // Default: 'x' used out of scope.  // No warning when funcscope:true  } |
| [глобалстрикт](https://jshint.com/docs/options/#globalstrict) | **Предупреждение**этот параметр был объявлен устаревшим и будет сохранен удален в следующем крупном выпуске JSHint.  Использовать strict: "global".  Этот параметр подавляет предупреждения об использовании глобального строгого режима. Глобальный строгий режим может сломать сторонние виджеты, поэтому это не так Рекомендуемые.  Для получения дополнительной информации о строгом режиме см. strictпараметр. |
| [итератор](https://jshint.com/docs/options/#iterator) | Этот параметр подавляет предупреждения о \_\_iterator\_\_свойстве. Этот свойство не поддерживается всеми браузерами, поэтому используйте его осторожно. |
| [lastsemic](https://jshint.com/docs/options/#lastsemic) | Этот параметр подавляет предупреждения об отсутствии точек с запятой, но только когда точка с запятой опущена для последнего оператора в однострочном блоке:  var name = (function() { return 'Anton' }());  Это очень узкий случай использования, который полезен только при использовании автоматического Генераторы кода JavaScript. |
| [laxbreak](https://jshint.com/docs/options/#laxbreak) | **Предупреждение**этот параметр был объявлен устаревшим и будет сохранен удален в следующем крупном выпуске JSHint.  JSHint ограничивает свою область применения вопросами корректности кода. Если вы хотите применить правила, относящиеся к стилю кода, проверьте [АО проект](https://github.com/jscs-dev/node-jscs).  Этот параметр подавляет большинство предупреждений о возможной небезопасной линии взломы в вашем коде. Он не подавляет предупреждения о запятых-первых стиль кодирования. Чтобы подавить те, которые вы должны использовать laxcomma(см. ниже). |
| [лакскомма](https://jshint.com/docs/options/#laxcomma) | **Предупреждение**этот параметр был объявлен устаревшим и будет сохранен удален в следующем крупном выпуске JSHint.  JSHint ограничивает свою область применения вопросами корректности кода. Если вы хотите применить правила, относящиеся к стилю кода, проверьте [АО проект](https://github.com/jscs-dev/node-jscs).  Этот параметр подавляет предупреждения о стиле кодирования запятая-первая:  var obj = {  name: 'Anton'  , handle: 'valueof'  , role: 'SW Engineer'  }; |
| [loopfunc](https://jshint.com/docs/options/#loopfunc) | Этот параметр подавляет предупреждения о функциях внутри циклов. Определение функций внутри циклов может привести к таким ошибкам, как эта:  var nums = [];  for (var i = 0; i < 10; i++) {  nums[i] = function (j) {  return i + j;  };  }  nums[0](2); // Prints 12 instead of 2  Чтобы исправить приведенный выше код вам нужно скопировать значениеi:  var nums = [];  for (var i = 0; i < 10; i++) {  (function (i) {  nums[i] = function (j) {  return i + j;  };  }(i));  } |
| [МОЗ](https://jshint.com/docs/options/#moz) | Эта опция сообщает JSHint, что ваш код использует Mozilla JavaScript Расширения. Если только вы не разрабатываете специально для веб-браузера Firefox вам не нужен этот вариант.  Подробная информация:   * [Новое в JavaScript 1.7](https://developer.mozilla.org/en-US/docs/JavaScript/New_in_JavaScript/1.7) |
| [многополюсник](https://jshint.com/docs/options/#multistr) | **Предупреждение**этот параметр был объявлен устаревшим и будет сохранен удален в следующем крупном выпуске JSHint.  JSHint ограничивает свою область применения вопросами корректности кода. Если вы хотите применить правила, относящиеся к стилю кода, проверьте [АО проект](https://github.com/jscs-dev/node-jscs).  Этот параметр подавляет предупреждения о многострочных строках. Многострочный строки могут быть опасны в JavaScript, потому что весь ад вырвется на свободу, если вы случайно поставили пробел между escape-символом (\) и новая линия.  Обратите внимание, что даже если эта опция позволяет корректировать многострочные строки, она по-прежнему предупреждает о многострочных строках без escape-символов или с все, что находится между escape-символом и пробелом.  // jshint multistr:true  var text = "Hello\  World"; // All good.  text = "Hello  World"; // Warning, no escape character.  text = "Hello\  World"; // Warning, there is a space after \ |
| [notypeof](https://jshint.com/docs/options/#notypeof) | Этот параметр подавляет предупреждения о недопустимых typeofзначениях операторов. Этот оператор имеет только [ограниченный набор возможных возвратов ценности](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/typeof). По умолчанию JSHint предупреждает, когда вы сравниваете его результат с недопустимым значение, которое часто может быть опечаткой.  // 'fuction' instead of 'function'  if (typeof a == "fuction") { // Invalid typeof value 'fuction'  // ...  }  Не используйте эту опцию, если вы абсолютно уверены, что не хотите эта проверка. |
| [нойилд](https://jshint.com/docs/options/#noyield) | Этот параметр подавляет предупреждения о функциях генератора без yieldзаявление в них. |
| [ПлюсПлюс](https://jshint.com/docs/options/#plusplus) | Этот параметр запрещает использование унарного приращения и декремента операторы. Некоторые люди думают, что ++это и --снижает качество жизни. их стили кодирования и существуют языки программирования—такие как Python-это то, что идет совершенно без этих операторов. |
| [прото](https://jshint.com/docs/options/#proto) | Этот параметр подавляет предупреждения о \_\_proto\_\_свойстве. |
| [scripturl](https://jshint.com/docs/options/#scripturl) | Этот параметр подавляет предупреждения об использовании целевого сценария URL-адреса-например javascript:.... |
| [SUB](https://jshint.com/docs/options/#sub) | **Предупреждение**этот параметр был объявлен устаревшим и будет сохранен удален в следующем крупном выпуске JSHint.  JSHint ограничивает свою область применения вопросами корректности кода. Если вы хотите применить правила, относящиеся к стилю кода, проверьте [АО проект](https://github.com/jscs-dev/node-jscs).  Этот параметр подавляет предупреждения об использовании []нотации, когда это возможно. выражается в точечной нотации: person['name']vs person.name. |
| [сверхновая](https://jshint.com/docs/options/#supernew) | Этот параметр подавляет предупреждения о "странных" конструкциях, таких как new function () { ... }и new Object;... Такими конструкциями являются иногда используется для создания синглетов в JavaScript:  var singleton = new function() {  var privateVar;  this.publicMethod = function () {}  this.publicMethod2 = function () {}  }; |
| [действительное это](https://jshint.com/docs/options/#validthis) | Этот параметр подавляет предупреждения о возможных строгих нарушениях, когда код выполняется в строгом режиме, и вы используете thisв a функция, не являющаяся конструктором. Этот параметр следует использовать—в области действия только-когда вы уверены, что ваше использование thisдопустимо в строгий режим (например, если вы вызываете свою функцию с помощью Function.call).  **Примечание:**этот параметр можно использовать только внутри области действия функции. JSHint завершится ошибкой, если вы попытаетесь установить этот параметр глобально. |
| [withstmt](https://jshint.com/docs/options/#withstmt) | Этот параметр подавляет предупреждения об использовании withоператора. Семантика withвысказывания может вызвать путаницу среди разработчики и случайное определение глобальных переменных.  Подробная информация:   * [с учетом заявления Вредный](http://yuiblog.com/blog/2006/04/11/with-statement-considered-harmful/) |

### [Окружающая среда](https://jshint.com/docs/options/" \l "environments)

Эти параметры позволяют JSHint узнать о некоторых предопределенных глобальных переменных.

|  |  |
| --- | --- |
| [браузер](https://jshint.com/docs/options/#browser) | Этот параметр определяет глобалы, открытые современными браузерами: полностью от старые добрые documentи navigatorк HTML5 FileReaderи прочим новые разработки в мире браузеров.  **Примечание:**этот параметр не предоставляет такие переменные, как alertили console. develДополнительную информацию смотрите в разделе опция. |
| [browserify](https://jshint.com/docs/options/#browserify) | Этот параметр определяет глобалы, доступные при использовании [Browserify инструмент](http://browserify.org/)для построения проекта. |
| [диван](https://jshint.com/docs/options/#couch) | Этот параметр определяет глобалы, открытые с помощью [CouchDB](http://couchdb.apache.org/). CouchDB-это документ, ориентированный база данных, которая может быть запрошена и индексирована способом MapReduce с помощью язык JavaScript. |
| [девел](https://jshint.com/docs/options/#devel) | Этот параметр определяет глобалы, которые обычно используются для регистрации бедных людей. отладка: console,alert, etc. Обычно это хорошая идея, чтобы не делать этого отгружайте их в производство, потому что, например, console.logломается устаревшие версии Internet Explorer. |
| [додзе](https://jshint.com/docs/options/#dojo) | Этот параметр определяет глобалы, выставляемые [Додзе Инструментарий](http://dojotoolkit.org/). |
| [жасмин](https://jshint.com/docs/options/#jasmine) | Этот параметр определяет глобалы, открытые [модульным тестированием Jasmine рамки](https://jasmine.github.io/). |
| [jquery](https://jshint.com/docs/options/#jquery) | Этот параметр определяет глобалы, предоставляемые [jQuery](http://jquery.com/) библиотека JavaScript. |
| [мокко](https://jshint.com/docs/options/#mocha) | Этот параметр определяет глобалы, выставляемые UIs " BDD " и "TDD" [Mocha unit testing framework](http://mochajs.org/). |
| [модуль](https://jshint.com/docs/options/#module) | Этот параметр сообщает JSHint, что входной код описывает ECMAScript 6 модуль. Весь код модуля интерпретируется как код строгого режима. |
| [спорные вопросы](https://jshint.com/docs/options/#mootools) | Этот параметр определяет глобалы, предоставляемые [MooTools](http://mootools.net/) JavaScript framework. |
| [узел](https://jshint.com/docs/options/#node) | Этот параметр определяет глобалы, доступные при выполнении кода внутри среды выполнения узла. [Узел.js](http://nodejs.org/)- это a серверная среда JavaScript, использующая асинхронный событийная модель. Этот параметр также пропускает некоторые предупреждения, которые имеют смысл в среде браузера, но не имеют смысла в узле, таком как use strictпрагмы и console.logутверждения на уровне файлов. |
| [нестандартный](https://jshint.com/docs/options/#nonstandard) | Этот параметр определяет нестандартные, но широко распространенные глобалы, такие как escapeи unescapeеще . |
| [призрак](https://jshint.com/docs/options/#phantom) | Этот параметр определяет глобалы, доступные, когда ваше ядро работает внутри среды выполнения PhantomJS. [PhantomJS](http://phantomjs.org/) это безголовый webkit scriptable с JavaScript API. Он имеет быстрый и встроенная поддержка различных веб-стандартов: обработка DOM, селектор CSS, JSON, Canvas и SVG. |
| [prototypejs](https://jshint.com/docs/options/#prototypejs) | Этот параметр определяет глобалы, выставляемые [Прототип](http://www.prototypejs.org/)JavaScript-фреймворка. |
| [qunit](https://jshint.com/docs/options/#qunit) | Этот параметр определяет глобалы, открытые [модульным тестированием QUnit рамки](http://qunitjs.com/). |
| [носорог](https://jshint.com/docs/options/#rhino) | Этот параметр определяет глобалы, доступные при выполнении кода внутри среды выполнения Rhino. [Носорог](http://www.mozilla.org/rhino/) это реализация JavaScript с открытым исходным кодом, полностью написанная на Java. |
| [shelljs](https://jshint.com/docs/options/#shelljs) | Этот параметр определяет глобалы, выставляемые [ShellJS библиотека](http://documentup.com/arturadib/shelljs). |
| [набранный](https://jshint.com/docs/options/#typed) | Этот параметр определяет глобалы для типизированных конструкторов массивов.  Подробная информация:   * [Типизированный JavaScript массивы](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Typed_arrays) |
| [работник](https://jshint.com/docs/options/#worker) | Этот параметр определяет глобалы, доступные при выполнении кода внутри о веб-работнике. [Паутина Работники](https://developer.mozilla.org/en/Using_web_workers)обеспечивают а простые средства для веб-контента для запуска скриптов в фоновых потоках. |
| [wsh](https://jshint.com/docs/options/#wsh) | Этот параметр определяет глобалы, доступные, когда ваш код выполняется как a скрипт для скрипта [Windows Хозяин](http://en.wikipedia.org/wiki/Windows_Script_Host). |
| [Юи](https://jshint.com/docs/options/#yui) | Этот параметр определяет глобалы, предоставляемые [YUI](http://yuilibrary.com/) JavaScript framework. |

### [Нестабильный](https://jshint.com/docs/options/" \l "unstable)

Эти параметры включают поведение, которое может изменяться между основными выпусками JSHint. Соблюдайте осторожность при включении их в автоматизированных средах (например, непрерывная интеграция).

|  |  |
| --- | --- |
| [bigint](https://jshint.com/docs/options/#bigint) | [Предложение BigInt](https://github.com/tc39/proposal-bigint)расширяет Граммер языка для числовых литералов для поддержки целочисленных значений произвольная точность. Он также вводит новое значениеtypeof оператор, "бигинт".  Математические операции, использующие как BigInt, так и традиционный ECMAScript Числовые значения могут не иметь желаемого эффекта. Из-за слабо набранного текста природа языка, JSHint не в состоянии идентифицировать такие случаи. |
| [поставляемый.нестабильный = {](https://jshint.com/docs/options/#exports.unstable%20=%20{) | Нестабильные опции позволяют контролировать синтаксический анализ и линтинг предлагаемых дополнений к языку JavaScript. Так же как и языковые особенности, которые они описывают, наличие и поведение этих опций является изменчивым; JSHint резервирует право удалять или изменять их между основными выпусками версий. |