**Введение в JavaScript**

Давайте посмотрим, что такого особенного в JavaScript, почему именно он, и какие еще технологии существуют, кроме JavaScript.

**[Что такое JavaScript?](https://learn.javascript.ru/intro" \l "что-такое-javascript)**

*JavaScript* изначально создавался для того, чтобы сделать web-странички «живыми». Программы на этом языке называются *скриптами*. В браузере они подключаются напрямую к HTML и, как только загружается страничка – тут же выполняются.

**Программы на JavaScript – обычный текст**. Они не требуют какой-то специальной подготовки.

В этом плане JavaScript сильно отличается от другого языка, который называется [Java](http://ru.wikipedia.org/wiki/Java).

**Почему JavaScript?**

Когда создавался язык JavaScript, у него изначально было другое название: «LiveScript». Но тогда был очень популярен язык Java, и маркетологи решили, что схожее название сделает новый язык более популярным.

Планировалось, что JavaScript будет эдаким «младшим братом» Java. Однако, история распорядилась по-своему, JavaScript сильно вырос, и сейчас это совершенно независимый язык, со своей спецификацией, которая называется [ECMAScript](https://ru.wikipedia.org/wiki/ECMAScript), и к Java не имеет никакого отношения.

У него много особенностей, которые усложняют освоение, но по ходу учебника мы с ними разберёмся.

JavaScript может выполняться не только в браузере, а где угодно, нужна лишь специальная программа – [интерпретатор](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D1%80%D0%B5%D1%82%D0%B0%D1%82%D0%BE%D1%80). Процесс выполнения скрипта называют «интерпретацией».

**Компиляция и интерпретация, для программистов**

Для выполнения программ, не важно на каком языке, существуют два способа: «компиляция» и «интерпретация».

* *Компиляция* – это когда исходный код программы, при помощи специального инструмента, другой программы, которая называется «компилятор», преобразуется в другой язык, как правило – в машинный код. Этот машинный код затем распространяется и запускается. При этом исходный код программы остаётся у разработчика.
* *Интерпретация* – это когда исходный код программы получает другой инструмент, который называют «интерпретатор», и выполняет его «как есть». При этом распространяется именно сам исходный код (скрипт). Этот подход применяется в браузерах для JavaScript.

Современные интерпретаторы перед выполнением преобразуют JavaScript в машинный код или близко к нему, оптимизируют, а уже затем выполняют. И даже во время выполнения стараются оптимизировать. Поэтому JavaScript работает очень быстро.

Во все основные браузеры встроен интерпретатор JavaScript, именно поэтому они могут выполнять скрипты на странице. Но, разумеется, JavaScript можно использовать не только в браузере. Это полноценный язык, программы на котором можно запускать и на сервере, и даже в стиральной машинке, если в ней установлен соответствующий интерпретатор.

**Поговорим о браузерах**

Далее в этой главе мы говорим о возможностях и ограничениях JavaScript именно в контексте браузера.

**[Что умеет JavaScript?](https://learn.javascript.ru/intro" \l "что-умеет-javascript)**

Современный JavaScript – это «безопасный» язык программирования общего назначения. Он не предоставляет низкоуровневых средств работы с памятью, процессором, так как изначально был ориентирован на браузеры, в которых это не требуется.

Что же касается остальных возможностей – они зависят от окружения, в котором запущен JavaScript. В браузере JavaScript умеет делать всё, что относится к манипуляции со страницей, взаимодействию с посетителем и, в какой-то мере, с сервером:

* Создавать новые HTML-теги, удалять существующие, менять стили элементов, прятать, показывать элементы и т.п.
* Реагировать на действия посетителя, обрабатывать клики мыши, перемещения курсора, нажатия на клавиатуру и т.п.
* Посылать запросы на сервер и загружать данные без перезагрузки страницы (эта технология называется "AJAX").
* Получать и устанавливать cookie, запрашивать данные, выводить сообщения…
* …и многое, многое другое!

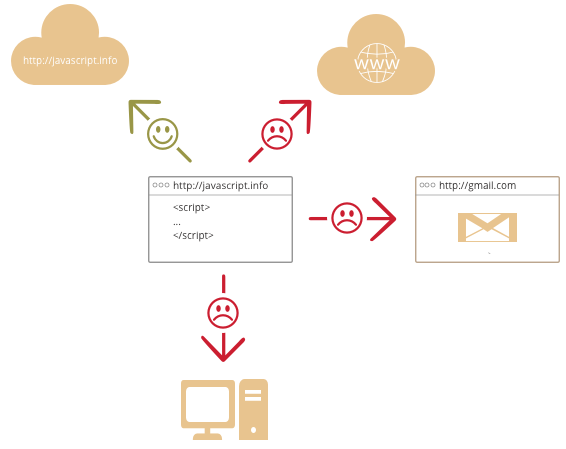
**[Что НЕ умеет JavaScript?](https://learn.javascript.ru/intro" \l "что-не-умеет-javascript)**

JavaScript – быстрый и мощный язык, но браузер накладывает на его исполнение некоторые ограничения…

Это сделано для безопасности пользователей, чтобы злоумышленник не мог с помощью JavaScript получить личные данные или как-то навредить компьютеру пользователя.

Этих ограничений нет там, где JavaScript используется вне браузера, например на сервере. Кроме того, современные браузеры предоставляют свои механизмы по установке плагинов и расширений, которые обладают расширенными возможностями, но требуют специальных действий по установке от пользователя

**Большинство возможностей JavaScript в браузере ограничено текущим окном и страницей.**



* JavaScript не может читать/записывать произвольные файлы на жесткий диск, копировать их или вызывать программы. Он не имеет прямого доступа к операционной системе.

Современные браузеры могут работать с файлами, но эта возможность ограничена специально выделенной директорией – *«песочницей»*. Возможности по доступу к устройствам также прорабатываются в современных стандартах и частично доступны в некоторых браузерах.

* JavaScript, работающий в одной вкладке, не может общаться с другими вкладками и окнами, за исключением случая, когда он сам открыл это окно или несколько вкладок из одного источника (одинаковый домен, порт, протокол).

Есть способы это обойти, и они раскрыты в учебнике, но они требуют специального кода на оба документа, которые находятся в разных вкладках или окнах. Без него, из соображений безопасности, залезть из одной вкладки в другую при помощи JavaScript нельзя.

* Из JavaScript можно легко посылать запросы на сервер, с которого пришла страница. Запрос на другой домен тоже возможен, но менее удобен, т. к. и здесь есть ограничения безопасности.

**[В чём уникальность JavaScript?](https://learn.javascript.ru/intro" \l "в-чём-уникальность-javascript)**

Есть как минимум *три* замечательных особенности JavaScript:

* Полная интеграция с HTML/CSS.
* Простые вещи делаются просто.
* Поддерживается всеми распространёнными браузерами и включён по умолчанию.

**Этих трёх вещей одновременно нет больше ни в одной браузерной технологии.**

Поэтому JavaScript и является самым распространённым средством создания браузерных интерфейсов.

**[Тенденции развития](https://learn.javascript.ru/intro" \l "тенденции-развития)**

Перед тем, как вы планируете изучить новую технологию, полезно ознакомиться с её развитием и перспективами. Здесь в JavaScript всё более чем хорошо.

**[HTML 5](https://learn.javascript.ru/intro" \l "html-5)**

*HTML 5* – эволюция стандарта HTML, добавляющая новые теги и, что более важно, ряд новых возможностей браузерам.

Вот несколько примеров:

* Чтение/запись файлов на диск (в специальной «песочнице», то есть не любые).
* Встроенная в браузер база данных, которая позволяет хранить данные на компьютере пользователя.
* Многозадачность с одновременным использованием нескольких ядер процессора.
* Проигрывание видео/аудио, без Flash.
* 2D и 3D-рисование с аппаратной поддержкой, как в современных играх.

Многие возможности HTML5 всё ещё в разработке, но браузеры постепенно начинают их поддерживать.

Тенденция: JavaScript становится всё более и более мощным и возможности браузера растут в сторону десктопных приложений.

**[ECMAScript 6](https://learn.javascript.ru/intro" \l "ecmascript-6)**

Сам язык JavaScript улучшается. Современный стандарт ECMAScript 5 включает в себя новые возможности для разработки, ECMAScript 6 будет шагом вперёд в улучшении синтаксиса языка.

Современные браузеры улучшают свои движки, чтобы увеличить скорость исполнения JavaScript, исправляют баги и стараются следовать стандартам.

Тенденция: JavaScript становится всё быстрее и стабильнее, в язык добавляются новые возможности.

Очень важно то, что новые стандарты HTML5 и ECMAScript сохраняют максимальную совместимость с предыдущими версиями. Это позволяет избежать неприятностей с уже существующими приложениями.

Впрочем, небольшая проблема с «супер-современными штучками» всё же есть. Иногда браузеры стараются включить новые возможности, которые ещё не полностью описаны в стандарте, но настолько интересны, что разработчики просто не могут ждать.

…Однако, со временем стандарт меняется и браузерам приходится подстраиваться к нему, что может привести к ошибкам в уже написанном, основанном на старой реализации, JavaScript-коде. Поэтому следует дважды подумать перед тем, как применять на практике такие «супер-новые» решения.

При этом все браузеры сходятся к стандарту, и различий между ними уже гораздо меньше, чем всего лишь несколько лет назад.

Тенденция: всё идет к полной совместимости со стандартом.

**[Альтернативные браузерные технологии](https://learn.javascript.ru/intro" \l "альтернативные-браузерные-технологии)**

Вместе с JavaScript на страницах используются и другие технологии. Связка с ними может помочь достигнуть более интересных результатов в тех местах, где браузерный JavaScript пока не столь хорош, как хотелось бы.

**[Java](https://learn.javascript.ru/intro" \l "java)**

Java – язык общего назначения, на нём можно писать самые разные программы. Для интернет-страниц есть особая возможность – написание *апплетов*.

*Апплет* – это программа на языке Java, которую можно подключить к HTML при помощи тега applet, выглядит это примерно так:

<applet code="BTApplet.class" codebase="/files/tutorial/intro/alt/">

<param name="nodes" value="50,30,70,20,40,60,80,35,65,75,85,90">

<param name="root" value="50">

</applet>

Такой тег загружает Java-программу из файла BTApplet.class и выполняет её с параметрами param. Апплет выполняется в отдельной части страницы, в прямоугольном «контейнере». Все действия пользователя внутри него обрабатывает апплет. Контейнер, впрочем, может быть и спрятан, если апплету нечего показывать.

Конечно, для этого на компьютере должна быть установлена и включена среда выполнения Java, включая браузерный плагин. Кроме того, апплет должен быть подписан сертификатом издателя (в примере выше апплет без подписи), иначе Java заблокирует его.

**Чем нам, JavaScript-разработчикам, может быть интересен Java?**

В первую очередь тем, что подписанный Java-апплет может всё то же, что и обычная программа, установленная на компьютере посетителя. Конечно, для этого понадобится согласие пользователя при открытии такого апплета.

**Достоинства**

* Java может делать *всё* от имени посетителя, совсем как установленная программа. Потенциально опасные действия требуют подписанного апплета и согласия пользователя.

**Недостатки**

* Java требует больше времени для загрузки.
* Среда выполнения Java, включая браузерный плагин, должна быть установлена на компьютере посетителя и включена.
* Java-апплет не интегрирован с HTML-страницей, а выполняется отдельно. Но он может вызывать функции JavaScript.

**[Плагины и расширения для браузера](https://learn.javascript.ru/intro" \l "плагины-и-расширения-для-браузера)**

Все современные браузеры предоставляют возможность написать плагины. Для этого можно использовать как JavaScript (Chrome, Opera, Firefox), так и язык С (ActiveX для Internet Explorer).

Эти плагины могут как отображать содержимое специального формата (плагин для проигрывания музыки, для показа PDF), так и взаимодействовать со страницей.

Как и в ситуации с Java-апплетом, у них широкие возможности, но посетитель поставит их в том случае, если вам доверяет.

**[Adobe Flash](https://learn.javascript.ru/intro" \l "adobe-flash)**

Adobe Flash – кросс-браузерная платформа для мультимедиа-приложений, анимаций, аудио и видео.

*Flash-ролик* – это скомпилированная программа, написанная на языке ActionScript. Её можно подключить к HTML-странице и запустить в прямоугольном контейнере.

В первую очередь Flash полезен тем, что позволяет **кросс-браузерно** работать с микрофоном, камерой, с буфером обмена, а также поддерживает продвинутые возможности по работе с сетевыми соединениями.

**Достоинства**

* Сокеты, UDP для P2P и другие продвинутые возможности по работе с сетевыми соединениями
* Поддержка мультимедиа: изображения, аудио, видео. Работа с веб-камерой и микрофоном.

**Недостатки**

* Flash должен быть установлен и включён. А на некоторых устройствах он вообще не поддерживается.
* Flash не интегрирован с HTML-страницей, а выполняется отдельно.
* Существуют ограничения безопасности, однако они немного другие, чем в JavaScript.

Из Flash можно вызывать JavaScript и наоборот, поэтому обычно сайты используют JavaScript, а там, где он не справляется – можно подумать о Flash.

**[Языки поверх JavaScript](https://learn.javascript.ru/intro" \l "языки-поверх-javascript)**

Синтаксис JavaScript устраивает не всех: одним он кажется слишком свободным, другим – наоборот, слишком ограниченным, третьи хотят добавить в язык дополнительные возможности, которых нет в стандарте…

Это нормально, ведь требования и проекты у всех разные.

В последние годы появилось много языков, которые добавляют различные возможности «поверх» JavaScript, а для запуска в браузере – при помощи специальных инструментов «трансляторов» превращаются в обычный JavaScript-код.

Это преобразование происходит автоматически и совершенно прозрачно, при этом неудобств в разработке и отладке практически нет.

При этом разные языки выглядят по-разному и добавляют совершенно разные вещи:

* Язык [CoffeeScript](http://coffeescript.org/) – это «синтаксический сахар» поверх JavaScript. Он сосредоточен на большей ясности и краткости кода. Как правило, его особенно любят программисты на Ruby.
* Язык [TypeScript](http://www.typescriptlang.org/) сосредоточен на добавлении строгой типизации данных. Он предназначен для упрощения разработки и поддержки больших систем. Его разрабатывает Microsoft.
* Язык [Dart](https://www.dartlang.org/) интересен тем, что он не только транслируется в JavaScript, как и другие языки, но и имеет свою независимую среду выполнения, которая даёт ему ряд возможностей и доступна для встраивания в приложения (вне браузера). Он разрабатывается компанией Google.

**ES6 и ES7 прямо сейчас**

Существуют также трансляторы, которые берут код, использующий возможности будущих стандартов JavaScript, и преобразуют его в более старый вариант, который понимают все браузеры.

Например, [babeljs](https://babeljs.io/).

Благодаря этому, мы можем использовать многие возможности будущего уже сегодня.

**[Итого](https://learn.javascript.ru/intro" \l "итого)**

Язык JavaScript уникален благодаря своей полной интеграции с HTML/CSS. Он работает почти у всех посетителей.

…Но хороший JavaScript-программист не должен забывать и о других технологиях.

Ведь наша цель – создание хороших приложений, и здесь Flash, Java и браузерные расширения имеют свои уникальные возможности, которые можно использовать вместе с JavaScript.

Что же касается CoffeeScript, TypeScript и других языков, построенных над JavaScript – они могут быть очень полезны. Рекомендуется посмотреть их, хотя бы в общих чертах, но, конечно, уже после освоения самого JavaScript.

**Справочники и спецификации**

В этом разделе мы познакомимся со справочниками и спецификациями.

Если вы только начинаете изучение, то вряд ли они будут нужны прямо сейчас. Тем не менее, эта глава находится в начале, так как предсказать точный момент, когда вы захотите заглянуть в справочник – невозможно, но точно известно, что этот момент настанет.

Поэтому рекомендуется кратко взглянуть на них и взять на заметку, чтобы при необходимости вернуться к ним в будущем.

**[Справочники, и как в них искать](https://learn.javascript.ru/pre-coding" \l "справочники-и-как-в-них-искать)**

Самая полная и подробная информация по JavaScript и браузерам есть в справочниках.

Её объём таков, что перевести всё с английского невозможно. Даже сделать «единый полный справочник» не получается, так как изменений много и они происходят постоянно.

Тем не менее, жить вполне можно если знать, куда смотреть.

**Есть три основных справочника по JavaScript на английском языке**:

1. [Mozilla Developer Network](https://developer.mozilla.org/) – содержит информацию, верную для основных браузеров. Также там присутствуют расширения только для Firefox (они помечены).

Когда мне нужно быстро найти «стандартную» информацию по RegExp – ввожу в Google **«RegExp MDN»**, и ключевое слово «MDN» (Mozilla Developer Network) приводит к информации из этого справочника.

1. [MSDN](http://msdn.microsoft.com/) – справочник от Microsoft. Там много информации, в том числе и по JavaScript (они называют его «JScript»). Если нужно что-то, специфичное для IE – лучше лезть сразу туда.

Например, для информации об особенностях RegExp в IE – полезное сочетание: **«RegExp msdn»**. Иногда к поисковой фразе лучше добавить термин «JScript»: **«RegExp msdn jscript»**.

1. [Safari Developer Library](https://developer.apple.com/library/safari/navigation/index.html) – менее известен и используется реже, но в нём тоже можно найти ценную информацию.

Есть ещё справочники, не от разработчиков браузеров, но тоже хорошие:

1. [http://help.dottoro.com](http://help.dottoro.com/) – содержит подробную информацию по HTML/CSS/JavaScript.
2. <http://javascript.ru/manual> – справочник по JavaScript на русском языке, он содержит основную информацию по языку, без функций для работы с документом. К нему можно обращаться и по адресу, если знаете, что искать. Например, так: <http://javascript.ru/RegExp>.
3. [http://www.quirksmode.org](http://www.quirksmode.org/) – информация о браузерных несовместимостях. Этот ресурс сам по себе довольно старый и, в первую очередь, полезен для поддержки устаревших браузеров. Для поиска можно пользоваться комбинацией **«quirksmode onkeypress»** в Google.
4. [http://caniuse.com](http://caniuse.com/) – ресурс о поддержке браузерами новейших возможностей HTML/CSS/JavaScript. Например, для поддержки функций криптографии: <http://caniuse.com/#feat=cryptography>.
5. <https://kangax.github.io/compat-table> – таблица с обзором поддержки спецификации ECMAScript различными платформами.

**[Спецификации](https://learn.javascript.ru/pre-coding" \l "спецификации)**

Спецификация – это самый главный, определяющий документ, в котором написано, как себя ведёт JavaScript, браузер, CSS и т.п.

Если что-то непонятно, и справочник не даёт ответ, то спецификация, как правило, раскрывает тему гораздо глубже и позволяет расставить точки над i.

**[Спецификация ECMAScript](https://learn.javascript.ru/pre-coding" \l "спецификация-ecmascript)**

Спецификация (формальное описание синтаксиса, базовых объектов и алгоритмов) языка Javascript называется [ECMAScript](http://www.ecma-international.org/publications/standards/Ecma-262.htm).

Её перевод есть на сайте в разделе [стандарт языка](http://es5.javascript.ru/).

**Почему не просто "JavaScript" ?**

Вы можете спросить: «Почему спецификация для JavaScript не называется просто *«JavaScript»*, зачем существует какое-то отдельное название?»

Всё потому, что JavaScript™ – зарегистрированная торговая марка, принадлежащая корпорации Oracle.

Название «ECMAScript» было выбрано, чтобы сохранить спецификацию независимой от владельцев торговой марки.

Спецификация может рассказать многое о том, как работает язык, и она является самым фундаментальным, доверенным источником информации.

**[Спецификации HTML/DOM/CSS](https://learn.javascript.ru/pre-coding" \l "спецификации-html-dom-css)**

JavaScript – язык общего назначения, поэтому в спецификации ECMAScript нет ни слова о браузерах.

Главная организация, которая занимается HTML, CSS, XML и множеством других стандартов – [Консорциум Всемирной паутины](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%81%D0%BE%D1%80%D1%86%D0%B8%D1%83%D0%BC_%D0%92%D1%81%D0%B5%D0%BC%D0%B8%D1%80%D0%BD%D0%BE%D0%B9_%D0%BF%D0%B0%D1%83%D1%82%D0%B8%D0%BD%D1%8B) (World Wide Consortium, сокращённо W3C).

Информацию о них можно найти на сайте [w3.org](http://w3.org/). К сожалению, найти в этой куче то, что нужно, может быть нелегко, особенно когда неизвестно в каком именно стандарте искать. Самый лучший способ – попросить Google с указанием сайта.

Например, для поиска document.cookie набрать [document.cookie site:w3.org](https://www.google.com/search?q=document.cookie+site%3Aw3.org).

Последние версии стандартов расположены на домене [dev.w3.org](http://dev.w3.org/).

Кроме того, в том, что касается HTML5 и DOM/CSS, W3C активно использует наработки другой организации – [WhatWG](https://whatwg.org/). Поэтому самые актуальные версии спецификаций по этим темам обычно находятся на <https://whatwg.org/specs/>.

Иногда бывает так, что информация на сайте [http://dev.w3.org](http://dev.w3.org/) отличается от [http://whatwg.org](http://whatwg.org/). В этом случае, как правило, следует руководствоваться [http://whatwg.org](http://whatwg.org/).

[**Итого**](https://learn.javascript.ru/pre-coding#итого)

Итак, посмотрим какие у нас есть источники информации.

Справочники:

* [Mozilla Developer Network](https://developer.mozilla.org/) – информация для Firefox и большинства браузеров. Google-комбо: "RegExp MDN", ключевое слово «MDN».
* [MSDN](http://msdn.microsoft.com/) – информация по IE. Google-комбо: "RegExp msdn". Иногда лучше добавить термин «JScript»: "RegExp msdn jscript".
* [Safari Developer Library](https://developer.apple.com/library/safari/navigation/index.html) – информация по Safari.
* [http://help.dottoro.com](http://help.dottoro.com/) – подробная информация по HTML/CSS/JavaScript с учётом браузерной совместимости. Google-комбо: "RegExp dottoro".
* <http://javascript.ru/manual> – справочник по JavaScript на русском языке. К нему можно обращаться и по адресу, если знаете, что искать. Например, так: <http://javascript.ru/RegExp>. Google-комбо: "RegExp site:javascript.ru".

Спецификации содержат важнейшую информацию о том, как оно «должно работать»:

* JavaScript, современный стандарт [ES5 (англ)](http://www.ecma-international.org/publications/standards/Ecma-262.htm), и предыдущий [ES3 (рус)](http://javascript.ru/ecma).
* HTML/DOM/CSS – на сайте [http://w3.org](http://www.w3.org/). Google-комбо: "document.cookie site:w3.org".
* …А самые последние версии стандартов – на [http://dev.w3.org](http://dev.w3.org/) и на [http://whatwg.org/specs/](https://whatwg.org/specs/).

То, как оно на самом деле работает и несовместимости:

* <http://quirksmode.org/>. Google-комбо: "innerHeight quirksmode".

Поддержка современных и новейших возможностей браузерами:

* [http://caniuse.com](http://caniuse.com/). Google-комбо: "caniuse geolocation".

**Редакторы для кода**

Для разработки обязательно нужен хороший редактор.

Выбранный вами редактор должен иметь в своем арсенале:

1. Подсветку синтаксиса.
2. Автодополнение.
3. «Фолдинг» (от англ. folding) – возможность скрыть-раскрыть блок кода.

**[IDE](https://learn.javascript.ru/editor" \l "ide)**

Термин IDE (Integrated Development Environment) – «интегрированная среда разработки», означает редактор, который расширен большим количеством «наворотов», умеет работать со вспомогательными системами, такими как багтрекер, контроль версий, и много чего ещё.

Как правило, IDE загружает весь проект целиком, поэтому может предоставлять автодополнение по функциям всего проекта, удобную навигацию по его файлам и т.п.

Если вы ещё не задумывались над выбором IDE, присмотритесь к следующим вариантам.

* Продукты IntelliJ: [WebStorm](http://www.jetbrains.com/webstorm/), а также в зависимости от дополнительного языка программирования [PHPStorm (PHP)](http://www.jetbrains.com/phpstorm/), [IDEA (Java)](http://www.jetbrains.com/idea/), [RubyMine (Ruby)](http://www.jetbrains.com/ruby/) и другие.
* Visual Studio, в сочетании с разработкой под .NET (Win)
* Продукты на основе Eclipse, в частности [Aptana](http://www.aptana.com/) и Zend Studio
* [Komodo IDE](http://www.activestate.com/komodo-ide) и его облегчённая версия [Komodo Edit](http://www.activestate.com/komodo-edit).
* [Netbeans](http://netbeans.org/)

Почти все они, за исключением Visual Studio, кросс-платформенные.

Сортировка в этом списке ничего не означает. Выбор осуществляется по вкусу и по другим технологиям, которые нужно использовать вместе с JavaScript.

Большинство IDE – платные, с возможностью скачать и бесплатно использовать некоторое время. Но их стоимость, по сравнению с зарплатой веб-разработчика, невелика, поэтому ориентироваться можно на удобство.

**[Лёгкие редакторы](https://learn.javascript.ru/editor" \l "лёгкие-редакторы)**

Лёгкие редакторы – не такие мощные, как IDE, но они быстрые и простые, мгновенно стартуют.

Основная сфера применения лёгкого редактора – мгновенно открыть нужный файл, чтобы что-то в нём поправить.

На практике «лёгкие» редакторы могут обладать большим количеством плагинов, так что граница между IDE и «лёгким» редактором размыта, спорить что именно редактор, а что IDE – не имеет смысла.

Достойны внимания:

* [Sublime Text](http://www.sublimetext.com/) (кросс-платформенный, shareware).
* [Atom](https://atom.io/) (кросс-платформенный, free).
* [SciTe](http://www.scintilla.org/) простой, легкий и очень быстрый (Windows, бесплатный).
* [Notepad++](http://sourceforge.net/projects/notepad-plus/) (Windows, бесплатный).
* Vim, Emacs. Если умеете их готовить.

**[Мои редакторы](https://learn.javascript.ru/editor" \l "мои-редакторы)**

Лично мои любимые редакторы:

* Как IDE – редакторы от Jetbrains: для чистого JavaScript [WebStorm](http://www.jetbrains.com/webstorm/), если ещё какой-то язык, то в зависимости от языка: [PHPStorm (PHP)](http://www.jetbrains.com/phpstorm/), [IDEA (Java)](http://www.jetbrains.com/idea/), [RubyMine (Ruby)](http://www.jetbrains.com/ruby/). У них есть и другие редакторы под разные языки, но я ими не пользовался.
* Как быстрый редактор – [Sublime Text](http://www.sublimetext.com/).
* Иногда Visual Studio, если разработка идёт под платформу .NET (Win).

Если не знаете, что выбрать – можно посмотреть на них ;)

**[Не будем ссориться](https://learn.javascript.ru/editor" \l "не-будем-ссориться)**

В списках выше перечислены редакторы, которые использую я или мои знакомые – хорошие разработчики. Конечно, существуют и другие отличные редакторы, если вам что-то нравится – пользуйтесь.

Выбор редактора, как и любого инструмента, во многом индивидуален и зависит от ваших проектов, привычек, личных предпочтений.

**Консоль разработчика**

При разработке скриптов всегда возможны ошибки… Впрочем, что я говорю? У вас абсолютно точно будут ошибки, если конечно вы – человек, а не [робот](http://ru.wikipedia.org/wiki/%D0%91%D0%B5%D0%BD%D0%B4%D0%B5%D1%80_(%D0%A4%D1%83%D1%82%D1%83%D1%80%D0%B0%D0%BC%D0%B0)).

Чтобы читать их в удобном виде, а также получать массу полезной информации о выполнении скриптов, в браузерах есть *инструменты разработки*.

**Для разработки рекомендуется использовать Chrome или Firefox.**

Другие браузеры, как правило, находятся в положении «догоняющих» по возможностям встроенных инструментов разработки. Если ошибка, к примеру, именно в Internet Explorer, тогда уже смотрим конкретно в нём, но обычно – Chrome/Firefox.

В инструментах разработчика предусмотрена масса возможностей, но на текущем этапе мы просто посмотрим, как их открывать, смотреть в консоли ошибки и запускать команды JavaScript.

**[Google Chrome](https://learn.javascript.ru/devtools" \l "google-chrome)**

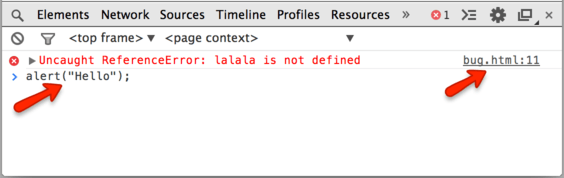
Откройте страницу [bug.html](https://learn.javascript.ru/article/devtools/bug.html).

В её JavaScript-коде есть ошибка. Конечно, обычному посетителю она не видна, нужно открыть инструменты разработчика.

Для этого используйте клавишу F12 под Windows, а если у вас Mac, то Cmd+Opt+J.

При этом откроются инструменты разработчика и вкладка Console, в которой будет ошибка.

Выглядеть будет примерно так:



* При клике на bug.html вы перейдёте во вкладку с кодом к месту ошибки, там будет и краткое описание ошибки. В данном случае ошибка вызвана строкой lalala, которая интерпретатору непонятна.
* В этом же окошке вы можете набирать команды на JavaScript. Например, наберите alert("Hello") – команду вывода сообщения и запустите её нажатием Enter. Мы познакомимся с этой и другими командами далее.
* Для перевода курсора на следующую строку (если команда состоит из нескольких строк) – используется сочетание Shift+Enter.

Далее в учебнике мы подробнее рассмотрим отладку в Chrome в главе [Отладка в браузере Chrome](https://learn.javascript.ru/debugging-chrome).

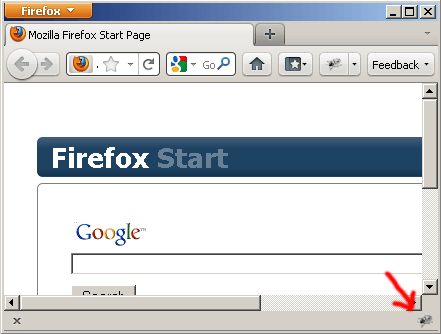
**[Firefox](https://learn.javascript.ru/devtools" \l "firefox)**

Для разработки в Firefox используется расширение Firebug.

1. Первым делом его надо установить.

Это можно сделать со страницы <https://addons.mozilla.org/ru/firefox/addon/firebug/>.

Перезапустите браузер. Firebug появится в правом-нижнем углу окна:

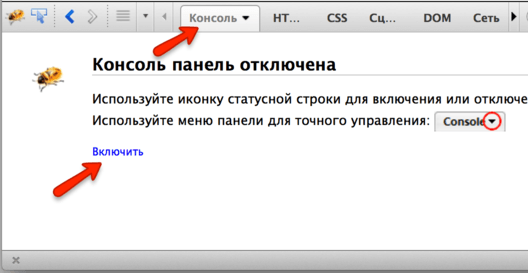


Если иконки не видно – возможно, у вас выключена панель расширений. Нажмите Ctrl+\ для её отображения.

Ну а если её нет и там, то нажмите F12 – это горячая клавиша для запуска Firebug, расширение появится, если установлено.

1. Далее, для того чтобы консоль заработала, её надо включить.

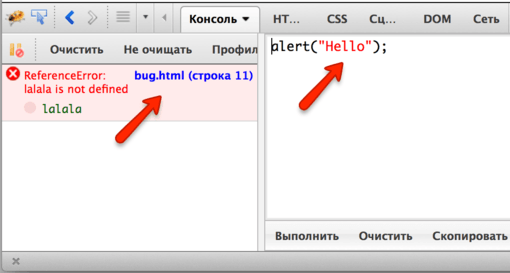
Если консоль уже была включена ранее, то этот шаг не нужен, но если она серая – выберите в меню Консоль и включите её:



1. Для того, чтобы Firebug работал без глюков, желательно сначала открыть Firebug, а уже потом – зайти на страницу.

С открытым Firebug зайдите на страницу с ошибкой: [bug.html](https://learn.javascript.ru/devtools/bug.html).

Консоль покажет ошибку:



Кликните на строчке с ошибкой и браузер покажет исходный код. При необходимости включайте дополнительные панели.

Как и в Chrome, можно набирать и запускать команды. Область для команд на рисунке находится справа, запуск команд осуществляется нажатием Ctrl+Enter (для Mac – Cmd+Enter).

Можно перенести её вниз, нажав на кнопочку https://learn.javascript.ru/article/devtools/firefox_console_down.png (на рисунке её не видно, но она присутствует в правом нижнем углу панели разработки).

Об основных возможностях можно прочитать на сайте [firebug.ru](http://firebug.ru/).

**[Internet Explorer](https://learn.javascript.ru/devtools" \l "internet-explorer)**

Панель разработчика запускается нажатием F12.

Откройте её и зайдите на страницу с ошибкой: [bug.html](https://learn.javascript.ru/devtools/bug.html). Если вы разобрались с Chrome/Firefox, то дальнейшее будет вам более-менее понятно, так как инструменты IE построены позже и по аналогии с Chrome/Firefox.

**[Safari](https://learn.javascript.ru/devtools" \l "safari)**

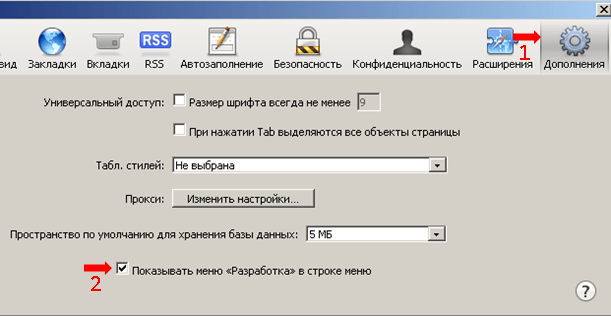
Горячие клавиши: Ctrl+Shift+I, Ctrl+Alt+C для Mac – Cmd вместо Ctrl.

Для доступа к функционалу разработки через меню:

1. В Safari первым делом нужно активировать меню разработки.

Откройте меню, нажав на колесико справа-сверху и выберите Настройки.

Затем вкладка Дополнительно:



Отметьте Показывать меню "Разработка" в строке меню. Закройте настройки.

1. Нажмите на колесико и выберите Показать строку меню.

Инструменты будут доступны в появившейся строке меню, в пункте Разработка.

[**Итого**](https://learn.javascript.ru/devtools#итого)

Мы разобрали, как открывать инструменты разработчика и смотреть ошибки, а также запускать простые команды, не отходя от браузера.

Далее мы приступим к изучению JavaScript.