## Введение в JavaScript

**JavaScript** – это *интерпретируемый язык* программирования с объектно-ориентированными возможностями.   
**Интерпретируемый язык программирования** — язык программирования, в котором исходный код программы не преобразовывается в машинный код для непосредственного выполнения центральным процессором (как в компилируемых языках), а исполняется с помощью специальной программы-интерпретатора.

С точки зрения синтаксиса базовый язык JavaScript напоминает C, C++ и Java такими программными конструкциями, как инструкция if, цикл while и оператор &&. Однако это подобие ограничивается синтаксической схожестью.

JavaScript – это язык c *динамической типизацией*, т. е. в нем не требуется определять типы переменных. Объекты в JavaScript отображают имена свойств на произвольные значения.

Ядро языка JavaScript поддерживает работу с такими простыми типами данных, как числа, строки и булевы значения. Помимо этого, он обладает встроенной поддержкой массивов, дат и объектов регулярных выражений.

Обычно JavaScript применяется в веб-браузерах, а расширение его возможностей за счет введения объектов позволяет организовать взаимодействие с пользователем, управлять веб-браузером и изменять содержимое документа, отображаемое в пределах окна веб-браузера. Эта встроенная версия JavaScript запускает сценарии, внедренные в HTML-код веб-страниц. Как правило, эта версия называется *клиентским* языком JavaScript, чтобы подчеркнуть, что сценарий исполняется на клиентском компьютере, а не на веб-сервере.

В основе языка JavaScript и поддерживаемых им типов данных лежат международные стандарты, благодаря чему обеспечивается прекрасная совместимость между реализациями. Некоторые части клиентского JavaScript формально стандартизированы, другие части стали стандартом де-факто, но есть части, которые являются специфическими расширениями конкретной версии браузера.

Совместимость реализаций JavaScript в разных браузерах зачастую приносит немало беспокойств программистам, использующим клиентский язык JavaScript.

### Что такое JavaScript

Вокруг JavaScript довольно много дезинформации и путаницы. Прежде чем двигаться дальше в изучении JavaScript, важно развенчать некоторые распространенные мифы, связанные с этим языком.

JavaScript – это не Java

Одно из наиболее распространенных заблуждений о JavaScript состоит в том, что этот язык представляет собой упрощенную версию Java, языка программирования, разработанного в компании Sun Microsystems. Кроме некоторой синтаксической схожести и способности предоставлять исполняемое содержимое для веб-браузеров, эти два языка между собой ничто не связывает.

Схожесть имен – не более чем уловка маркетологов (первоначальное название языка – LiveScript – было изменено на JavaScript в последнюю минуту). Однако JavaScript и Java могут взаимодействовать друг с другом.

JavaScript не простой язык

Поскольку JavaScript является интерпретируемым языком, очень часто он позиционируется как язык сценариев, а не как язык программирования, при этом подразумевается, что языки сценариев проще и в большей степени ориентированы не на программистов, а на обычных пользователей. В самом деле, при отсутствии контроля типов JavaScript прощает многие ошибки, которые допускают неопытные программисты. Благодаря этому многие веб-дизайнеры могут использовать JavaScript для решения ограниченного круга задач, выполняемых по точным рецептам.

Однако за внешней простотой JavaScript скрывается полноценный язык программирования, столь же сложный, как любой другой, и даже более сложный, чем многие. Программисты, пытающиеся решать с помощью JavaScript нетривиальные задачи, часто разочаровываются в процессе разработки из-за того, что недостаточно понимают возможности этого языка.

### История JavaScript

#### Предпосылки

В [1992 году](https://ru.wikipedia.org/wiki/1992_%D0%B3%D0%BE%D0%B4) компания Nombas начала разработку встраиваемого скриптового языка Cmm (Си-минус-минус), который, по замыслу разработчиков, должен был стать достаточно мощным, чтобы заменить макросы, сохраняя при этом схожесть с Си, чтобы разработчикам не составляло труда изучить его. Главным отличием от Си была работа с памятью. В новом языке всё управление памятью осуществлялось автоматически: не было необходимости создавать [буферы](https://ru.wikipedia.org/wiki/%D0%91%D1%83%D1%84%D0%B5%D1%80_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), объявлять переменные, осуществлять преобразование типов. В остальном языки сильно походили друг на друга: в частности, Cmm поддерживал стандартные функции и операторы Си.

Cmm был переименован в ScriptEase, поскольку исходное название звучало слишком негативно, а упоминание в нём Си «отпугивало» людей. На основе этого языка был создан [проприетарный продукт](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%BF%D1%80%D0%B8%D0%B5%D1%82%D0%B0%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5" \o "Проприетарное программное обеспечение) CEnvi. В конце ноября [1995 года](https://ru.wikipedia.org/wiki/1995_%D0%B3%D0%BE%D0%B4) Nombas разработала версию CEnvi, внедряемую в веб-страницы. Страницы, которые можно было изменять с помощью скриптового языка, получили название Espresso Pages — они демонстрировали использование скриптового языка для создания игры, проверки пользовательского ввода в формы и создания анимации. Espresso Pages позиционировались как [демоверсия](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BC%D0%BE%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F), призванная помочь представить, что случится, если в браузер будет внедрён язык Cmm. Работали они только в 16-битовом [Netscape Navigator](https://ru.wikipedia.org/wiki/Netscape_Navigator" \o "Netscape Navigator) под управлением Windows.

#### JavaScript

Перед *[Бренданом Эйхом](https://ru.wikipedia.org/wiki/%D0%AD%D0%B9%D1%85,_%D0%91%D1%80%D0%B5%D0%BD%D0%B4%D0%B0%D0%BD" \o "Эйх, Брендан)*, нанятым в компанию *[Netscape](https://ru.wikipedia.org/wiki/Netscape_Communications" \o "Netscape Communications)* 4 апреля [1995 года](https://ru.wikipedia.org/wiki/1995_%D0%B3%D0%BE%D0%B4), была поставлена задача внедрить язык программирования [Scheme](https://ru.wikipedia.org/wiki/Scheme) или что-то похожее в браузер Netscape. Поскольку требования были размыты, Эйха перевели в группу, ответственную за серверные продукты, где он проработал месяц, занимаясь улучшением протокола [HTTP](https://ru.wikipedia.org/wiki/HTTP). В мае разработчик был переброшен обратно, в команду, занимающуюся клиентской частью (браузером), где он немедленно начал разрабатывать концепцию нового языка программирования.

Менеджмент разработки браузера, включая Тома Пакина, [Михаэля Тоя](https://ru.wikipedia.org/w/index.php?title=%D0%A2%D0%BE%D0%B9,_%D0%9C%D0%B8%D1%85%D0%B0%D1%8D%D0%BB%D1%8C&action=edit&redlink=1) , Рика Шелла, был убеждён, что Netscape должен поддерживать язык программирования, встраиваемый в [HTML](https://ru.wikipedia.org/wiki/HTML)-код страницы.

Помимо Брендана Эйха в разработке участвовали сооснователь [Netscape Communications](https://ru.wikipedia.org/wiki/Netscape_Communications" \o "Netscape Communications) [Марк Андрессен](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B4%D1%80%D0%B5%D1%81%D1%81%D0%B5%D0%BD,_%D0%9C%D0%B0%D1%80%D0%BA) и сооснователь [Sun Microsystems](https://ru.wikipedia.org/wiki/Sun_Microsystems" \o "Sun Microsystems) [Билл Джой](https://ru.wikipedia.org/wiki/%D0%94%D0%B6%D0%BE%D0%B9,_%D0%91%D0%B8%D0%BB%D0%BB): чтобы успеть закончить работы над языком к релизу браузера, компании заключили соглашение о сотрудничестве в разработке. Они ставили перед собой цель обеспечить «язык для склеивания» составляющих частей веб-ресурса: изображений, плагинов, Java-апплетов, который был бы удобен для веб-дизайнеров и программистов, не обладающих высокой квалификацией.

Первоначально язык назывался *Mocha*, затем он был переименован в *LiveScript* и предназначался как для программирования на стороне клиента, так и для программирования на стороне сервера (там он должен был называться LiveWire). На синтаксис оказали влияние языки [Си](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)) и [Java](https://ru.wikipedia.org/wiki/Java" \o "Java), и, поскольку Java в то время было [модным словом](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%BD%D1%8B%D0%B5_%D1%81%D0%BB%D0%BE%D0%B2%D0%B0), 4 декабря 1995 года LiveScript переименовали в JavaScript, получив соответствующую лицензию у [Sun](https://ru.wikipedia.org/wiki/Sun_Microsystems" \o "Sun Microsystems). Анонс JavaScript со стороны представителей Netscape и Sun состоялся накануне выпуска второй [бета-версии](https://ru.wikipedia.org/wiki/%D0%91%D0%B5%D1%82%D0%B0-%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) Netscape Navigator. В нём декларируется, что 28 лидирующих ИТ-компаний выразили намерение использовать в своих будущих продуктах JavaScript как объектный скриптовый язык с открытым стандартом.

В [1996 году](https://ru.wikipedia.org/wiki/1996_%D0%B3%D0%BE%D0%B4) компания [Microsoft](https://ru.wikipedia.org/wiki/Microsoft" \o "Microsoft) выпустила аналог языка JavaScript, названный [JScript](https://ru.wikipedia.org/wiki/Microsoft_JScript" \o "Microsoft JScript). Анонсирован этот язык был 18 июля 1996 года. Первым браузером, поддерживающим эту реализацию, был [Internet Explorer](https://ru.wikipedia.org/wiki/Internet_Explorer" \o "Internet Explorer) 3.0.

По инициативе компании Netscape была проведена стандартизация языка ассоциацией [ECMA](https://ru.wikipedia.org/wiki/ECMA_International). Стандартизированная версия имеет название [ECMAScript](https://ru.wikipedia.org/wiki/ECMAScript" \o "ECMAScript), описывается стандартом [ECMA-262](https://ru.wikipedia.org/wiki/ECMA-262). Первой версии спецификации соответствовал JavaScript версии 1.1, а также языки JScript и [ScriptEasy](https://ru.wikipedia.org/w/index.php?title=ScriptEasy&action=edit&redlink=1" \o "ScriptEasy (страница отсутствует)).

### Популярность

В статье «Самый неправильно понятый язык программирования в мире стал самым популярным в мире языком программирования» [Дуглас Крокфорд](https://ru.wikipedia.org/w/index.php?title=%D0%9A%D1%80%D0%BE%D0%BA%D1%84%D0%BE%D1%80%D0%B4,_%D0%94%D1%83%D0%B3%D0%BB%D0%B0%D1%81&action=edit&redlink=1)  утверждает, что лидирующую позицию JavaScript занял в связи с развитием [AJAX](https://ru.wikipedia.org/wiki/AJAX), поскольку браузер стал превалирующей системой доставки приложений.

Он также констатирует растущую популярность JavaScript, то, что этот язык встраивается в приложения, отмечает значимость языка.

Согласно [TIOBE Index](https://ru.wikipedia.org/wiki/TIOBE_Index), базирующемуся на данных поисковых систем [Google](https://ru.wikipedia.org/wiki/Google" \o "Google), [MSN](https://ru.wikipedia.org/wiki/MSN), [Yahoo!](https://ru.wikipedia.org/wiki/Yahoo!" \o "Yahoo!), [Википедия](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%BA%D0%B8%D0%BF%D0%B5%D0%B4%D0%B8%D1%8F) и [YouTube](https://ru.wikipedia.org/wiki/YouTube" \o "YouTube), в апреле 2015 года JavaScript находился на 6 месте (год назад на 9).

(Индекс TIOBE (TIOBE programming community index) — индекс, оценивающий популярность [языков программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), на основе подсчета результатов поисковых запросов, содержащих название языка).

JavaScript является самым популярным языком программирования, используемым для разработки веб-приложений [на стороне клиента](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D1%82%D0%BE%D1%80%D0%BE%D0%BD%D0%B0_%D0%BA%D0%BB%D0%B8%D0%B5%D0%BD%D1%82%D0%B0&action=edit&redlink=1) ([англ.](https://en.wikipedia.org/wiki/Client-side))

**Дополнительные возможности JavaScript**

*Библиотеки*

Для обеспечения высокого [уровня абстракции](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%BE%D0%B9_%D0%B0%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%B3%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) и достижения приемлемой степени [кросс-браузерности](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81-%D0%B1%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80%D0%BD%D0%BE%D1%81%D1%82%D1%8C) при разработке веб-приложений используются библиотеки JavaScript. Они представляют собой набор многократно используемых объектов и функций.

Среди известных JavaScript библиотек можно отметить [Adobe life](https://ru.wikipedia.org/w/index.php?title=Adobe_life&action=edit&redlink=1" \o "Adobe life (страница отсутствует)), [Dojo](https://ru.wikipedia.org/wiki/Dojo) Toolkit, [Extjs](https://ru.wikipedia.org/wiki/Extjs), [jQuery](https://ru.wikipedia.org/wiki/JQuery), [Mootools](https://ru.wikipedia.org/wiki/Mootools), [Prototype](https://ru.wikipedia.org/wiki/Prototype_(%D1%84%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA)), [Qooxdoo](https://ru.wikipedia.org/w/index.php?title=Qooxdoo&action=edit&redlink=1), [Underscore](https://ru.wikipedia.org/wiki/Underscore).

*Фреймворки*

Интересными возможностями JavaScript также являются различные фреймворки, библиотеки и утилиты, построенные на его базе. Например, MV\* фреймворки Angular.js, Backbone.js, Ember.js, React.js, Ext.js. Фреймворки – это такие библиотеки, которые позволяют уйти от спагетти-кода, они помогают поддерживать структуру и организованность в проекте, облегчают ремонтопригодность в будущем. Есть даже фреймворк Matreshka.js. Все эти библиотеки очень помогают при работе с веб-приложениями и прочими.

<https://angularjs.org/>

<http://backbonejs.org/>

<http://emberjs.com/>

<https://facebook.github.io/react/>

<https://www.sencha.com/products/extjs/#overview>

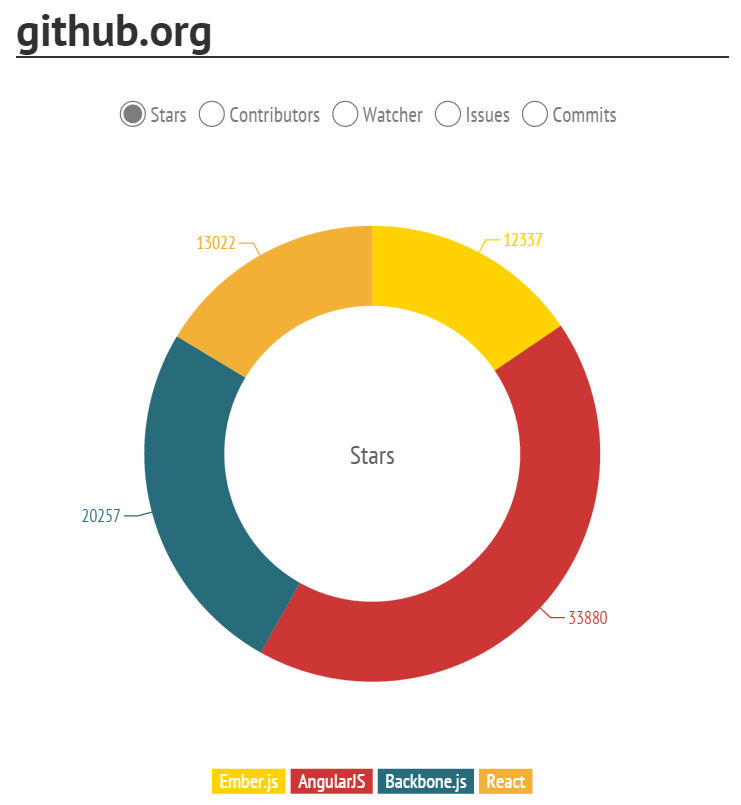
<http://matreshka.io/>

*Популярность фреймворков*

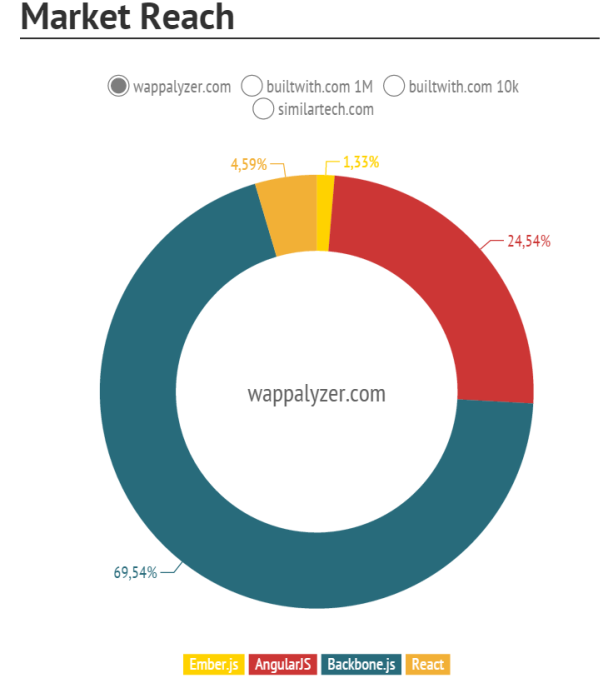
Интерес к Javascript MV\* фреймворкам вызвал их подъем.

[Meteor](http://www.meteor.com/), [Ember](http://emberjs.com/), [Angular](http://angularjs.org/),  [Backbone](http://www.backbonejs.org/), все они действительно популярны на Github. Измерить популярность довольно сложно, но хорошим показателем может быть количество Github-звездочек.

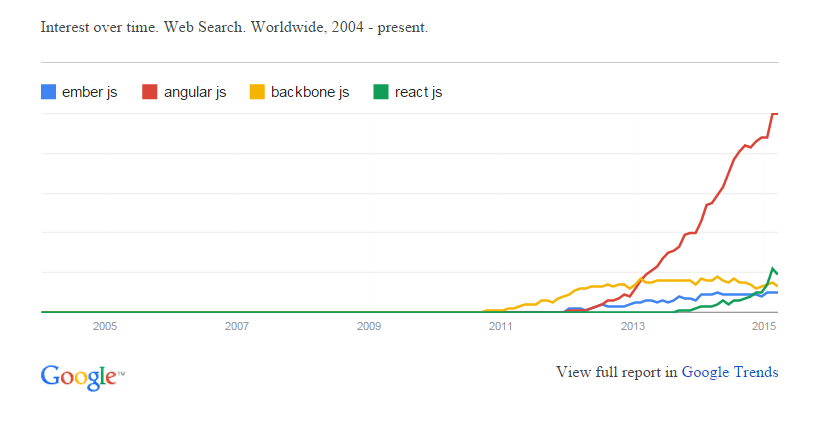
Звездочки на Github-e:



Зрелость проектов:



Google Trends:



*Jswiki*

Jswiki — это проект на github, который постарался собрать все достойные JavaScript библиотеки и ресурсы. На страницах описания библиотек, так же собраны ссылки на статьи для начинающих, чтобы читатель мог как можно быстрее начать использовать ту или иную библиотеку.

<https://github.com/bebraw/jswiki/wiki>

**Область применения**

**Веб-приложения**

JavaScript используется в клиентской части веб-приложений: клиент-серверных программ, в котором клиентом является браузер, а сервером — веб-сервер, имеющих распределённую между сервером и клиентом логику. Обмен информацией в веб-приложениях происходит по сети. Одним из преимуществ такого подхода является тот факт, что клиенты не зависят от конкретной операционной системы пользователя, поэтому веб-приложения являются кроссплатформенными сервисами.

**AJAX**

JavaScript используется в [AJAX](https://ru.wikipedia.org/wiki/AJAX), популярном подходе к построению интерактивных пользовательских интерфейсов веб-приложений, заключающемся в «фоновом» асинхронном обмене данными браузера с веб-сервером. В результате, при обновлении данных веб-страница не перезагружается полностью и интерфейс веб-приложения становится быстрее, чем это происходит при традиционном подходе (без применения AJAX).

**Comet**

Comet — широкое понятие, описывающее механизм работы веб-приложений, использующих постоянные HTTP-соединения, что позволяет веб-серверу отправлять данные браузеру без дополнительного запроса со стороны браузера. Для таких приложений используются технологии, непосредственно поддерживаемые браузерами. В частности, в них широко используется JavaScript.

**Браузерные операционные системы**

JavaScript широко используется в [браузерных операционных системах](https://ru.wikipedia.org/wiki/WebOS" \o "WebOS). Так, например, исходный код [IndraDesktop WebOS](https://ru.wikipedia.org/w/index.php?title=IndraDesktop_WebOS&action=edit&redlink=1" \o "IndraDesktop WebOS (страница отсутствует)) на 75 % состоит из JavaScript, код браузерной операционной системы [IntOS](https://ru.wikipedia.org/w/index.php?title=IntOS&action=edit&redlink=1" \o "IntOS (страница отсутствует)) — на 70 %. Доля JavaScript в исходном коде [eyeOS](https://ru.wikipedia.org/wiki/EyeOS" \o "EyeOS) — 5 %, однако и в рамках этой операционной системы JavaScript играет важную роль, участвуя в визуализации на клиенте и являясь необходимым механизмом для коммуницирования клиента и сервера.

**Букмарклеты**

JavaScript используется для создания небольших программ, размещаемых в закладки браузера. При этом используются URL-адреса со спецификатором javascript:.

**Пользовательские скрипты в браузере**

Пользовательские скрипты в браузере — это программы, написанные на JavaScript, выполняемые в браузере пользователя при загрузке страницы. Они позволяют автоматически заполнять формы, переформатировать страницы, скрывать нежелательное содержимое и встраивать желательное для отображения содержимое, изменять поведение клиентской части веб-приложений, добавлять элементы управления на страницу и т. д.

Для управления пользовательскими скриптами в [Mozilla Firefox](https://ru.wikipedia.org/wiki/Mozilla_Firefox" \o "Mozilla Firefox) используется расширение [Greasemonkey](https://ru.wikipedia.org/wiki/Greasemonkey" \o "Greasemonkey); [Opera](https://ru.wikipedia.org/wiki/Opera" \o "Opera) и [Google Chrome](https://ru.wikipedia.org/wiki/Google_Chrome" \o "Google Chrome) предоставляют средства поддержки пользовательских скриптов и возможности для выполнения ряда скриптов Greasemonkey.

**Серверные приложения**

Приложения, написанные на JavaScript, могут исполняться на серверах, использующих [Java](https://ru.wikipedia.org/wiki/Java" \o "Java) 6 и более поздних версий. Это обстоятельство используется для построения серверных приложений, позволяющих обрабатывать JavaScript на стороне сервера.

Помимо Java 6, существует ряд платформ, использующих существующие движки (интерпретаторы) JavaScript для исполнения серверных приложений. (Как правило, речь идёт о повторном использовании движков, ранее созданных для исполнения кода JavaScript в браузерах WWW.)

|  |  |  |  |
| --- | --- | --- | --- |
| Платформы исполнения серверных приложений на JavaScript | | | |
| Название | Используемый движок JavaScript | Языки, на которых написан движок и платформа | Лицензия |
| Jaxer[[60]](https://ru.wikipedia.org/wiki/JavaScript" \l "cite_note-72) | [SpiderMonkey](https://ru.wikipedia.org/wiki/SpiderMonkey)[[61]](https://ru.wikipedia.org/wiki/JavaScript#cite_note-73) | C++, C[[62]](https://ru.wikipedia.org/wiki/JavaScript#cite_note-74) | [GPL 3](https://ru.wikipedia.org/wiki/GNU_General_Public_License#GPL_v3)[[63]](https://ru.wikipedia.org/wiki/JavaScript#cite_note-75) |
| persevere-framework[[64]](https://ru.wikipedia.org/wiki/JavaScript" \l "cite_note-76) | [Rhino](https://ru.wikipedia.org/wiki/Rhino) | Java[[65]](https://ru.wikipedia.org/wiki/JavaScript" \l "cite_note-77) | [Модифицированная лицензия BSD](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D1%8F_BSD)[[66]](https://ru.wikipedia.org/wiki/JavaScript#cite_note-78) |
| Helma[[67]](https://ru.wikipedia.org/wiki/JavaScript" \l "cite_note-79) | Rhino | Java, JavaScript[[68]](https://ru.wikipedia.org/wiki/JavaScript" \l "cite_note-80) | [BSD-подобная](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D1%8F_BSD#.D0.9B.D0.B8.D1.86.D0.B5.D0.BD.D0.B7.D0.B8.D0.B8_.D1.82.D0.B8.D0.BF.D0.B0_BSD) Helma License 2.0[[69]](https://ru.wikipedia.org/wiki/JavaScript#cite_note-81) |
| v8cgi | [V8](https://ru.wikipedia.org/wiki/V8_(%D0%B4%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA_JavaScript)) | C++, JavaScript[[70]](https://ru.wikipedia.org/wiki/JavaScript" \l "cite_note-82) | [Лицензия BSD](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D1%8F_BSD)[[71]](https://ru.wikipedia.org/wiki/JavaScript#cite_note-83) |
| node.js | V8 | C++[[72]](https://ru.wikipedia.org/wiki/JavaScript#cite_note-84) | [Лицензия MIT](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D1%8F_MIT)[[73]](https://ru.wikipedia.org/wiki/JavaScript#cite_note-85) |

JavaScript на стороне сервера используется в проектах [Google](https://ru.wikipedia.org/wiki/Google_(%D0%BA%D0%BE%D0%BC%D0%BF%D0%B0%D0%BD%D0%B8%D1%8F)" \o "Google (компания)). Так например, [Google Sites](https://ru.wikipedia.org/wiki/Google_Sites" \o "Google Sites) допускает подстройку с помощью JavaScript-сценариев, исполняемых движком Rhino.

Можно привести пример, когда JavaScript на стороне сервера удобнее использовать, чем тот же язык Java. Возьмем для рассмотрения Node.js.

Node.js обладает полезными свойствами, такими как поддержка большого числа соединений и быстрая их обработка, в отличие от Java.  
Также Node.js работает в разы быстрее.

Интеграция с клиентом осуществляется гораздо проще: один и тот же язык на обеих сторонах, передача JSON не составляет никакого труда, дополнительные утилиты не требуются.

Запросы для некоторых относительно новых баз данных, например, MongoDB,  CouchDB пишутся на языке JavaScript.

**Мобильные приложения**

Перевод мобильных устройств [Palm](https://ru.wikipedia.org/wiki/Palm_(%D0%9A%D0%9F%D0%9A)" \o "Palm (КПК)) на использование [Palm webOS](https://ru.wikipedia.org/wiki/Palm_webOS" \o "Palm webOS) в качестве операционной системы с [Mojo SDK](https://ru.wikipedia.org/w/index.php?title=Mojo_SDK&action=edit&redlink=1" \o "Mojo SDK (страница отсутствует)) в качестве [комплекта средств разработки](https://ru.wikipedia.org/wiki/SDK) позволяет использовать JavaScript в качестве языка разработки мобильных приложений.

Native script (NS) – это библиотека, позволяющая делать кросс-платформенные приложения, используя XML, CSS, JavaScript. Native script решает ту же задачу, что и уже всем известный phonegap (создание кросс-платформенных приложений), но подходы у них разные. Phonegap использует движок браузера, чтобы отобразить ваш UI (фактически вы получаете веб-страницу), Native script использует нативный рендеринг, использует элементы нативного UI. Следующее важное отличие: чтобы получить доступ к камере, gps и так далее в phonegap необходимо устанавливать плагины, в то время как NS дает доступ из коробки.  
Стоит подчеркнуть, что приложения можно писать для Android 4.2 и выше, и для iOS 7.1 и выше.

Для этих целей также можно использовать библиотеку React Native.

<https://www.nativescript.org/>

<http://phonegap.com/>

<http://www.reactnative.com/>

**Виджеты**

Виджет — вспомогательная мини-программа, графический модуль которой размещается в рабочем пространстве соответствующей [родительской программы](https://ru.wikipedia.org/w/index.php?title=%D0%94%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA_%D0%B2%D0%B8%D0%B4%D0%B6%D0%B5%D1%82%D0%BE%D0%B2&action=edit&redlink=1), служащая для украшения рабочего пространства, развлечения, решения отдельных рабочих задач или быстрого получения информации из интернета без помощи веб-браузера.

JavaScript используется как для реализации виджетов, так и для реализации движков виджетов. В частности, при помощи JavaScript реализованы [Apple Dashboard](https://ru.wikipedia.org/wiki/Apple_Dashboard" \o "Apple Dashboard), [Microsoft Gadgets](https://ru.wikipedia.org/w/index.php?title=Microsoft_Gadgets&action=edit&redlink=1" \o "Microsoft Gadgets (страница отсутствует)), [Yahoo! Widgets](https://ru.wikipedia.org/w/index.php?title=Yahoo!_Widgets&action=edit&redlink=1" \o "Yahoo! Widgets (страница отсутствует)), [Google Gadgets](https://ru.wikipedia.org/w/index.php?title=Google_Gadgets&action=edit&redlink=1" \o "Google Gadgets (страница отсутствует)), [Klipfolio Dashboard](https://ru.wikipedia.org/w/index.php?title=Klipfolio_Dashboard&action=edit&redlink=1" \o "Klipfolio Dashboard (страница отсутствует)).

**Прикладное программное обеспечение**

JavaScript используется для написания [прикладного ПО](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5). Например, 16,4 % исходного кода [Mozilla Firefox](https://ru.wikipedia.org/wiki/Mozilla_Firefox" \o "Mozilla Firefox)написано на JavaScript.

[Google Chrome OS](https://ru.wikipedia.org/wiki/Google_Chrome_OS) в качестве прикладного ПО использует [веб-приложения](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5).

**Манипуляция объектами приложений**

JavaScript также находит применение в качестве скриптового языка доступа к объектам приложений. Платформа Mozilla ([XUL](https://ru.wikipedia.org/wiki/XUL)/[Gecko](https://ru.wikipedia.org/wiki/Gecko_(%D0%B4%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA)" \o "Gecko (движок))) использует JavaScript. Среди сторонних продуктов, например, [Java](https://ru.wikipedia.org/wiki/Sun_Java_Runtime_Environment" \o "Sun Java Runtime Environment), начиная с версии 6, содержит встроенный интерпретатор JavaScript на базе [Rhino](https://ru.wikipedia.org/wiki/Rhino" \o "Rhino). Сценарии JavaScript поддерживаются в таких приложениях Adobe, как [Adobe Photoshop](https://ru.wikipedia.org/wiki/Adobe_Photoshop" \o "Adobe Photoshop), [Adobe Dreamweaver](https://ru.wikipedia.org/wiki/Adobe_Dreamweaver" \o "Adobe Dreamweaver),[Adobe Illustrator](https://ru.wikipedia.org/wiki/Adobe_Illustrator) и [Adobe InDesign](https://ru.wikipedia.org/wiki/Adobe_InDesign" \o "Adobe InDesign).

**Офисные приложения**

JavaScript используется в [офисных приложениях](https://ru.wikipedia.org/wiki/%D0%9E%D1%84%D0%B8%D1%81%D0%BD%D1%8B%D0%B9_%D0%BF%D0%B0%D0%BA%D0%B5%D1%82) для автоматизации рутинных действий, написания [макросов](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D0%BA%D1%80%D0%BE%D1%81), организации доступа со стороны веб-служб.

**Microsoft Office**

В [Excel Services](https://ru.wikipedia.org/wiki/SharePoint) 2010 добавились два новых интерфейса программирования приложений: [REST](https://ru.wikipedia.org/wiki/REST) API и JavaScript Object Model ([JSOM](https://ru.wikipedia.org/w/index.php?title=JSOM&action=edit&redlink=1)).

* Excel Services 2010 REST API позволяет осуществлять доступ к объектам рабочих книг, таким как таблицы, диаграммы и именованные серии данных; получать изображения, HTML, [Atom](https://ru.wikipedia.org/wiki/Atom" \o "Atom), рабочие книги; устанавливать значения и обновлять вычисления перед запрашиванием элементов.
* JSOM даёт возможность реагировать на действия пользователя в отношении Excel Web Access ([EWA](https://ru.wikipedia.org/w/index.php?title=EWA&action=edit&redlink=1)), программно взаимодействовать с составляющими EWA. Использование JSOM осуществляется при помощи помещения кода JavaScript на страницу, содержащую компоненты EWA.

**OpenOffice.org**

JavaScript — один из языков программирования, используемых для написания макросов в приложениях, входящих в состав OpenOffice.org. В OpenOffice.org интегрирован интерпретатор JavaScript Rhino. По состоянию на декабрь 2009 года поддержка JavaScript носила ограниченный характер. Ограничения, присущие разработке макросов OpenOffice.org на JavaScript:

* среда выполнения JavaScript поддерживает загрузку лишь тех классов Java, которые развёрнуты сценарием JavaScript;
* среда выполнения JavaScript не предоставляет сообщения об ошибках, произошедших во время выполнения скрипта;
* ещё не реализована поддержка интерактивной разработки JavaScript-сценариев.

В OpenOffice.org имеется редактор и отладчик JavaScript-сценариев.

**Обучение информатике**

JavaScript обладает [пропедевтической](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%BF%D0%B5%D0%B4%D0%B5%D0%B2%D1%82%D0%B8%D0%BA%D0%B0) ценностью, позволяя сочетать при обучении [информатике](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0) интенсивную практику программирования и широту используемых технологий. Преподавание данного языка в школе позволяет создать базу для изучения [веб-программирования](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), использовать на уроках творческие проекты. Соответствующий курс позволяет обеспечить углубленный уровень изучения информатики и его имеет смысл включать в [элективные](https://ru.wikipedia.org/wiki/%D0%90%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) курсы углубленного уровня подготовки.

JavaScript — подходящий язык для обучения [программированию игр](https://ru.wikipedia.org/w/index.php?title=%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%B8%D0%B3%D1%80&action=edit&redlink=1). По сравнению с альтернативами, он функционально достаточен, прост в изучении и в применении, снижает сложность для обучения, мотивирует обучаемых делиться своими играми с другими.

Не включённые в книгу [Николаса Закаса](https://ru.wikipedia.org/w/index.php?title=%D0%97%D0%B0%D0%BA%D0%B0%D1%81,_%D0%9D%D0%B8%D0%BA%D0%BE%D0%BB%D0%B0%D1%81&action=edit&redlink=1) «Professional JavaScript for Web Developers» части о реализации на JavaScript классических [алгоритмов](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC), техник, [структур данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), послужили началу проекта Computer science in JavaScript.

**Версии**

|  |  |  |
| --- | --- | --- |
| JavaScript | Соответствующая версия JScript | Существеsнные изменения |
| 1.0 ([Netscape](https://ru.wikipedia.org/wiki/Netscape_Navigator" \o "Netscape Navigator) 2.0, март 1996) | 1.0 (ранние версии[IE](https://ru.wikipedia.org/wiki/Microsoft_Internet_Explorer) 3.0, август 1996) | Оригинальная версия языка JavaScript. |
| 1.1 (Netscape 3.0, август 1996) | 2.0 (поздние версии IE 3.0, январь 1997) | В данной версии был реализован объект Array и устранены наиболее серьёзные ошибки. |
| 1.2 (Netscape 4.0, июнь 1997) |  | Реализован переключатель switch, регулярные выражения. Практически приведён в соответствии с первой редакцией спецификации ECMA-262. |
| 1.3 (Netscape 4.5, октябрь 1998) | 3.0 (IE 4.0, октябрь 1997) | Совместим с первой редакцией ECMA-262. |
| 1.4 (только Netscape Server) | 4.0 ([Visual Studio](https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio) 6, нет версии IE) | Применяется только в серверных продуктах Netscape. |
|  | 5.0 (IE 5.0, март 1999) |  |
|  | 5.1 (IE 5.01) |  |
| 1.5 (Netscape 6.0, ноябрь 2000; также поздние версии Netscape и[Mozilla](https://ru.wikipedia.org/wiki/Mozilla)) | 5.5 (IE 5.5, июль 2000) | Редакция 3 (декабрь 1999). Совместим с третьей редакцией спецификации ECMA-262. |
|  | 5.6 (IE 6.0, октябрь 2001) |  |
| 1.6 ([Gecko](https://ru.wikipedia.org/wiki/Gecko_(%D0%B4%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA)" \o "Gecko (движок)) 1.8, [Firefox](https://ru.wikipedia.org/wiki/Mozilla_Firefox" \o "Mozilla Firefox) 1.5, ноябрь 2005) |  | Редакция 3 с некоторыми совместимыми улучшениями: [E4X](https://ru.wikipedia.org/wiki/ECMAScript_for_XML), дополнения к Array (например,Array.prototype.forEach), упрощения для Array и String[[92]](https://ru.wikipedia.org/wiki/JavaScript" \l "cite_note-105) |
| 1.7 (Gecko 1.8.1, Firefox 2.0, осень 2006), расширение JavaScript 1.6 |  | Редакция 3, с добавлением всех улучшений из JavaScript 1.6, генераторов и списочных выражений (list comprehensions, [a\*a for (a in iter)]) из [Python](https://ru.wikipedia.org/wiki/Python_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)" \o "Python (язык программирования)), блоковых областей с использованием let и деструктурирующего присваивания (var [a, b] = [1, 2])[[93]](https://ru.wikipedia.org/wiki/JavaScript#cite_note-106). |
|  | [JScript .NET](https://ru.wikipedia.org/wiki/Microsoft_JScript_.NET)([ASP.NET](https://ru.wikipedia.org/wiki/Microsoft_ASP.NET); нет версии IE) | (Считается, что JScript .NET разработан при участии других членов [ECMA](https://ru.wikipedia.org/wiki/ECMA)) |
| 1.8 (Gecko 1.9, Firefox 3.0, осень 2008), расширение JavaScript 1.7 |  | Новая форма записи для функций, сходная с типичными [лямбда-выражениями](https://ru.wikipedia.org/wiki/%D0%9B%D1%8F%D0%BC%D0%B1%D0%B4%D0%B0-%D0%B2%D1%8B%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F), [генераторы](https://ru.wikipedia.org/w/index.php?title=%D0%93%D0%B5%D0%BD%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)&action=edit&redlink=1) ([англ.](https://en.wikipedia.org/wiki/Generator_(computer_science))), новые методы итеративной обработки массивов reduce() и reduceRight(). |
| 1.8.1 (Gecko 1.9.1, Firefox 3.5) |  | Встроенная поддержка JSON, метод getPrototypeOf() у Object, методы [trim()](https://ru.wikipedia.org/wiki/Trim_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)" \o "Trim (программирование)), trimLeft(),trimRight() у String |
| 2.0 |  | Редакция 4 (разработка не закончена, название зарезервировано ECMA, но не было использовано для публикации) |
|  |  | Редакция 5 (ранее известная под названием ECMAScript 3.1. Финальная версия принята 3 декабря 2009 года) |

**Примеры использования клиентского JavaScript**

Веб-браузер, оснащенный интерпретатором JavaScript, позволяет распространять через Интернет исполняемое содержимое в виде JavaScript-сценариев. В примере 1.1 показана простая программа на языке JavaScript, которая и представляет собой сценарий, встроенный в веб-страницу.

*Пример 1.1. Простая программа на языке JavaScript*

<html>

<head><title>Факториалы</title></head>

<body>

<h2>Таблица факториалов</h2>

<script>

var fact = 1;

for(i = 1; i < 10; i++) {

fact = fact\*i;

document.write(i + "! = " + fact + "<br>");

}

</script>

</body>

</html>

После загрузки в браузер, поддерживающий JavaScript, этот сценарий выдаст результат, показанный на рисунке.



Как видно из этого примера, для встраивания JavaScript-кода в HTML-файл были использованы теги <script> и </script>. Главное, что демонстрируется в данном примере, – это использование метода document.write(). Этот метод позволяет динамически выводить HTML-текст внутри HTML-документа по мере его загрузки веб-браузером. JavaScript обеспечивает возможность управления не только содержимым HTML-документов, но и их поведением. Другими словами, JavaScript-программа может реагировать на действия пользователя: ввод значения в текстовое поле или щелчок мышью в области изображения в документе. Это достигается путем определения *обработчиков событий* для документа – фрагментов JavaScript\_кода, исполняемых при возникновении определенного события, например щелчка на кнопке. В примере 1.2 показан простой фрагмент HTML\_кода, который включает в себя обработчик события, вызываемый в ответ на такой щелчок.

*Пример 1.2. HTML'кнопка с обработчиком события на языке JavaScript*

<button onclick="alert('Был зафиксирован щелчок на кнопке');">

Щелкни здесь

</button>

На рисунке показан результат щелчка на кнопке.



Атрибут onclick из примера 1.2 – это строка JavaScript\_кода, исполняемого, когда пользователь щелкает на кнопке. В данном случае обработчик события onclick вызывает функцию alert(). Как видно из рисунка, функция alert() выводит диалоговое окно с указанным сообщением.

Примеры 1.1 и 1.2 демонстрируют лишь простейшие возможности клиентского JavaScript. Реальная его мощь состоит в том, что сценарии имеют доступ к содержимому HTML-документов.

**Утечки памяти**

Иногда могут возникать утечки памяти, они в реальной жизни проявляют себя в двух ситуациях:

1. Приложение, в котором посетитель все время на одной странице и работает со сложным JavaScript-интерфейсом. В этом случае утечки могут постепенно съедать доступную память.
2. Страница регулярно делает что-то, вызывающее утечку памяти. Посетитель (например, менеджер) оставляет компьютер на ночь включенным, чтобы не закрывать браузер с кучей вкладок. Приходит утром — а браузер съел всю память и сильно тормозит.

Утечки бывают из-за ошибок браузера, ошибок в расширениях браузера и, гораздо реже, по причине ошибок в архитектуре JavaScript-кода.

**Изучение JavaScript**

Наиболее очевидный подход к изучению JavaScript – это написание простых сценариев. Одно из достоинств клиентского JavaScript состоит в том, что любой, кто имеет веб-браузер и простейший текстовый редактор, имеет и полноценную среду разработки. Для того чтобы начать писать программы на JavaScript, нет необходимости в покупке или загрузке специального ПО.

Например, чтобы вместо факториалов вывести последовательность чисел Фибоначчи, пример 1.1 можно переписать следующим образом:

<script>

document.write("<h2>Числа Фибоначчи </h2>");

for (i=0, j=1, k=0, fib =0; i<50; i++, fib=j+k, j=k, k=fib){

document.write("Fibonacci (" + i + ") = " + fib);

document.write("<br>");

}

</script>

Этот отрывок может показаться запутанным, но для того чтобы поэкспериментировать с подобными короткими программами, достаточно набрать код и запустить его в веб-браузере в качестве файла с локальным URL-адресом. Обратите внимание, что для вывода результатов вычислений используется метод document.write().

Это полезный прием при экспериментах с JavaScript. В качестве альтернативы для отображения текстового результата в диалоговом окне можно применять метод alert():

alert("Fibonacci (" + i + ") = " + fib);

Отметим, что в подобных простых экспериментах с JavaScript можно опускать теги <html>, <head> и <body> в HTML\_файле.

Для еще большего упрощения экспериментов с JavaScript можно использовать URL-адрес со спецификатором псевдопротокола javascript: для вычисления значения JavaScript-выражения и получения результата. Такой URL-адрес состоит из спецификатора псевдопротокола (javascript:), за которым указывается произвольный JavaScript-код (инструкции отделяются одна от другой точками с запятой). Загружая URL-адрес с псевдопротоколом, браузер просто исполняет JavaScript-код. Значение последнего выражения в таком URL-адресе преобразуется в строку, и эта строка выводится веб-браузером в качестве нового документа. Например, для того чтобы проверить свое понимание некоторых операторов и инструкций языка JavaScript, можно набрать следующие URL-адреса в адресном поле веб-браузера:

javascript:5%2

javascript:x = 3; (x < 5)? "значение x меньше": "значение x больше"

javascript:d = new Date(); typeof d;

javascript:for(i=0,j=1,k=0,fib=1; i<5; i++,fib=j+k,k=j,j=fib) alert(fib);

javascript:s=""; for(i in navigator) s+=i+":"+navigator[i]+"\n"; alert(s);

В веб-браузере Firefox однострочные сценарии вводятся в JavaScript-консоли, доступ к которой можно получить из меню Инструменты. Просто введите выражение или инструкцию, которую требуется проверить. При использовании JavaScript-консоли спецификатор псевдопротокола (javascript:) можно опустить.

**Тестирование и дебаг**

Не любой код, написанный вами при изучении JavaScript, будет работать так, как ожидается, и вам захочется его отладить. Базовая методика отладки JavaScript-кода совпадает с методикой для многих других языков: вставка в код инструкций, которые будут выводить значения нужных переменных так, чтобы можно было понять, что же на самом деле происходит. Как мы уже видели, иногда для этих целей можно использовать метод document.write() или alert().

Также существуют методы тестирования кода. После завершения разработки новой функции (возможен вариант написания тестов и до начала разработки) девелопер пишет специальный код для тестирования своего кода. В коде для тестирования нужно сымитировать различные ситуации и возвращаемые значения.

QUnit пользуется особой популярностью среди JavaScript разработчиков. Во-первых, она отлично документирована и проста в использовании, а во-вторых она создана авторами jQuery. Библиотека подходит как для тестирования кода, созданного на базе jQuery, так и нативного JavaScript

.  
<http://qunitjs.com/>

Альтернативы QUnit.js: [Buster.JS](http://busterjs.org/), [Capybara](http://goo.gl/pdlPjl), [Mocha](http://goo.gl/4dQlE), [FuncUnit](http://funcunit.com/), [Hiro](http://hirojs.com/), [Jasmine](http://goo.gl/EUCNl), [Laika](http://goo.gl/sT3Tz), [RobotFramework](http://goo.gl/MnN50R), [TapeDeck](http://goo.gl/PPtPb6), [Casper.js](http://casperjs.org/)

Если же речь идет о работе с Node.js, то с помощью команды node-debug filename Вы сможете дебажить программу с помощью консольной утилиты Node Debugger.

Также можно установить отдельно Node Inspector для более удобного пользования.

**Конкуренты JavaScript**

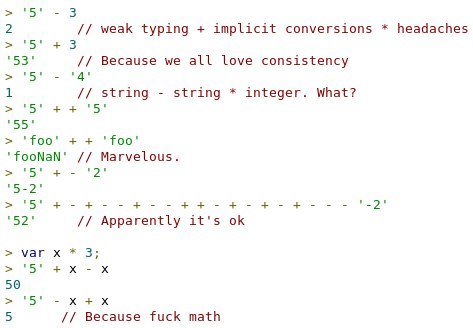
* + 1. VBScript  
       Visual Basic Scripting Edition (или просто VBScript) — это язык программирования от компании Microsoft, предназначенный для создания сценариев (скриптов). Он является подмножеством языка Visual Basic и широко используется при создании административных сценариев в системе Windows.
    2. JScript  
       JScript — это язык программирования от компании Microsoft. Он предназначен для создания сценариев и является реализацией стандарта ECMAScript.
    3. Python  
       Python (питон) — интерпретируемый, объектно-ориентированный язык программирования высокого уровня. Он поддерживает классы, модули (которые могут быть объединены в пакеты), обработку исключений, а также многонитевую обработку.
    4. Tcl  
       Tcl (Tool Command Language) — интерпретируемый язык программирования высокого уровня.
    5. Ruby  
       Ruby — интерпретируемый скриптовый язык высокого уровня для быстрого и удобного объектно-ориентированного программирования.
    6. PHP  
       PHP (пи-эйч-пи) — интерпретируемый скриптовый язык программирования, созданный для генерации HTML-страниц на веб-сервере и работы с базами данных.
    7. Perl  
       Perl — интерпретируемый скриптовый язык программирования, один из самых распространённых в области веб-программирования.

**Языки на базе JavaScript**

* CoffeeScript  
  CoffeeScript это небольшой язык, который транслируется в Javascript. Рубистам он кажется похожим на руби, питонистам он похож на питон, и конечно же, он похож на яваскрипт. CoffeeScript старается упростить использование яваскрипта, сохранив все его сильные стороны.   
  <http://coffeescript.org/>
* Dart  
  Dart это объектно-ориентированый язык с полноценной системой классов, лексическими скоупами, замыканиями, и опционально — статической типизацией. Dart помогает создавать структурированные веб приложения и легок в изучении для широкого круга разработчиков.   
  <https://www.dartlang.org/>
* TypeScript  
  TypeScript отличается от JavaScript возможностью явного определения типов (статическая типизация), поддержкой использования полноценных классов (как в традиционных объектно-ориентированных языках), а также поддержкой подключения модулей.  
  <http://www.typescriptlang.org/>
* ClojureScript  
  ClojureScript — это расширение языка Clojure, с возможностью компиляции в Javascript. Напоминает Lisp.  
  <https://github.com/clojure/clojurescript>
* Opal  
  Компилятор из Ruby в Javascript.   
  <http://opalrb.org/>
* IcedCoffeeScript  
  IcedCoffeeScript это надстройка над CoffeeScript, упрощающая контроль за асинхронными операциями. Вместо колбеков вводятся два новых оператора: await и defer.   
  <http://maxtaco.github.io/coffee-script/>
* LiveScript  
  Ещё один форк от CoffeeScript. Добавляет поддержку функционального стиля программирования, а также вводит небольшие улучшения в текущую ООП-модель  
  <http://livescript.net/>
* Kaffeine  
  Расширяет синтакс яваскрипта, не изобретая ещё один язык программирования. Код на Kaffeine строка к строке соответствует скомпилированому яваскрипт коду. Данная фича должна существенно упростить отладку приложения.   
  <http://weepy.github.io/kaffeine/>
* Roy

Экспериментальный язык программирования, преобразующий код в Javascript. Включает в себя возможности статичных функциональных языков  
http://roy.brianmckenna.org/

**“Логичность JavaScript – не баг, а фича”**



NaN === NaN; // true or false?  
  
typeof NaN; // what type?

Также для проверки является ли переменная NaN существует функция isNaN, но она работает не до конца корректно. С самых ранних версий функции isNaN её поведение для не числовых переменных или литералов было довольно-таки запутанным. Когда аргументом функции isNaN является переменная, тип которой не [Number](http://es5.github.com/" \l "x8.5" \o "http://es5.github.com/#x8.5), она преобразуется к типу Number. К примеру, если мы возьмем строку var hello = «Hello»; и проверим isNaN(hello), то получим true. Но на самом деле, hello имеет тип String, но из-за неудавшегося преобразования к числу, получаем соответствующий результат. Но если мы возьмем пустую строку: isNaN(‘’), то получим false, так как пустая строка может быть преобразована к числу 0. Аналогично функция возвращает false, если строка состоит из числа. Даже isNaN('NaN') возвращает true. Эта функция является одной из ошибок логики языка JavaScript. От этого программисты ее просто не используют, а значение NaN проверяется из того, что NaN != NaN.

0.1+ 0.2 !== 0.3 (0.30000000000000004)

'3'-+-+-+'1'+'1'/'3'\*'6'+'2'

1. Операции выполняются в порядке приоритета:   
   ‘1’ / ’3’ = 0.3333333333333333,   
   0.3333333333333333 \* ’6’ = 2
2. '3'-+-+-+'1' = 2
3. 2 + 2 = 4
4. 4 + ‘2’ = ‘42’

Больше странностей JS можно обнаружить на ресурсе:

<http://wtfjs.com/>

**Перспективы на 2015-2016 годы**

* Новый стандарт ECMAScript 6. Утверждение, реализация в браузерах, адаптация в сообществе и фремворках.

Новый стандарт несет давно ожидаемые возможности, которые существенно облегчат создание сложных решений: классы, модули, коллекции, итераторы, генераторы, прокси, типизированные массивы, обещания, новые методы и свойства для стандартных объектов и новые синтаксические возможности и еще много чего.

* Рост использования TypeScript в реальных проектах, развитие альтернативных проектов и их взаимное обогащение.

TypeScript – разработка компании Microsoft.

TypeScript является надмножеством JavaScript. То есть любой корректный код на JavaScript также является корректным кодом на TypeScript.

TypeScript использует статическую типизацию, то есть все типы проверяются при компиляции. Кстати сам компилятор TypeScript написан на TypeScript и является open source

TypeScript добавляет возможность объявлять модули, классы и интерфейсы. Это позволяет масштабировать разработку сложных JavaScript приложений.

На выходе получается обычный JavaScript, который не требует дополнительных библиотек или специальных компонентов.

С TypeScript можно работать на Visual Studio, при несовпадении типов компилятор будет «ругаться».

Код TypeScript является открытым.

// TypeScript

class Greeter {

greeting: string;

constructor(message: string) {

this.greeting = message;

}

greet() {

return "Hello, " + this.greeting;

}

}  
  
Прелесть TypeScript в том, что, пока вы пишите код, вы получаете возможность удобно описывать сложные структуры данных, а компилятор при этом помогает вам отслеживать, что вы нигде ничего не напутали и правильно работаете с типами.  
  
Еще одно замечательное свойство TS, точнее его компилятора (который, кстати, [открыт также, как и сам язык!](https://github.com/Microsoft/TypeScript)), состоит в том, что в результате компиляции получается чистый код на JavaScript, причем, примерно такой, какой вы бы и сами написали, следуя современным практикам:

// TypeScript to JavaScript

var Greeter = (function () {

function Greeter(message) {

this.greeting = message;

}

Greeter.prototype.greet = function () {

return "Hello, " + this.greeting;

};

return Greeter;

})();

Таким образом, на выходе получается код, работающий в современных   
браузерах на любых операционных системах. К слову, под Node.js   
тоже можно писать на TypeScript.

<http://www.typescriptlang.org/>

* Развитие инструментов для кросс-платформенной разработки на JS, продолжение стирания границ между сайтами и приложениями.

В первой задаче критичным является стремление научить веб-сайты делать то, что умеют делать приложения: например, интегрироваться в операционную систему – от иконок и живых плиток до пуш-уведомлений и поддержки локальных контрактов. При этом важным остается система обновления содержимого таких приложений через веб-сайт, что позволяет сохранять преимущества веб-подхода.

* Рост умных телевизоров и консолей с разработкой на JavaScript, нативная разработка на JS на многих современных платформах (но не всех).
* Развитие API доступа к нативным возможностям устройства из JavaScript, адаптация NUI в JS. (Затянется на несколько лет.)

С самыми базовыми вещами вроде геолокации или ориентации устройства мы уже научились работать, но впереди [большая работа по стандартизации](http://www.w3.org/2009/dap/) и реализации в движках браузеров большого блока возможностей, доступных в случае нативной разработки, но, как правило, неподвластных в случае разработки для браузера:

* + Вибрация
  + Статус батареи
  + Сенсоры (например, света)
  + Камера и микрофон
  + и др.
* Новые переработанные версии популярных библиотек, повышение входного порога для создания комплексных фреймворков, нишевые решения на базе ES6.
* Адаптацию веб-компонент браузерами, принятие новых технологий разработчиками элементов управления и различных фреймворков.
* Принятие менеджеров пакетов и систем сборки для JavaScript в корпоративной и учебной среде, интеграция в популярные инструменты веб-разработки.
* Развитие графических библиотек на JS, показательная адаптация новых технологий крупными или заметными игроками рынка (игры и интерактивный контент — основные драйверы).
* Unity 5 с рендерингом в WebGL, развитие 3d и игровых библиотек, потенциальный прорыв через социальные сети.
* Адаптация Node.js в корпоративной среде, адаптация нового ES6 в самом Node.js. Запасаемся попкорном и смотрим историю с ответвлением io.js.

Кстати, в конце 2015 нас ждет большой праздник – 20-летие JavaScript ☺