**JavaScript** – один из самых популярных языков программирования в мире. Он используется для создания интерактивных веб-страниц, мобильных приложений, в серверной разработке. Изучать JS мы будем с нуля, с самых азов. Первый модуль – плацдарм  для написания осмысленных программ. В нем мы разберем, как написать свой первый код на JS. Расскажем, что такое комментарии и зачем они нужны. На примере проверки ваших решений рассмотрим, что такое тестирование и как читать вывод тестов.

Стек вызовов

**JavaScript** — однопоточный язык программирования. Это означает, что у него один стек вызовов. Таким образом, в некий момент времени он может выполнять лишь какую-то одну задачу.  
  
Стек вызовов — это структура данных, которая, говоря упрощённо, записывает сведения о месте в программе, где мы находимся. Если мы переходим в функцию, мы помещаем запись о ней в верхнюю часть стека. Когда мы из функции возвращаемся, мы вытаскиваем из стека самый верхний элемент и оказываемся там, откуда вызывали эту функцию. Это — всё, что умеет стек.

**Как работают движки?**

Движки сложны. Но основы понять легко.

1. Движок (встроенный, если это браузер) читает («парсит») текст скрипта.
2. Затем он преобразует («компилирует») скрипт в машинный язык.
3. После этого машинный код запускается и работает достаточно быстро.

Движок применяет оптимизации на каждом этапе. Он даже просматривает скомпилированный скрипт во время его работы, анализируя проходящие через него данные, и применяет оптимизации к машинному коду, полагаясь на полученные знания. В результате скрипты работают очень быстро.

**[Что может JavaScript в браузере?](https://learn.javascript.ru/intro" \l "chto-mozhet-javascript-v-brauzere)**

Современный JavaScript – это «безопасный» язык программирования. Он не предоставляет низкоуровневый доступ к памяти или процессору, потому что изначально был создан для браузеров, не требующих этого.

Возможности JavaScript сильно зависят от окружения, в котором он работает. Например, [Node.JS](https://ru.wikipedia.org/wiki/Node.js) поддерживает функции чтения/записи произвольных файлов, выполнения сетевых запросов и т.д.

В браузере для JavaScript доступно всё, что связано с манипулированием веб-страницами, взаимодействием с пользователем и веб-сервером.

Например, в браузере JavaScript может:

* Добавлять новый HTML-код на страницу, изменять существующее содержимое, модифицировать стили.
* Реагировать на действия пользователя, щелчки мыши, перемещения указателя, нажатия клавиш.
* Отправлять сетевые запросы на удалённые сервера, скачивать и загружать файлы (технологии [AJAX](https://ru.wikipedia.org/wiki/AJAX) и [COMET](https://ru.wikipedia.org/wiki/Comet_(%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))).
* Получать и устанавливать куки, задавать вопросы посетителю, показывать сообщения.
* Запоминать данные на стороне клиента («local storage»).

**[Чего НЕ может JavaScript в браузере?](https://learn.javascript.ru/intro" \l "chego-ne-mozhet-javascript-v-brauzere)**

Возможности JavaScript в браузере ограничены ради безопасности пользователя. Цель заключается в предотвращении доступа недобросовестной веб-страницы к личной информации или нанесения ущерба данным пользователя.

Примеры таких ограничений включают в себя:

* JavaScript на веб-странице не может читать/записывать произвольные файлы на жёстком диске, копировать их или запускать программы. Он не имеет прямого доступа к системным функциям ОС.

Современные браузеры позволяют ему работать с файлами, но с ограниченным доступом, и предоставляют его, только если пользователь выполняет определённые действия, такие как «перетаскивание» файла в окно браузера или его выбор с помощью тега <input>.

Существуют способы взаимодействия с камерой/микрофоном и другими устройствами, но они требуют явного разрешения пользователя. Таким образом, страница с поддержкой JavaScript не может незаметно включить веб-камеру, наблюдать за происходящим и отправлять информацию в [ФСБ](https://ru.wikipedia.org/wiki/%D0%A4%D0%B5%D0%B4%D0%B5%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%BB%D1%83%D0%B6%D0%B1%D0%B0_%D0%B1%D0%B5%D0%B7%D0%BE%D0%BF%D0%B0%D1%81%D0%BD%D0%BE%D1%81%D1%82%D0%B8_%D0%A0%D0%BE%D1%81%D1%81%D0%B8%D0%B9%D1%81%D0%BA%D0%BE%D0%B9_%D0%A4%D0%B5%D0%B4%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%B8).

* Различные окна/вкладки не знают друг о друге. Иногда одно окно, используя JavaScript, открывает другое окно. Но даже в этом случае JavaScript с одной страницы не имеет доступа к другой, если они пришли с разных сайтов (с другого домена, протокола или порта).

Это называется «Политика одинакового источника» (Same Origin Policy). Чтобы обойти это ограничение, обе страницы должны согласиться с этим и содержать JavaScript-код, который специальным образом обменивается данными.

Это ограничение необходимо, опять же, для безопасности пользователя. Страница https://anysite.com, которую открыл пользователь, не должна иметь доступ к другой вкладке браузера с URL https://gmail.com и воровать информацию оттуда.

* JavaScript может легко взаимодействовать с сервером, с которого пришла текущая страница. Но его способность получать данные с других сайтов/доменов ограничена. Хотя это возможно в принципе, для чего требуется явное согласие (выраженное в заголовках HTTP) с удалённой стороной. Опять же, это ограничение безопасности.

**[Что делает JavaScript особенным?](https://learn.javascript.ru/intro" \l "chto-delaet-javascript-osobennym)**

Как минимум, *три* сильные стороны JavaScript:

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

JavaScript – это единственная браузерная технология, сочетающая в себе все эти три вещи.

Вот что делает JavaScript особенным. Вот почему это самый распространённый инструмент для создания интерфейсов в браузере.

Хотя, конечно, JavaScript позволяет делать приложения не только в браузерах, но и на сервере, на мобильных устройствах и т.п.

**[Языки «над» JavaScript](https://learn.javascript.ru/intro" \l "yazyki-nad-javascript)**

Синтаксис JavaScript подходит не под все нужды. Разные люди хотят иметь разные возможности.

Это естественно, потому что проекты разные и требования к ним тоже разные.

Так, в последнее время появилось много новых языков, которые *транспилируются* (конвертируются) в JavaScript, прежде чем запустятся в браузере.

Современные инструменты делают транспиляцию очень быстрой и прозрачной, фактически позволяя разработчикам писать код на другом языке, автоматически преобразуя его в JavaScript «под капотом».

Примеры таких языков:

* [CoffeeScript](http://coffeescript.org/) добавляет «синтаксический сахар» для JavaScript. Он вводит более короткий синтаксис, который позволяет писать чистый и лаконичный код. Обычно такое нравится Ruby-программистам.
* [TypeScript](http://www.typescriptlang.org/) концентрируется на добавлении «строгой типизации» для упрощения разработки и поддержки больших и сложных систем. Разработан Microsoft.
* [Flow](http://flow.org/) тоже добавляет типизацию, но иначе. Разработан Facebook.
* [Dart](https://www.dartlang.org/) стоит особняком, потому что имеет собственный движок, работающий вне браузера (например, в мобильных приложениях). Первоначально был предложен Google, как замена JavaScript, но на данный момент необходима его транспиляция для запуска так же, как для вышеперечисленных языков.
* [Brython](https://brython.info/) транспилирует Python в JavaScript, что позволяет писать приложения на чистом Python без JavaScript.

Есть и другие. Но даже если мы используем один из этих языков, мы должны знать JavaScript, чтобы действительно понимать, что мы делаем.

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

* JavaScript изначально создавался только для браузера, но сейчас используется на многих других платформах.
* Сегодня JavaScript занимает уникальную позицию в качестве самого распространённого языка для браузера, обладающего полной интеграцией с HTML/CSS.
* Многие языки могут быть «транспилированы» в JavaScript для предоставления дополнительных функций. Рекомендуется хотя бы кратко рассмотреть их после освоения JavaScript.