

Современный учебник JavaScript

© Илья Кантор

Сборка от 23 июля 2013 для чтения с устройств

Внимание, эта сборка может быть устаревшей и не соответствовать текущему тексту.

Актуальный онлайн-учебник, с интерактивными примерами, доступен по адресу <http://learn.javascript.ru>.

Вопросы по JavaScript можно задавать в комментариях на сайте или на форуме javascript.ru/forum.

Вопросы по сборке, предложения по её улучшению – можно писать мне, по адресу iliakan@javascript.ru.

Глава: Общая информация

В файле находится только одна глава учебника. Это сделано в целях уменьшения размера файла, для удобного чтения с устройств.

Содержание

[Введение в JavaScript](#)

[Что такое JavaScript?](#)

[Что умеет JavaScript?](#)

[Что НЕ умеет JavaScript?](#)

[В чем уникальность JavaScript?](#)

[Тенденции развития.](#)

[HTML 5](#)

[EcmaScript](#)

[Недостатки JavaScript](#)

[Альтернативные технологии](#)

Java

ActiveX/NPAPI, плагины и расширения для браузера

Adobe Flash

Dart

Итого

Книги по JS, HTML/CSS и не только

CSS

JavaScript

jQuery

Объектно-ориентированное программирование

Регулярные выражения

Алгоритмы и структуры данных

Разработка и организация кода

Терминология, поддержка, справочники

Уровни поддержки

HTML 5

JS-Фреймворки

Справочники, и как в них искать

Спецификации

Спецификация ECMAScript

Спецификации HTML/CSS

Итого

Редакторы для кода

IDE

Лёгкие редакторы

Топ 3

P.S.

Sublime Text: шпаргалка

Горячие клавиши

Плагины

Установка браузеров, JS-консоль

Firefox

Установка Firebug

Включите консоль

Просмотр ошибок

Internet Explorer

Включаем отладку

Просмотр ошибок

Другие браузеры

Google Chrome

Safari

Opera

IE<8

Тестирование в старых браузерах

Internet Explorer

Firefox

Chrome

Opera

Привет, мир!

Тег SCRIPT

Внешние скрипты

Решения задач

Введение в JavaScript

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

Что такое JavaScript?

JavaScript изначально создавался для того, чтобы сделать web-странички «ЖИВЫМИ».

Программы на этом языке называются *скриптами*. 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.



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](#), и к Java не имеет никакого отношения.

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

Чтобы читать и выполнять текст на JavaScript, нужна специальная программа — [интерпретатор](#). Процесс выполнения скрипта называют «*интерпретацией*».



Компиляция и интерпретация, для программистов

Строго говоря, для выполнения программ существуют «компиляторы» и «интерпретаторы».

Когда-то между ними была большая разница. Компиляторы преобразовывали программу в машинный код, который потом можно выполнять. А интерпретаторы — просто выполняли.

Сейчас разница гораздо меньше. Современные интерпретаторы

перед выполнением преобразуют JavaScript в машинный код (или близко к нему), чтобы выполнялся он быстрее. То есть, по сути, компилируют, а затем запускают.

Во все основные браузеры встроен интерпретатор JavaScript, именно поэтому они могут выполнять скрипты на странице.

Но, разумеется, этим возможности JavaScript не ограничены. Это полноценный язык, программы на котором можно запускать и на сервере, и даже в стиральной машинке, если в ней установлен соответствующий интерпретатор.

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.

В браузере JavaScript умеет делать все, 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, перемещение курсора, нажатие 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!

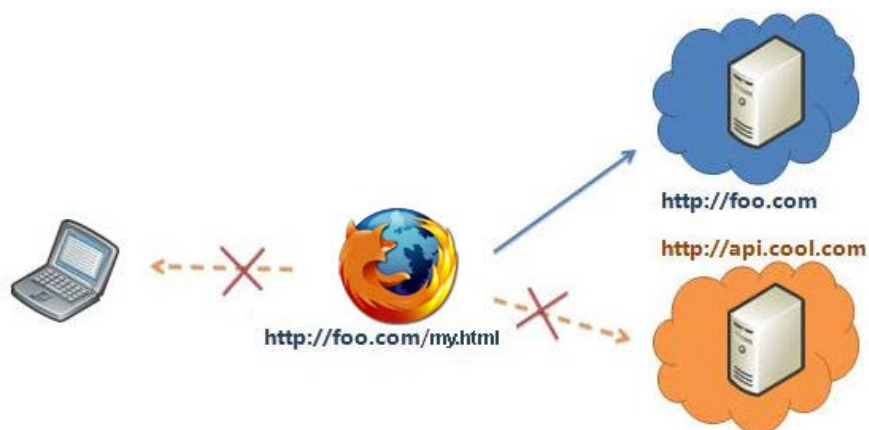
Что НЕ умеет JavaScript?

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. В браузере Firefox существует способ «подписи» скриптов с целью обхода части ограничений, но он нестандартный и не кросс-браузерный.

Этих ограничений нет там, где JavaScript используется вне браузера, например на сервере.

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, почти 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 для этого требуется как минимум явное согласие обеих сторон. Просто так взять и залезть в произвольную вкладку с другого домена нельзя.

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

В чем уникальность JavaScript?

Есть как минимум *три* замечательных особенности JavaScript:

-  → Полная интеграция с HTML/CSS.
- Простые вещи делаются просто.
- Поддерживается всеми распространенными браузерами и включен по умолчанию.

Этих трех вещей одновременно нет больше ни в одной браузерной технологии. Поэтому JavaScript и является самым распространенным средством создания браузерных интерфейсов.

Тенденции развития.

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

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 все еще в разработке, но браузеры постепенно начинают их поддерживать.

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

EcmaScript

Сам язык JavaScript улучшается. Современный стандарт EcmaScript 5 включает в себя новые возможности для разработки.

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

Тенденция: JavaScript становится все быстрее и стабильнее.

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 HTML5 все же есть. Иногда браузеры стараются включить новые возможности, которые еще 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 (старом) 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: все идет к полной совместимости со стандартом.

Недостатки JavaScript

Зачастую, недостатки подходов и технологий — это обратная сторона их полезности. Стоит ли упрекать молоток в том, что он — тяжелый? Да, неудобно, зато гвозди забиваются лучше.

В JavaScript, однако, есть вполне объективные недоработки, связанные с тем, что язык, по выражению его автора (Brendan Eich) делался «за 10 бессонных дней и ночей». Поэтому некоторые моменты продуманы плохо, есть и откровенные ошибки (которые признает тот же Brendan).

Конкретные примеры мы увидим в дальнейшем, т.к. их удобнее обсуждать в процессе освоения языка.

Пока что нам важно знать, что некоторые «странности» языка не являются чем-то очень умным, а просто не были достаточно хорошо продуманы в своё время. В этом учебнике мы будем обращать особое внимание на основные недоработки и «грабли». Ничего критичного в них нет, если знаешь — не наступишь.

В новых версиях JavaScript (ECMAScript) эти недостатки постепенно убирают. Процесс внедрения небыстрый, в первую очередь из-за старых версий IE, но они постепенно вымирают. Современный IE в этом отношении несравнимо лучше.

Альтернативные технологии

Вместе с JavaScript на страницах используются и другие технологии. Самые известные — это Flash, Java, ActiveX/NPAPI.

Они бывают нужны, так как возможности JavaScript ограничены. Связка JavaScript с Java, Flash или ActiveX может помочь достигнуть более

интересных результатов. Важно и то, что все эти технологии хорошо взаимодействуют между собой.

Java

Java — язык общего назначения, на нем 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`:

```
1 <applet code="BTApplet.class"
  codebase="/files/tutorial/intro/alt/">
2   <param name="nodes"
  value="50,30,70,20,40,60,80,35,65,75,85,90">
3   <param name="root" value="50">
4 </applet>
```

Такой тег загружает Java-программу из файла `BTApplet.class` и выполняет ее с параметрами `param`.

Конечно, для этого на компьютере должна быть установлена и включена среда выполнения Java. Статистика показывает, что примерно на 80% компьютеров Java будет работать.

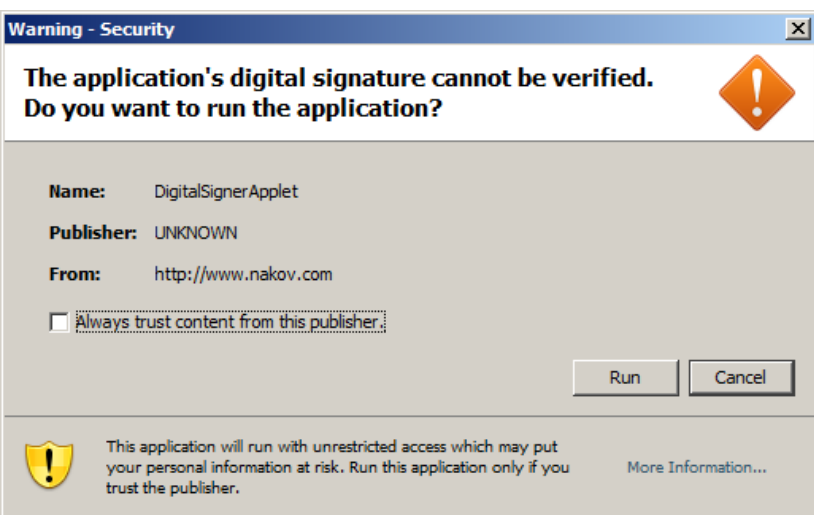
Апплет выполняется в отдельной части страницы, в прямоугольном «контейнере». Все действия пользователя внутри него обрабатывает апплет. Контейнер, впрочем, может быть и скрыт, если апплету нечего показывать.

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

В первую очередь тем, что ограничения безопасности JavaScript в принципе нельзя преодолеть... А специальным образом подписанный (это несложно) Java-апплет может делать все, если посетитель ему доверяет.

То есть, возможности, которые по ограничениям безопасности не поддерживает JavaScript, реализуемы через доверенный Java-апплет.

При попытке сделать потенциально опасное действие — пользователь получает вопрос, который выглядит примерно так:



Обойти это подтверждение или поменять его внешний вид нельзя. То есть, согласие посетителя действительно необходимо.

Произвольному апплету посетитель доверять не будет, но если он доверяет сайту — то разрешит. Например, если функционал находится в админской части сайта, или это сайт для банка, или это сервис, за который посетитель платит вам деньги... Тогда посетитель уже доверяет и позволит вашему апплету выполниться.



→ Java может делать все от имени посетителя, совсем как установленная десктопная программа. В целях безопасности, потенциально опасные действия требуют подписанного апплета и доверия пользователя.

→ Java требует больше времени для загрузки

→ Среда выполнения Java должна быть установлена на компьютере посетителя и включена. Таких посетителей в интернет — около 80%.

→ Java-апплет не интегрирован с HTML-страницей, а выполняется отдельно. Но он может вызывать функции JavaScript.

Подписанный Java-апплет - это возможность делать все, что

удовно, на компьютере посетителя, если он вам доверяет. Можно вынести в него все вызовы, которым нужно обойти контекст безопасности, а для самой страницы использовать JavaScript.

ActiveX/NPAPI, плагины и расширения для браузера

ActiveX для IE и NPAPI для остальных браузеров позволяют писать плагины для браузера, в том числе на языке C. Как и в ситуации с Java-апплетом, посетитель поставит их в том случае, если вам доверяет. Эти плагины могут как отображать содержимое специального формата (плагин для проигрывания музыки, для показа PDF), так и взаимодействовать со страницей.

ActiveX при этом еще и очень удобен в установке. Лично я - не фанат Microsoft, но видел отличные приложения, написанные на ActiveX и я могу понять, почему люди используют его и привязываются к IE.

Adobe Flash

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

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

Нам Flash интересен тем, что позволяет делать многое, что JavaScript пока не умеет, например работа с микрофоном, камерой, с буфером обмена.

В отличие от технологий, рассмотренных ранее, он не может «все», но зато работает безопасно и не требует доверия посетителя.



→ Большие возможности для работы в сети(сокеты, UDP для P2P)

→ Поддержка мультимедиа: изображения, аудио, видео. Работа с

веб-камерой и микрофоном.

- Flash должен быть установлен и включен. 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.

JavaScript и ActionScript могут вызывать функции друг друга, поэтому обычно сайты используют JavaScript, а там, где он не справляется - можно подумать о Flash.

Dart

Язык Dart предложен компанией Google как замена JavaScript, у которого, по выражению создателей Dart, есть **фатальные недостатки**.

Сейчас этот язык, хотя и доступен, находится в стадии разработки и тестирования. Многие из возможностей еще ожидают своей реализации, есть ряд проблем. Другие ведущие интернет-компании объявляли о своей незаинтересованности в Dart.

..Но в будущем он может составить конкуренцию JS, если его доведут до ума... Ну или если Google завоюет мир 😊.

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, ActiveX/NPAPI имеют свои уникальные возможности, которые можно использовать вместе с JavaScript.

На Dart сейчас тратить время не стоит, но, возможно, будет иметь смысл поглядеть на него через годик-другой.

Книги по JS, HTML/CSS и не только

При освоении JavaScript вам понадобятся как смежные технологии, так и знание общей методологии программирования.

Так как это учебник, то здесь вы найдете конкретную литературу, с которой целесообразно начинать изучение. Всего несколько книг на каждую тему. Из большого количества все равно пришлось бы выбирать.

Кстати, по всем книжкам, особенно тех, которые касаются технологий, всегда ищите последнее издание.

P.S. Скачать книги здесь нельзя. Эта страница содержит только рекомендации.

CSS

CSS стоит изучать по одной из этих книг. Можно сразу по обеим.

- ⇒ [CSS ручной работы.](#)

Дэн Седерхольм.

- ⇒ [Большая книга CSS.](#)

Дэвид Макфарланд.

Для того, чтобы разобраться в конкретных вопросах CSS, и в качестве справочника полезна книга Эрика Мейера [CSS. Каскадные таблицы стилей. Подробное руководство.](#), а также [стандарт CSS 2.1.](#)

JavaScript

Полезное чтение о языке, встроенных методах и конструкциях JavaScript:

- ⇒ [JavaScript. Подробное руководство.](#)

Дэвид Флэнаган.

- ⇒ [JavaScript. Шаблоны.](#)

Стоян Стефанов.

jQuery

Кроме [документации](#):

→ [jQuery. Подробное руководство по продвинутому JavaScript.](#)

Бер Бибо, Иегуда Кац.

Объектно-ориентированное программирование

Объектно-ориентированное программирование (ООП) — это концепция построения программных систем на основе объектов и взаимодействия между ними. При изучении ООП рассматриваются полезные архитектурные приёмы, как организовать программу более эффективно.

Умение создавать объект, конструктор, вызывать методы — это основные, самые базовые «кирпичики». Их следует освоить первыми, например используя этот учебник. Затем, когда основы более-менее освоены, стоит уделить внимание теории объектно-ориентированной разработки:

→ [Объектно-ориентированный анализ и проектирование с примерами приложений.](#)

Гради Буч и др..

→ [Приемы объектно-ориентированного проектирования. Паттерны проектирования.](#)

Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес.

Регулярные выражения

→ [Регулярные выражения.](#)

Джеффри Фридл.

Эта книга описывает более широкий класс регэкспов, по сравнению с текущим JavaScript. С одной стороны, какая-то информация будет лишней, с другой — регулярные выражения вообще очень важная и полезная тема.

Алгоритмы и структуры данных

→ [Алгоритмы. Построение и анализ.](#)

Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн.

Есть и другая классика, например «Искусство программирования», Дональд Кнут. Она требует более серьёзной математической подготовки. Будьте готовы читать и вникать долго и упорно. Результат — апгрейд мозговых извилин и общего умения программировать. Серьёзно, оно того стоит.

Разработка и организация кода

→ [Совершенный код.](#)

Стив Макконнелл.

Это желательно изучать уже после получения какого-то опыта в программировании.

Терминология, поддержка, справочники

В этом разделе мы познакомимся с важными терминами и справочниками — тем, что обязательно понадобится при создании веб-приложений.

Уровни поддержки

Иногда кросс-браузерная разработка становится непростым делом. Поэтому браузеры разделяют по степени поддержки.

Например:

A. Последние версии Firefox, Internet Explorer, Safari/Chrome

Идеальная поддержка.

B. Opera и предыдущие версии современных браузеров

Возможны незначительные «помарки», не ломающие функционал.

С. Старые браузеры

Поддерживаются только базовые возможности

D. Очень старые браузеры, текстовые браузеры

Не поддерживаются.

В зависимости от целевой аудитории сайта, Opera может включаться в уровень A, классификация может меняться, учитывать мобильные браузеры и т.п.

При любой веб-разработке важно определиться, что поддерживается и до какой степени.

В задачах учебника предусматривается поддержка современных браузеров, включая Firefox, Safari/Chrome, Opera, IE8+ в группе A, а также поддержка IE7 по группе B.

Вы можете выбрать свои уровни поддержки и решать задачи, соответственно ориентируясь на них.



Терминология

Иногда в интернет можно встретить названия не браузеров, а их движков. Полезно понимать, что есть что.

Gecko

Открытый движок **Firefox** и некоторых менее известных браузеров от компании Mozilla.

Webkit

Общий браузерный движок браузеров **Safari** и **Chrome**. Он отвечает за показ HTML/CSS, возможности HTML 5, кроме собственно языка JavaScript. Здесь браузеры расходятся: Chrome использует *Google V8*, а Safari — свой собственный закрытый движок *Squirrelfish*. Но так как оба этих браузера основаны на Webkit, то между ними много общего.

Presto

Старый движок браузера **Opera**. Этот браузер находится в процессе перехода на Webkit, после чего предыдущие трюки с

Opera станут неактуальны.

Trident

Закрытый движок браузера **Internet Explorer**. Trident ужасен в IE6 и IE7, улучшен в IE8, в IE9 он, за небольшим исключением, полностью поддерживает современный стандарт, ну а в IE10 вообще вполне хорош.

HTML 5

Как вам, наверное, известно из опыта работы с HTML, существует два режима документа: нормальный *Standards Mode* и старый *Quirks mode*. По правде говоря, есть еще и третий режим, который называется *Almost Standards Mode*,

Браузер выбирает режим в соответствии с директивой DOCTYPE, прописанной в начале HTML документа. Если для вас это новость - про режимы рендеринга можно почитать, например, на википедии, в статье [Wikipedia quirks mode page](#) .

Для нас здесь важно одно — **HTML обязан начинаться с DOCTYPE**.

Мы будем использовать DOCTYPE для HTML 5, который, впрочем, поддерживается и старыми браузерами:

```
1 <!DOCTYPE HTML>
2 <html>
3 ...
4 </html>
```



DOCTYPE влияет на всё!

DOCTYPE влияет не только на HTML/CSS — JavaScript также зависит от него. Использование неподходящего DOCTYPE или его отсутствие может стоить времени на отладку.

Просто используйте `<!DOCTYPE HTML>`.

JS-Фреймворки

Большая часть разработки осуществляется при помощи *фреймворков* — специальных библиотек, которые позволяют упростить код, а также содержат большое количество готовых компонентов.

Таких фреймворков много. Вот лишь некоторые из них:

- jQuery
- Dojo
- Ext.JS
- Angular.JS
- Backbone.JS
- ...

Бывают фреймворки общего назначения, которые содержат всего понемногу, бывают — направленные на конкретные аспекты разработки, например, создание интерфейсов или мобильные устройства.

Выбор конкретного фреймворка зависит от задачи. Фреймворк — это инструмент. Для решения задач бывает нужно несколько инструментов, а чем шире задача — тем больше.

Сначала нужно изучить сам JavaScript, затем работу с браузером, а уже потом приступать к фреймворкам.

Справочники, и как в них искать

Есть три основных справочника по JavaScript на английском языке:

1. [Mozilla Developer Network](#) — содержит информацию, верную для основных браузеров. Также там присутствуют расширения только для Firefox, они помечены.

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

2. [MSDN](#) — 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 «JavaScript»: «**RegExp msdn javascript**».

3. [Safari Developer Library](#) — 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> — 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> — информация о поддержке тех или иных возможностей и несовместимостях.

Для поиска можно пользоваться комбинацией «**quirksmode onkeypress**» в Google.

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

Спецификация (формальное описание синтаксиса, базовых объектов и алгоритмов) языка Javascript называется [ECMAScript](#).

Ее перевод есть на сайте в разделе [стандарт языка](#).



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 является самым фундаментальным, доверенным источником информации.

Мы живем во время, когда все быстро изменяется. Современный стандарт — это [ECMA-262 5.2](#) (или просто ES5), поддерживается всеми современными браузерами.

Не за горами — новая спецификация ES6, в которой предусмотрены еще много полезных возможностей, делающих разработку быстрее и веселее 😊

Спецификации HTML/CSS

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

Соответствующую информацию вы можете найти на сайте [w3.org](#). Там расположены стандарты HTML, CSS и многие другие.

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](#).

Последние версии стандартов расположены на домене [dev.w3.org](#).

Итого

Итак, посмотрим какие у нас есть источники информации.

Справочники:

- ⇒ [Mozilla Developer Network](#) — информация для Firefox и большинства браузеров.
Google-комбо: "RegExp MDC", ключевое слово «MDC».
- ⇒ [MSDN](#) — информация по IE.
Google-комбо: "RegExp msdn". Иногда лучше добавить термин «JavaScript»: "RegExp msdn javascript".
- ⇒ [Safari Developer Library](#) — информация по Safari.
- ⇒ <http://help.dottoro.com> — подробная информация по HTML/CSS/JavaScript с учетом 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 \(англ\)](#), и предыдущий [ES3 \(рус\)](#).
- ⇒ HTML/DOM/CSS — на сайте w3.org .
Google-комбо: "document.cookie site:w3.org".

То, как оно на самом деле работает и несовместимости:

- ⇒ Смотрите <http://www.quirksmode.org/>. Google-комбо: "innerHeight quirksmode".

См. также

- ⇒ [MSDN](#)
- ⇒ [Safari Developer Library](#)
- ⇒ [Mozilla Developer Network](#)
- ⇒ <http://help.dottoro.com>

Редакторы для кода

Есть два вида редакторов: IDE и «лёгкие».

Разница между ними — в том, что IDE загружает весь проект целиком, поэтому может предоставлять автодополнение по функциям всего проекта, удобную навигацию по его файлам и т.п.

Лёгкие редакторы — редактируют конкретный файл (или несколько) и знать не знают о связях между ними.

Некоторые IDE можно использовать как лёгкие редакторы, но обычно IDE сложнее, тяжелее и работают медленнее.

Обязательно нужен хороший редактор.

Тот, который вы выберете должен иметь в своем арсенале:

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

IDE

Если вы еще не задумывались над этим, присмотритесь к следующим вариантам.

- ⇒ Продукты IntelliJ: [WebStorm](#), а также в зависимости от дополнительного языка программирования [PHPStorm \(PHP\)](#), [IDEA \(Java\)](#) и другие.
- ⇒ Visual Studio, в сочетании с разработкой под .NET (Win)
- ⇒ Продукты на основе Eclipse, в частности [Aptana](#) и Zend Studio
- ⇒ [Komodo IDE](#) и его облегчённая версия [Komodo Edit](#).
- ⇒ [Netbeans](#)

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. Но их стоимость невелика, по сравнению с зарплатой веб-разработчика, поэтому ориентироваться можно на удобство.

Лёгкие редакторы

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

Как правило, под IDE понимают мощный редактор, с упором на проекты. А «лёгкие» редакторы предназначены в первую очередь для редактирования отдельных файлов. Но на практике граница между IDE и «лёгким» редактором может быть размыта, и спорить что именно редактор, а что IDE — не имеет смысла.

Достойны внимания:

- [Sublime Text](#) (кросс-платформенный, платный).
- TextMate (Mac, платный)
- [SciTe](#) простой, легкий и очень быстрый (Windows, бесплатный).
- [Notepad++](#) (Windows, бесплатный).
- Vim, Emacs. Если умеете их готовить.

Выберите любой редактор из перечисленных выше, главное чтобы он что-то умел, кроме простого блокнота.

Топ 3

Лично мои любимые редакторы:

- [Sublime Text](#).
- Редакторы от JetBrains: [WebStorm](#), а также в зависимости от дополнительного языка программирования [PHPStorm \(PHP\)](#), [IDEA \(Java\)](#) и другие.

→ Visual Studio, если разработка идёт под платформу .NET (Win)

Если не знаете, что выбрать — можно посмотреть на них 😊

P.S.

В списках выше перечислены редакторы, которые используют многие 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.

Sublime Text: шпаргалка

Одним из наиболее мощных, и при этом простых, редакторов является [Sublime Text](#).

В нём хорошо развита система команд и горячих клавиш. На этой странице размещена «шпаргалка» с плагинами и самыми частыми сочетаниями клавиш, которые сильно упрощают жизнь.

Горячие клавиши

Для наибольшего удобства «шпаргалка» должна быть распечатана и повешена перед глазами, поэтому она сделана в виде 3-колоночного PDF.

[Скачать шпаргалку в формате PDF](#)

Шпаргалка пока под Mac. Для Windows сочетания похожи, обычно вместо Mac-клавиши `Cmd` под Windows будет `Ctrl`. А если в сочетании есть и `Cmd` и `Ctrl`, то под Windows будет `Ctrl` + `Shift`.

Вы часто используете сочетание, но его нет в списке? Поделитесь им в комментариях!

Плагины

Мои любимые плагины:

- Package Control
- sublime-emmet
- JsFormat
- SideBarEnhancements
- AdvancedNewFile
- sublime-jsdocs
- SublimeCodeIntel

Остальные:

- Alignment
- CSSComb
- EncodingHelper
- GoToRecent
- HTML5
- jQuery
- Prefixr
- View In Browser

Чтобы узнать о плагине подробнее — введите его название в Google.

Есть и другие хорошие плагины, кроме перечисленных. Кроме того, Sublime позволяет легко писать свои плагины и команды.

Установка браузеров, JS-консоль

Мы будем писать скрипты, которые поддерживают все современные браузеры. Хотя они и стремятся поддерживать стандарты, но все-таки бывают отличия.

Как минимум, вам потребуются: [Firefox](#), [Chrome](#) и Internet Explorer.

Если вы работаете в Linux или MacOS, то вам потребуется виртуальная машина с Windows для IE.

Большинство разработчиков сначала пишут скрипты под Firefox или Chrome. Если все работает, скрипт тестируется в остальных браузерах.

Firefox

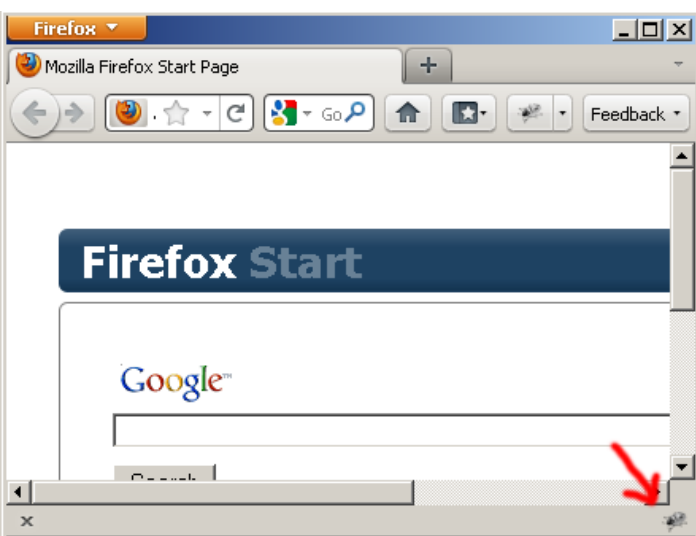
Для разработки в Firefox используется расширение Firebug. Его нужно поставить после установки браузера.

Установка Firebug

Поставьте его со страницы:

1. <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 ее показа.

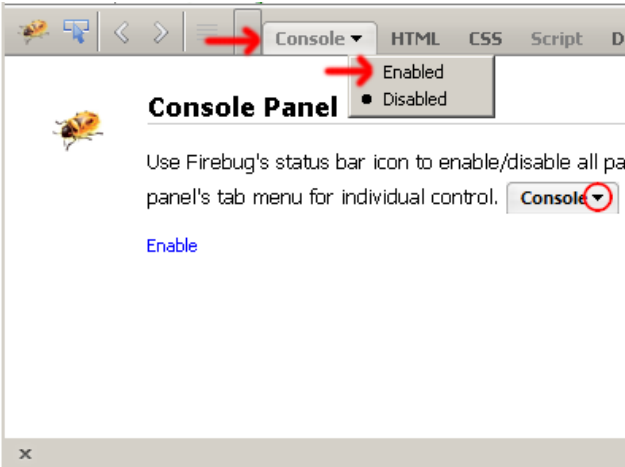
Ну а если ее нет и там, то нажмите `F12` — это горячая клавиша для запуска Firebug, мышкой его обычно никто не запускает.

Включите консоль

Итак, откройте Firebug. Здесь иллюстрации на английском, русский вариант аналогичен.

Консоль вначале выключена. Нужно включить её в меню

Консоль -> панель включена:

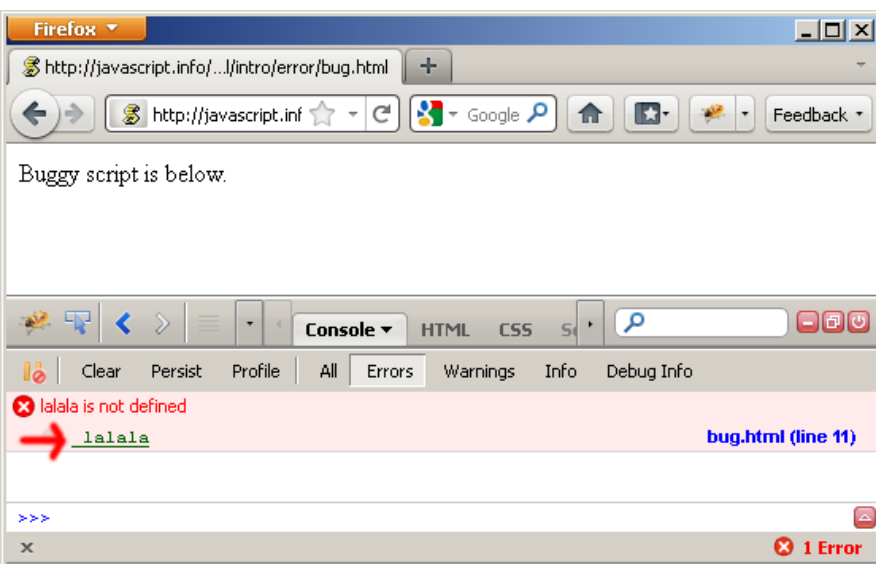


Просмотр ошибок

С открытым Firebug зайдите на страницу с ошибкой: bug.html.

Вы можете посмотреть её исходный код, нажав `Ctrl` + `U`.

Консоль покажет ошибку:



В данном случае код `lalala` непонятен интерпретатору и вызвал ошибку.

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

Об основных возможностях можно прочитать на сайте firebug.ru.

Internet Explorer

В IE начиная с версии 8 (а лучше 9) есть похожий отладчик. По умолчанию он отключен.

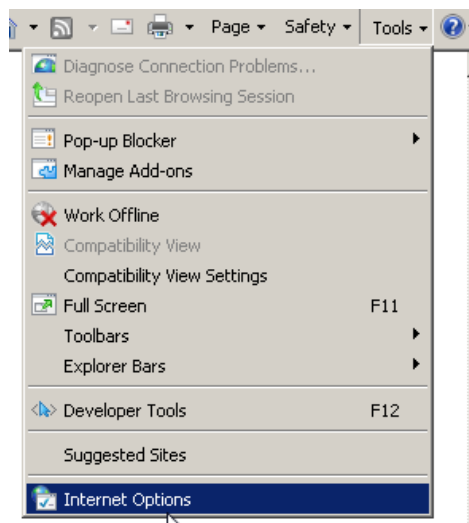
Включаем отладку

Зайдите в меню.

IE 8

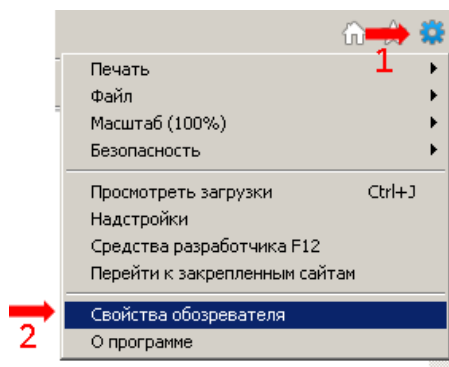
Tools -> Internet Options (рус.

Инструменты -> Свойства обозревателя)



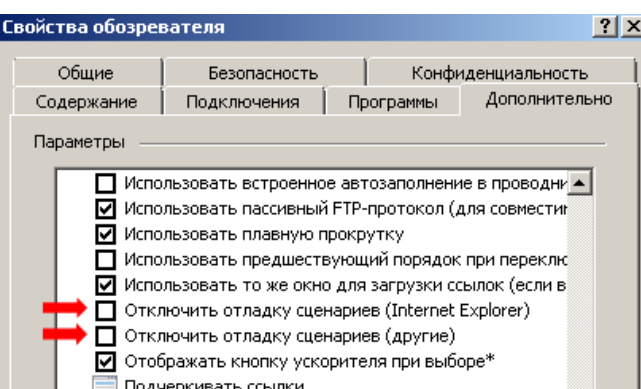
IE 9

Колесико в правом-верхнем углу - Свойства обозревателя:



Переключитесь во вкладку Дополнительно и прокрутите вниз, пока не увидите две галочки, которые начинаются с Отключить отладку сценариев.

По умолчанию они отмечены. Снимите с них отметку:

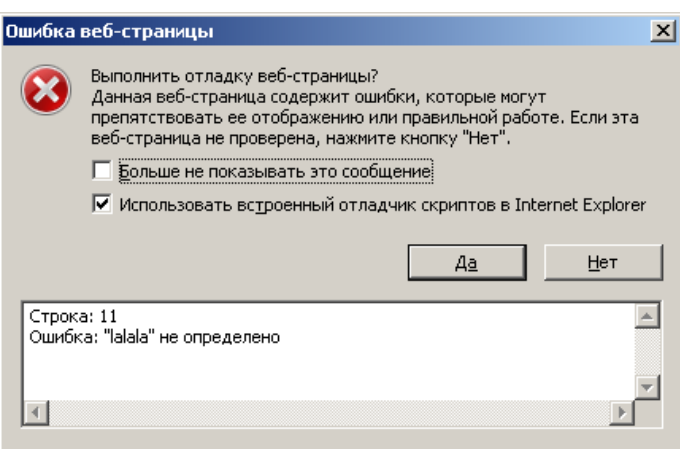


Перезапустите браузер.

Просмотр ошибок

Зайдите на страницу с ошибкой: bug.html.

Появится окно с предложением начать отладку. Нажмите «Да» — и вы в отладчике.



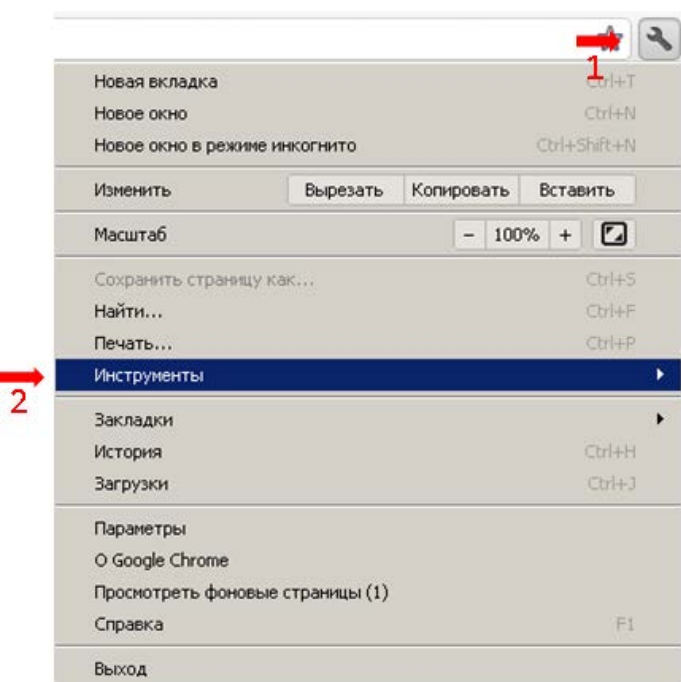
Теперь отладчик вместе с другими инструментами разработки доступен по кнопке **F12**.

Другие браузеры

Google Chrome

Горячие клавиши: **Ctrl+Shift+I**, **Ctrl+Shift+J**.

Меню Инструменты -> Инструменты разработчика:



Safari

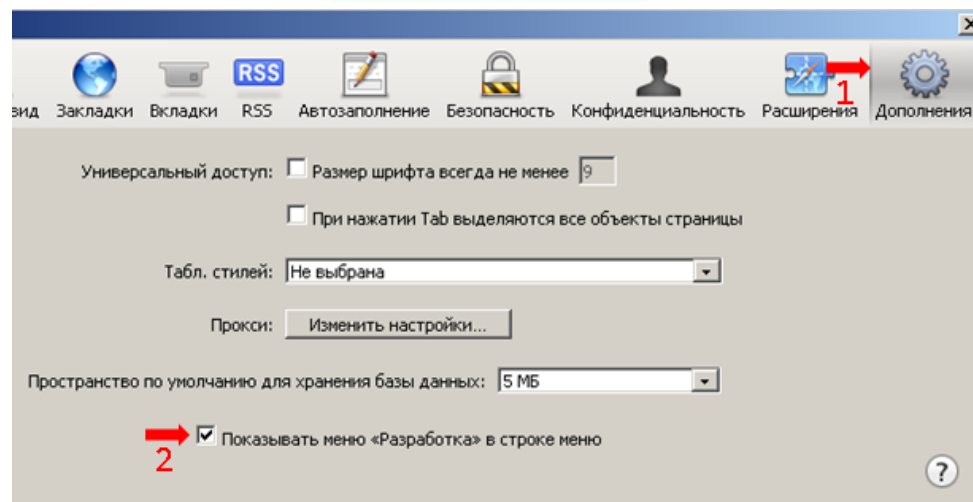
Горячие клавиши: **Ctrl+Shift+I**, **Ctrl+Alt+C**.

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.

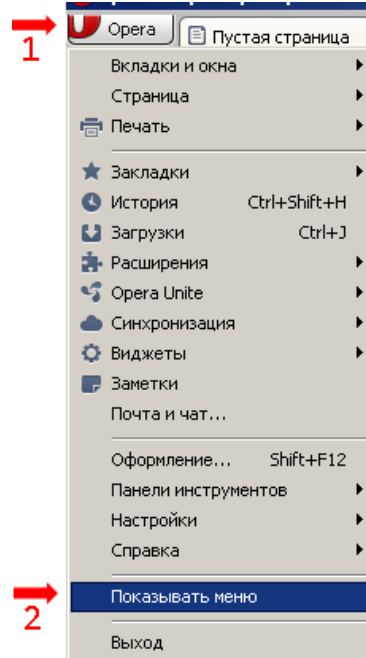
2. Нажмите на колесико и выберите **Показать строку меню**.

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

Opera

В Opera работает горячая клавиша **Ctrl+Shift+I**.

Можно включить и доступ через меню.



Для этого сначала нужно включить меню:

Теперь в меню: Инструменты -> Дополнительно -> Opera Dragonfly.
Вы на месте.

IE<8

Для IE<8, основной инструмент разработки - это Microsoft Script Debugger. У него есть 4 варианта, самый лучший входит в поставку Microsoft Visual Studio (платной, не Express).

При установке студии отключите все дополнительные опции, чтобы не ставить лишнего.

Также есть [Internet Explorer Developer Toolbar](#) для работы с документом. Она вам понадобится для поддержки IE7 и, возможно, IE6.

Тестирование в старых браузерах

Бывает, что ошибка возникает в старой версии популярного браузера, но она настолько важна, а посетителей с этой версией так много, что её хорошо бы поправить.

Особенно это касается IE, у которого между версиями самые существенные различия и, в меньшей степени, других браузеров.

Здесь мы рассмотрим, как поставить несколько версий различных

браузеров одновременно, для разработки и отладки.

Internet Explorer

Браузер Internet Explorer тесно интегрирован с системой. Особенно это касается IE<9.

Идёт это всё с древних времён, когда Микрософт обвиняли в монополизме и пытались заставить изъять ряд компонентов, в том числе браузер, из операционной системы. А Микрософт делать этого не хотела, аргументируя тем, что «браузер слишком сильно интегрирован в ОС, является её частью и изъять его ну никак нельзя». И, соответственно, старалась интегрировать 😊

Варианты разработки и тестирования под несколько версий IE:

Режим эмуляции в IE8+

В IE8+ есть режим эмуляции, переключается в панели инструментов, но работает он неполноценно.

По опыту, он на 100% надёжен в вёрстке, но примерно в 95% в JavaScript. Некоторые ошибки не воспроизводятся.

IETester

Один из лучших проектов по эмуляции старых IE. Содержит библиотеки от них, по возможности старается отделить один IE от другого, но это не всегда удаётся.

Тоже хорошо работает с вёрсткой, но в JavaScript не все глюки и особенности IE воспроизводятся, и кроме того, появляются свои, дополнительные, ошибки.

Виртуальная машина, например [VirtualPC](#), [VMware](#), [VirtualBox](#), [Parallels \(Mac\)](#)

В виртуальную машину ставится нужная версия Windows с соответствующим IE.

Это — единственный полностью надёжный способ.

Также существуют утилиты, позволяющие запускать приложение без явного входа в виртуальную машину, на одной панели задач с другими.

Как будто оно установлено в основную ОС. Например [ThinApp](#) , [Sandboxie](#). Можно ими пользоваться, но возможны неудобства, если к браузеру ставятся плагины, отладчик Visual Studio (IE6,7), профайлер (DynaTrace AJAX) и т.п.

Можно использовать несколько подходов одновременно: режим эмуляции при разработке, а при необходимости или для 100%-й гарантии — запускать виртуальную машину с нужной версией IE.

Firefox

С браузером Firefox проще, но иногда и между его версиями бывают существенные различия. К счастью, Firefox можно поставить в нескольких экземплярах на одну машину. Проблем не будет.

Обычно поддерживаются две версии Firefox: «текущая стабильная (или даже бета)» и «минимально допустимая». При этом в первой включено автообновление, а во второй — нет.

Единственное, что следует иметь в виду — это «профиль», т.е. директорию, в которой браузер хранит свои настройки и расширения. Физически он хранится в домашней директории пользователя. Под разные Firefox они могут быть разными, поэтому нужно указывать профиль явно.

Разным версиям Firefox должны соответствовать разные профили.

1. Профиль нужно сначала создать. Для этого при запуске Firefox используйте ключ:

```
firefox.exe --profilemanager
```

Запуск можно осуществлять из директории установки или по полному пути к Firefox, чтобы это точно была нужная версия.

Firefox выведет список профилей и позволит создать новый.

2. В дальнейшем создаётся «ярлык» (или его аналог, bash-файл и т.п.), явно запускающий Firefox с нужным профилем:

C:\Program Files\FirefoxLatest\firefox.exe -P latest

3. Чтобы запускать несколько версий Firefox одновременно, нужно добавить опцию `--no-remote`.

Chrome

Как правило, хватает одной — последней версии Chrome. Тем не менее, иногда хочется, например, иметь последнюю `stable`-версию и `dev` для тестирования самого нового функционала.

Под MacOS/Linux — возможно, всё совсем просто (напишите в комментариях, как?), а под Windows я использую [Chrome Portable](#).

Opera

Разные «мажорные», то есть отличающиеся главных цифрах (не после точки) версии Opera, как правило, могут жить на одной машине. В качестве альтернативы можно использовать [Opera Portable](#).

Привет, мир!

В этой статье мы создадим простой скрипт и посмотрим, как он работает.

Тег SCRIPT

Программы на языке JavaScript можно вставить в любое место HTML при помощи тега `SCRIPT`. Например:

```
01 <!DOCTYPE HTML>
02 <html>
03 <head>
04   <!-- Тег meta для указания кодировки -->
05   <meta charset="utf-8">
06 </head>
07 <body>
08
09   <p>Начало документа...</p>
10
11   <script>
```

```
12     alert('Привет, Мир!');
13     </script>
14
15     <p>...Конец документа</p>
16
17 </body>
18 </html>
```

Этот пример использует следующие элементы:

`<script> ... </script>`

Тег `script` содержит исполняемый код. Предыдущие стандарты HTML требовали обязательного указания атрибута `type`, но сейчас он уже не нужен. Достаточно просто `<script>`.

Браузер, когда видит `<script>`:

1. Начинает отображать страницу, показывает часть документа до `script`
2. Встретив тег `script`, переключается в JavaScript-режим и не показывает, а исполняет его содержимое.
3. Закончив выполнение, возвращается обратно в HTML-режим и отображает оставшуюся часть документа.

Попробуйте этот пример в действии, обратите внимание что **пока браузер не выполнит скрипт - он не может отобразить часть страницы после него.**

`alert(...)`

Отображает окно с сообщением и ждет, пока посетитель не нажмет «Ок»



Кодировка и тег META

При попытке сделать такой же файл у себя на диске и запустить, вы можете столкнуться с проблемой — выводятся «кракозяблы», «квадратики» и «вопросики» вместо русского текста.

Чтобы всё было хорошо, нужно:

1. Убедиться, что в HEAD есть строка `<meta charset="utf-8">`.
Если вы будете открывать файл с диска, то именно он укажет браузеру кодировку.
2. Убедиться, что редактор сохранил файл в кодировке UTF-8, а не, скажем, в windows-1251. На английском соответствующий параметр может называться «charset» или «encoding».

Указание кодировки — часть обычного HTML, к JavaScript не имеет отношения.

Очень важно не только читать, но и тестировать, пробовать писать что-то самому.

Решите задачу, чтобы удостовериться, что вы все правильно поняли.



Задача: Alert

Сделайте страницу, которая выводит «Я - JavaScript!», т.е. работает вот так:

Важность: 5

<http://learn.javascript.ru/files/tutorial/browser/script/alert/index.html>.

Создайте ее на диске, откройте в браузере, убедитесь, что все работает.

[Решение задачи "Alert" »»](#)



Современная разметка для тега SCRIPT

В старых скриптах оформление тега SCRIPT было немного сложнее. В них можно встретить следующие элементы:

Атрибут `<script type=...>`

В отличие от HTML5, стандарт HTML 4 требовал обязательного указания этого атрибута. Выглядел он так: `type="text/javascript"`.

Если вы укажете некорректные данные в атрибуте `type`, например `<script type="text/html">`, то содержимое тега не будет отображено. Но его можно получить средствами

JavaScript. Этот хитрый способ используют для добавления служебной информации на страницу.

Атрибут `<script language=...>`

Этот атрибут ставить не обязательно, т.к. язык по умолчанию — JavaScript.

Комментарии до и после скриптов

В старых руководствах и книгах иногда рекомендуют использовать HTML-комментарии внутри SCRIPT, чтобы спрятать Javascript от браузеров, которые не поддерживают его.

Выглядит это примерно так:

```
<script type="text/javascript"><!--  
...  
//--></script>
```

Браузер, для которого предназначались такие трюки, очень старый Netscape, давно умер. Поэтому в этих комментариях нет нужды.

Внешние скрипты

Если JavaScript-кода много — его выносят в отдельный файл, который подключается в HTML:

```
<script src="/path/to/script.js"></script>
```

Здесь `/path/to/script.js` - это абсолютный путь к файлу, содержащему скрипт (из корня сайта).

Браузер сам скачает скрипт и выполнит.

Например:

```
01 <html>  
02   <head>  
03     <meta charset="utf-8">  
04     <script
```

```
src="/files/tutorial/browser/script/rabbits.js"></script>
05 </head>
06
07 <body>
08   <script>
09     count_rabbits();
10   </script>
11 </body>
12
13 </html>
```

Содержимое файла /files/tutorial/browser/script/rabbits.js:

```
1 function count_rabbits() {
2   for(var i=1; i<=3; i++) {
3     alert("Кролик номер "+i)
4   }
5 }
```

Можно указать и полный URL, например:

```
<script src="http://code.jquery.com/jquery.js"></script>
```

Вы также можете использовать путь относительно текущей страницы, например `src="script.js"` если скрипт находится в том же каталоге, что и страница.

Чтобы подключить несколько скриптов, используйте несколько тегов:

```
<script src="/js/script1.js"></script>
<script src="/js/script2.js"></script>
...
```



Как правило, в HTML пишут только самые простые скрипты, а сложные выносят в отдельный файл.

Благодаря этому один и тот же скрипт, например, меню или библиотека функций, может использоваться на разных страницах.

Браузер скачает его только первый раз и в дальнейшем, при правильной настройке сервера, будет брать из своего **кэша** .



Если указан атрибут `src`, то содержимое тега игнорируется.

В одном теге `SCRIPT` нельзя одновременно подключить внешний скрипт и указать код.

Вот так не работает:

```
<script src="file.js">
  alert(1); // если указан src, то внутренняя часть
            скрипта игнорируется
</script>
```

Нужно выбрать: либо `SCRIPT` идёт с `src`, либо содержит код. Тег выше следует разбить на два: один — с `src`, другой с кодом:

```
1 <script src="file.js"></script>
2 <script>
3   alert(1);
4 </script>
```

Решения задач



Решение задачи: Alert

<http://learn.javascript.ru/play/tutorial/browser/script/alert/index.html>.