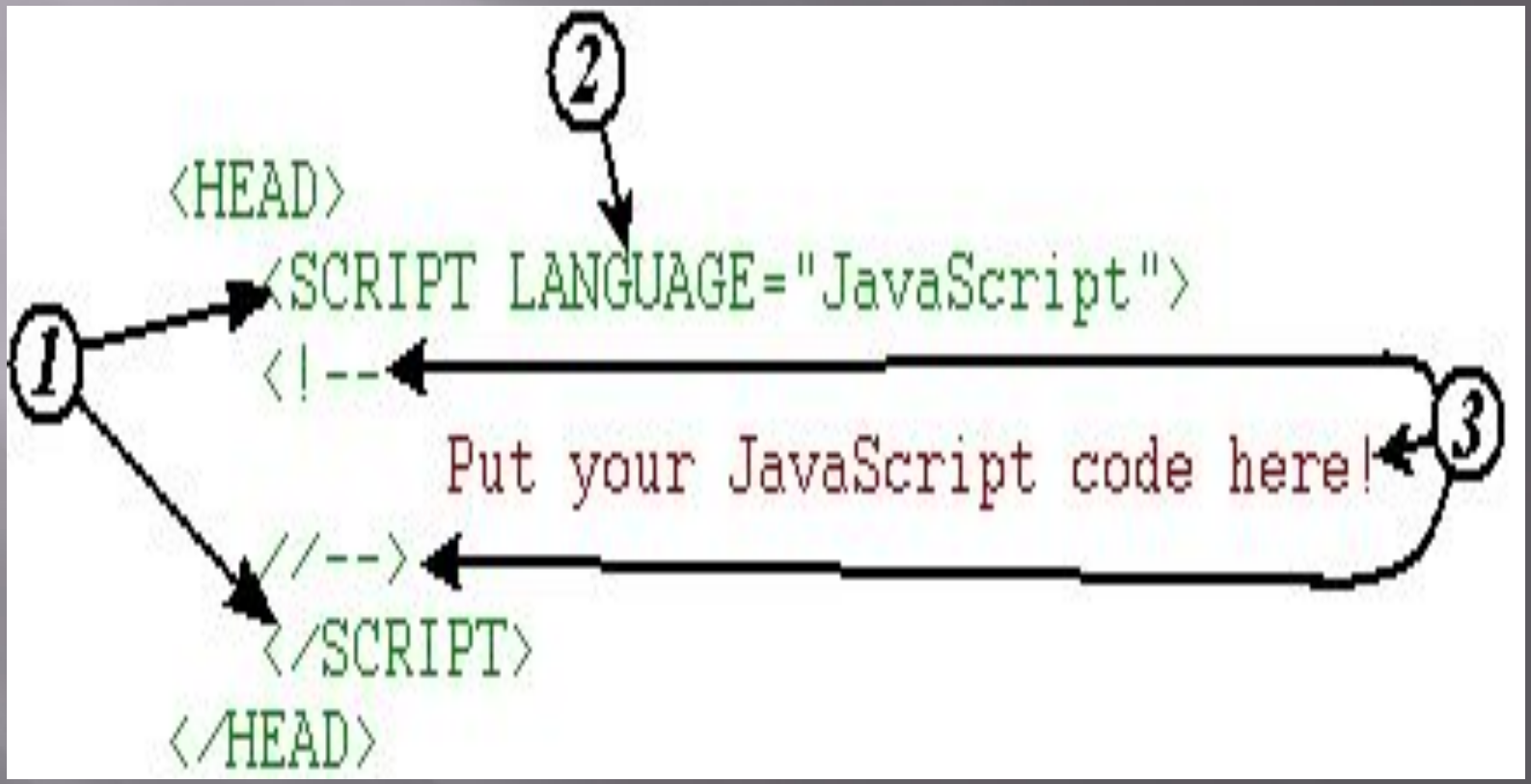
ИСТОРИЯ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ: РАЗБРОД И КОНСОЛИДАЦИЯ JAVASCRIPT

Автор: Георгий Аодония
11в класс.


2
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
1
<!--
Put your JavaScript code here!
3
//-->
</SCRIPT>
</HEAD>

При генерации страниц в Web возникает дилемма, связанная с архитектурой «клиент-сервер». Страницы можно генерировать как на стороне клиента, так и на стороне сервера. В 1995 году специалисты компании Netscape создали механизм управления страницами на клиентской стороне, разработав язык программирования JavaScript.

Таким образом, JavaScript — это язык управления сценариями просмотра гипертекстовых страниц Web на стороне клиента. Однако на самом деле JavaScript — это не только язык для программирования на стороне клиента. Liveware, прародитель JavaScript, является средством подстановок на стороне сервера Netscape. Тем не менее, наибольшую популярность JavaScript обеспечил front-end.

Основная идея JavaScript состоит в том, чтобы изменять отдельные значения атрибутов HTML-контейнеров и свойств среды отображения в процессе просмотра HTML-страницы пользователем. При этом для актуализации изменений перезагрузка страницы не требуется.

Netscape Communications Corporation представляет

Компания **Netscape Communications Corporation** активно участвовала в процессе развития всемирной паутины. Последняя явно многим обязана этой компании: среди прочего **Netscape** подарила вебу **JavaScript**. Изначально компании удалось достойно вступить в борьбу за первенство в интернет-отрасли благодаря созданию и бесплатному распространению (для использования в домашних условиях) браузера **Netscape Navigator**.

Но в апреле **1995** года **Netscape** наняла Брендона Эйха, на которого была возложена особая миссия. Перед ним стояла задача внедрить язык программирования **Scheme** (или что-то похожее) в браузер **Netscape Navigator**.

Scheme — это функциональный язык программирования, один из двух наиболее популярных в наши дни диалектов языка Лисп (другой популярный диалект — это **Common Lisp**).

Однако постановка задачи была, мягко говоря, не слишком точна, Эйха перевели в группу, ответственную за серверные продукты, где он проработал месяц, занимаясь улучшением протокола **HTTP**. В мае разработчик был переброшен обратно, в команду, занимающуюся клиентской частью (браузером), где он немедленно начал разрабатывать концепцию нового языка программирования.

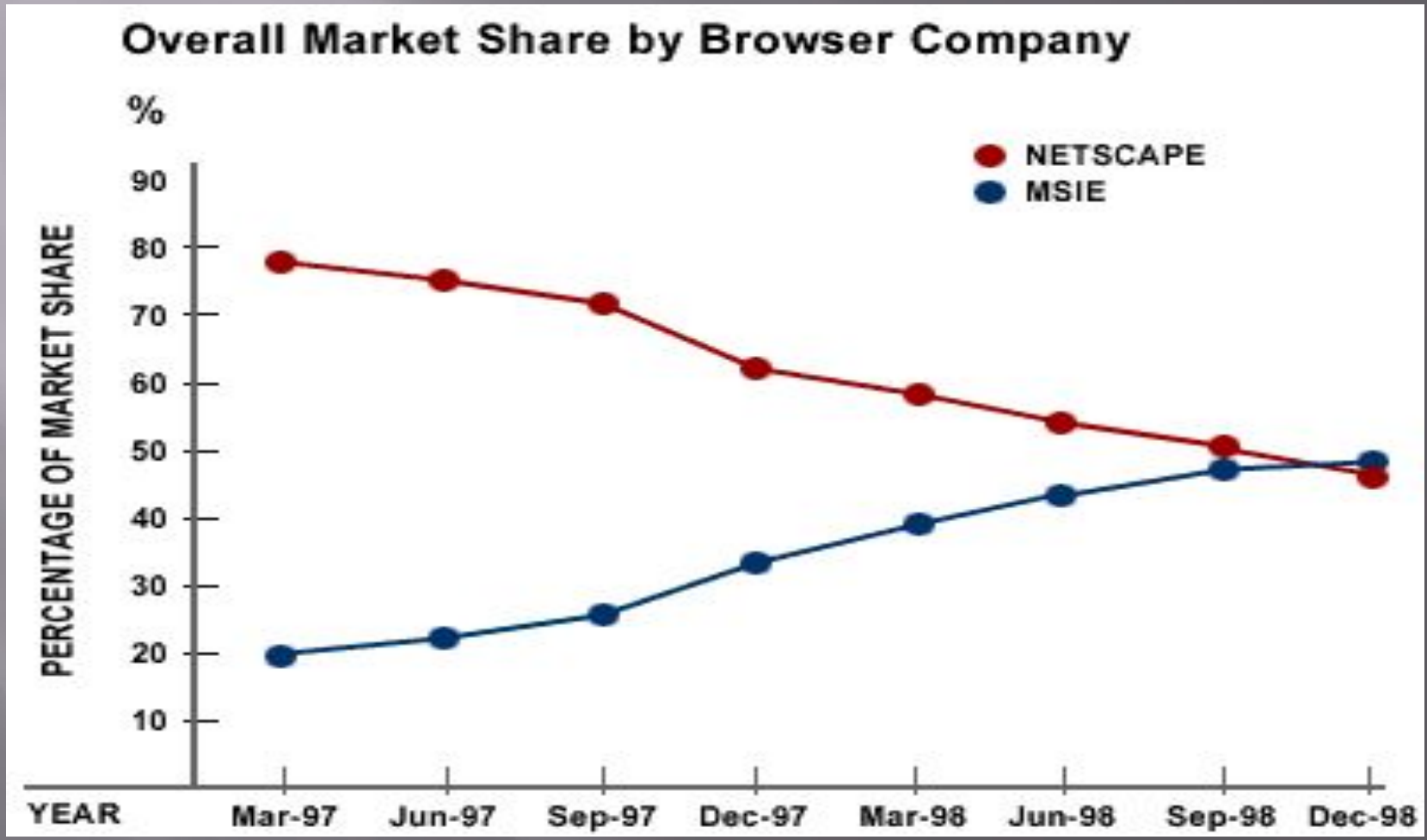
Через некоторое время появился скриптовый язык под названием **LiveScript**. Этот язык был создан для реализации интерактивности в **HTML**-документах, которые прежде были статичными. Поддержка **LiveScript** была реализована в первых версиях браузера **Netscape Navigator**, пользовался большой популярностью и успехом.

Помимо Брендана Эйха в разработке нового языка участвовали сооснователь **Netscape Communications** Марк Андрессен и сооснователь **Sun Microsystems** Билл Джой. Чтобы успеть закончить работы над языком к релизу браузера, компании заключили соглашение о сотрудничестве. Их целью было создать «язык для склеивания» составляющих частей веб-ресурса: изображений, плагинов, **Java**-апплетов, который был бы удобен для веб-дизайнеров и программистов, не обладающих высокой квалификацией.

В результате соглашения между **Netscape Communications** и **Sun Microsystems** и объединения идей **LiveScript** со структурой **Java** появилась среда под названием «**Mocha**», предназначенная для разработки сетевых приложений и, в конце концов, для создания динамичных **web**-страниц. Среда выпускалась как открытое ПО и была независима от используемой программной платформы.

Проект завершился созданием спецификаций, которые были опубликованы двумя компаниями в декабре 1995 года под названием **JavaScript 1.0**.

Netscape vs Microsoft vs стандартизация



Первым браузером, поддерживающим **JavaScript**, был **Netscape Navigator 2.0**. Однако корпорация **Microsoft** быстро сообразила, куда ветер дует и разработала свой «**JavaScript**», который получил название **JScript 1.0**. Естественно, его поддержка была реализована в браузере **Microsoft Internet Explorer 3.0** и **Internet Information Server**.

Несмотря на то, что **JScript** формально был независимой разработкой **Microsoft**, он оказался совместимым с **JavaScript 1.0** компании **Netscape**. Более того, сценарий, написанный для одного браузера, с большой вероятностью мог быть выполнен на другом браузере.

Позже компания **Netscape** выпустила версию **JavaScript 1.1** для **Netscape Navigator 3.0** и **Live Wire Web server**. В данной версии были сохранены все характерные черты языка **JavaScript 1.0** и добавлено множество новых возможностей.

Компания **Microsoft** также усовершенствовала собственный язык **JScript**, но решила не включать в него все нововведения **JavaScript 1.1**. С этого момента возникла несовместимость браузеров: при попытке запустить сценарии, написанные на **JavaScript 1.1**, они не распознавались или приводили к ошибкам при использовании в продуктах **Microsoft**.

Однако вскоре **Netscape**, **Microsoft** и другие компании решили, что будет лучше мирно договориться и выработать единый стандарт. Европейская ассоциация производителей компьютеров (**European Computer Manufacturing Association** — ECMA) начала работу над ним в ноябре **1996** года. В июле следующего года был создан новый язык, получивший название **ECMAScript**.

А пока шла работа над стандартом, конкуренты не теряли времени и выпустили новые версии собственных языков – **JavaScript 1.2** для браузера **Netscape Navigator 4.0**, и **JScript 2.0** для браузера **Microsoft Internet Explorer 3.0**. Таким образом проблемы совместимости языков выполнения сценариев еще больше усиливались: при доработке этих языков не был учтен общий стандарт (ECMA).

Жизнь **Web**-разработчиков сильно осложнилась. Им не только пришлось запоминать особенности каждого браузера, но и создавать **Web**-страницы, которые можно было бы просматривать в обоих браузерах.

Большинство из них было уверено, что ситуация никогда не изменится и будет только усугубляться. Однако оптимисты надеялись, что благодаря ECMA браузеры снова станут совместимыми.
Конкурирующие компании прекратили упрямиться только к выходу третьей редакции стандарта ECMA 262 (ECMAScript Edition 3) и выпустили JavaScript 1.5, и JScript 5.5. Эти версии были практически на 100% совместимы с ECMAScript Edition 3.

После этого стало возможно написать сценарий JavaScript, который мог бы одинаково хорошо работать в обоих браузерах. Теоретически. Однако различия между браузерами все равно осложняли эту задачу

Таблица 1.1. Поддержка JavaScript в браузерах Netscape

Браузер	Версия
Netscape 2.0	1.0
Netscape 3.0	1.1
Netscape 4.0	1.2
Netscape 4.5	1.3/1.4
Netscape 6.0	1.5

Таблица 1.2. Поддержка JavaScript в браузерах Internet Explorer

Браузер	Версия
Internet Explorer 3.0	1.0/2.0
Internet Explorer 4.0	3.0
Internet Explorer 5.0	5.0
Internet Explorer 5.5	5.5

Тем не менее, JavaScript нашел широкое применение не только у разработчиков браузеров, но и в целом у создателей открытого ПО.

По данным Black Duck Software, в разработке открытого программного обеспечения доля использования JavaScript росла. 36 % проектов, релизы которых состоялись с августа 2008 по август 2009, написаны с использованием JavaScript

JavaScript сегодня

Aug 2016	Aug 2015	Change	Programming Language	Ratings	Change
1	1		Java	19.010%	-0.26%
2	2		C	11.303%	-3.43%
3	3		C++	5.800%	-1.94%
4	4		C#	4.907%	+0.07%
5	5		Python	4.404%	+0.34%
6	7	^	PHP	3.173%	+0.44%
7	9	^	JavaScript	2.705%	+0.54%
8	8		Visual Basic .NET	2.518%	-0.19%
9	10	^	Perl	2.511%	+0.39%
10	12	^	Assembly language	2.364%	+0.60%
11	14	^	Delphi/Object Pascal	2.278%	+0.87%
12	13	^	Ruby	2.278%	+0.86%
13	11	v	Visual Basic	2.046%	+0.26%
14	17	^	Swift	1.983%	+0.80%
15	6	vv	Objective-C	1.884%	-1.31%
16	37	^^	Groovy	1.637%	+1.27%
17	20	^	R	1.605%	+0.60%
18	15	v	MATLAB	1.538%	+0.31%
19	19		PL/SQL	1.349%	+0.21%

Согласно **TIOBE Index**, базирующемуся на данных поисковых систем **Google**, **MSN**, **Yahoo!**, Википедия и **YouTube**, в августе 2016 года **JavaScript** находился на 7-ом месте. Год назад он занимал 9-ю позицию.

80 % открытого программного обеспечения написано на Си, C++, **Java**, **Shell** и **JavaScript**. При этом **JavaScript** — единственный из этих языков, чья доля использования увеличилась год к году (более чем на 2 процента, если считать в строках кода).

JavaScript является самым популярным языком программирования, используемым для разработки веб-приложений на стороне клиента.

Кроме того, **JavaScript** активно применяется в следующих направлениях разработки:

- Бэкенд
- Мобильные приложения
- Десктоп приложения
- **Embedded**. Холодильники/часы/чайники/**IoT**

В настоящее время **JavaScript** является наиболее популярным языком программирования с прекрасно развитой экосистемой модулей **OSS**, который, в отличие от других существующих альтернатив, действительно можно назвать универсальным: «пишешь один раз, используешь везде» — идея, о которой создатели **Java** мечтали. И вот благодаря **JavaScript** она стала реальностью, писал Эрик Эллиот.

ПЕРСПЕКТИВЫ JAVASCRIPT

Нет такого языка, или технологии, которые были бы однозначно признаны рынком как лучшее решение для разработчиков в какой-либо сфере. У каждого варианта есть свои достоинства и недостатки.

Сложность современных веб-решений давно требует существенного пересмотра. Поэтому большое внимание разработчики уделили новым версиям стандарта JavaScript – ECMAScript 6 и 7.

6-я версия стандарта (который, кстати, в пику несостоявшемуся выпуску ES4 иногда называют как ES6 Harmony) содержит изменения, 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.

В отличие от ECMAScript 6 спецификация ECMAScript 7 содержит относительно немного изменений, которые развивались в рамках непрерывно обновляемого варианта спецификации ECMAScript Next. В стандарт из данной черновой спецификации были перенесены уже поддерживаемые браузерами возможности, поэтому ECMAScript 7 сразу доступен во всех основных браузерах и не требует дополнительного времени на реализацию.

В ECMAScript 7 вошли изменения, связанные с устранением недоработок и внесением уточнений к ECMAScript 6.

JavaScript по-прежнему остается перспективным языком программирования. О его перспективах достаточно много уже было написано на Хабре

1. Рост использования TypeScript в реальных проектах, развитие альтернативных проектов и их взаимное обогащение.
2. Развитие инструментов для кроссплатформенной разработки на JS, продолжение стирания границ между сайтами и приложениями.
3. Рост умных телевизоров и консолей с разработкой на JavaScript, нативная разработка на JS на многих современных платформах (но не всех).
4. Развитие API доступа к нативным возможностям устройства из JavaScript, адаптация NUI в JS. (Затянется на несколько лет.)
5. Новые переработанные версии популярных библиотек, повышение входного порога для создания комплексных фреймворков, нишевые решения на базе ES6.
6. Адаптация веб-компонент браузерами, принятие новых технологий разработчиками элементов управления и различных фреймворков.
7. Применение менеджеров пакетов и систем сборки для JavaScript в корпоративной и учебной среде, интеграция в популярные инструменты веб-разработки.
8. Развитие графических библиотек на JS, показательная адаптация новых технологий крупными или заметными игроками рынка (игры и интерактивный контент — основные драйверы).
9. Unity 5 с рендерингом в WebGL, развитие 3d и игровых библиотек, потенциальный прорыв через социальные сети.
10. Облачные решения для IoT на базе Node.js, новые экспериментальные проекты на клиентской стороне.

Конец
спасибо за внимание)))