Содержание

Введение 8

Будущее www – какое оно? 9

История вопроса 11
Актуальность стандарта 11
XHTML – стандарт для стандартизаторов 11
За пределы разметки – ActiveX, Java, Flash 14
Рождение HTML5 16

RIA 18
Microsoft Silverlight 19
Adobe Flex 21
JavaFX 23
Google Native Client 25
RIA и HTML5 26

HTML5 сегодня 27
О браузерах 28
Проверять возможности, а не версии 28
Moz-, webkit- и прочие vender-грабли 29
Modernizr – бархатный путь в HTML5 31

HTML – это теги 37
Структура страницы 37
Воплощение концепции семантической разметки 39

Всякие полезности 44
Атрибуты и аксессуары 48
Атрибуты data-* 52
Отречемся от старого мира (что выбросили, что изменили) 53

HTML5-формы – о чем мы мечтали 57
Новые поля ввода 57
INPUT... а OUTPUT? 64
Не только разметка – объект ValidityState и другие 66

HTML5 Accessibility – всего лишь доступность (ARIA, WCAG) 70
WCAG – рекомендации, которые никто не слушал 70
WAI-ARIA – перманентно временное решение, которое работает 73
Проблемы доступности 74
ARIA-роли 75
Абстрактные роли (Abstract Roles) 75
Роли – виджеты (Widget Roles) 76
Роли структуры документа (Document Structure Roles) ... 77
Роли разметки (Landmark Roles) 78
Состояния и свойства объектов – ARIA-атрибуты 79
Атрибуты виджетов 79
Атрибуты для Live Region 79
Атрибуты перетаскивания (Drag-and-Drop) 80
Атрибуты отношений 80
Применение WAI-ARIA 82

Web с нечеловеческим лицом (микроформаты и микроданные) 83
Когда тегов не хватает 83
Микроформаты 84
Технология RDFa 87
Микроданные 91
Microdata DOM API 98

Canvas – холст для рисования на веб-странице 99
Черный ректангл 99
Использование примитивов 101
Начинаем рисовать 105
Работа с изображениями 113
За каждый пиксель! 119
Трансформации 121
Интерактивность и анимация 124
Свой Paint 124
Как нам организовать анимацию? 126
Play the Game! 129
Библиотеки для работы с Canvas 132

SVG – векторная графика в www 135
Рисуем тегами 135
Кривая вывезет 140
Группируй и властвуй 145
Усложняем жизнь – элементы symbol и image 147
SMIL – язык анимации SVG 149
Библиотеки для работы с SVG 152
Canvas vs SVG 153

WebGL – врываемся в третье измерение 155
Браузеры и драйверы 155
Шейдеры 161
Наконец-то 3D 166
Теперь в движении 168
Объем 170
Текстура и освещение 173
Инструментарий для работы с WebGL 181

Храним данные на клиенте – WebStorage/WebSQL/WebNoSQL 186
WebStorage – хранилище «ключ–значение» в браузере 186
WebSQL – реляционная база данных на веб-странице 187
IndexedDB – NoSQL в вебе 189

AppCache – управляем кэшированием вплоть до полного offline! ... 195

File, FileSystem и полный drag'n'drop ... 198
File API – Ура! Свершилось! ... 198
FileSystem API ... 202
Все это drag'n'drop! ... 207

Сервер, я здесь ... 217
Server-Sent Events – сервер тебя не оставит ... 218
Web Messaging – легальный XSS ... 221
XMLHttpRequest 2 ... 225

Звуки audio ... 229
MediaElement – медиаплеер на HTML ... 231
WebAudioAPI ... 236

Video ... 242

WebRTC – коммуникации через веб-страницу ... 245

Geolocation API. Непростой вопрос собственного местонахождения ... 251
Где это я? ... 251
Позиционируем с помощью Google Maps ... 253
Откуда? ... 254
Вглубь Geolocaton API ... 255

WebWorkers – судьба сетевого пролетариата ... 258
Параллельные вычисления на веб-странице ... 258
Sharedworker'ы – надо делиться ... 264

WebSockets – забудем про HTTP? 268
Web Sockets – TCP для веба 268
WebSocket-серверы 270
Работаем с phpDaemon 271

Web Intents – средство общения приложений 279

Прочие полезные вещи 283
События колесика мыши 283
Full-Screen API 284
Состояние online 285
Page VisibilityAPI 285
Battery Status – API, продиктованный жизнью 286
History Api 288
RequestAnimationFrame – решение проблем JavaScript-анимации 289
Prerendering – отрисовываем страницы заранее 291
Selectors API – простой синтаксис доступа к DOM-элементам 291
Расширения DOM 292
Web Notifications API – настоящие pop-up'ы 293
Mouse Lock/Pointer Lock API 295

Mozilla WebAPI – будущее наступило? 298

Приложение 1. Ресурсы для работы с HTML5-технологиями 302

Приложение 2. Спецификации W3C, имеющие отношение к HTML5-технологиям 303

Предметный указатель 305

Введение

Писать о технологии, стандарты которой не утверждены, просто здорово. Можно дать волю своей трактовке концепций, можно порассуждать о перспективах и еще не реализованных возможностях... Все это, наверное, заманчиво, может, даже слишком заманчиво, судя по количеству появившихся в последнее время публикаций по HTML5, перспективы утверждения спецификаций которого лежат еще не в очень близком будущем.

Конечно, огромный интерес к новой технологии – это хорошо. Плохо другое – многие из вышедших за последнее время книг подробно рассказывают об истории языка разметки, затем более или менее подробно освещают два-три нововведения (обычно это canvas, семантическая разметка и, разумеется, video) и... и все. Толку от этих, зачастую повторяющих друг друга публикаций для разработчика немного. Именно это обстоятельство и заставило меня попробовать создать такой материал, который мне самому (как веб-разработчику) был бы максимально полезен.

Цель написания этой книги – в том, чтобы любой программист, работающий в области интернет-технологий, мог быстро ознакомиться с новым для себя API или технологией (WebStorage, AppCache, FileSystem API – в HTML5 их множество), разобрал примеры кода и немедленно приступил к работе, творя новый, прекрасный веб будущего!

HTML5 – это уже не язык разметки, это сумма технологий, предоставляемых современными браузерами, каждая из которых привносит в существующий веб что-то новое. В книге дается описание тех из них, которые уже доступны в настоящее время. На каждое решение, каждое API дается пример работающего кода.

По мере возможности рассматриваемые технологии тематически сгруппированы в рамках одного направления.

В приложении перечислены спецификации консорциума W3C, находящиеся в той или иной степени готовности, регламентирующие упомянутые технологии.

Будущее www – какое оно?

Июньским вечером 2004 года представители консорциума W3C покидали зал заседаний в г. Сан-Хосе (северная Калифорния) в хорошем настроении. У них было на это основание. Два дня прошли в жарких дебатах, но здоровый консерватизм и академизм победил – в результате анонимного голосования 14 голосами против восьми было признано нерациональным предложение представителей веб-индустрии о пересмотре спецификаций HTML и DOM в сторону расширения. На встрече присутствовали представители Microsoft, Mozilla Foundation и Opera Software, ведущих на тот момент производителей браузеров. Они пытались донести до собравшихся свое видение веба будущего, но проиграли.

Естественно, уважаемые представители W3C тогда не подозревали, что основным следствием их решения станет серьезный подрыв авторитета консорциума и что жизнь и реальные подробности веб-общественности уже очень скоро возьмут свое. Мы еще вернемся к продолжению этой истории, а пока давайте посмотрим, что же такого революционного предлагали производители браузеров.

Сам исторический вопрос звучал следующим образом: *Should the W3C develop declarative extension to HTML and CSS and to address medium level Web Application requirements, as opposed to sophisticated, fully-fledged OS-level APIs?* (Должен ли W3C развивать декларативное расширение HTML и CSS и обязательно дополнять DOM для решения требований среднего уровня веб-приложений, в отличие от сложных API уровня ОС?)

От чего же отказались уважаемые члены консорциума и каких именно изменений хотели «практики»? Все было довольно просто. В существующем виде HTML перестал удовлетворять запросам пользователей www, и производители браузеров знали, что делать, чтобы исправить ситуацию. Введение пары или пары десятков новых тегов тут положение не спасало. Веб нуждался в новой функциональности, новых возможностях, которые включали бы в себя простые способы отображения мультимедиа-контента и графики, возможности геолокации, средства для создания умных интерактив-

ных приложений, расширенные средства коммуникации и многое, многое другое.

Самое обидное было то, что технические средства для решения многих из этих задач к тому моменту уже были. Впрочем, для лучшего понимания вопроса нам придется разобраться в ситуации чуть подробней.

История вопроса

Я честно не хотел писать какой-либо исторический очерк и ковыряться в перипетиях развития стандарта, но после некоторых колебаний понял, что знание истории поможет лучше понять подходы и концепции HTML5. Правда, я совершенно не готов (главным образом не на уровне эрудиции, а морально) начинать с Тима Бернерса-Ли и NCSA Mosaic. Лучше начнем с HTML 4, спецификация которого была утверждена в декабре 1997 года.

Актуальность стандарта

Да-да, если не считать некоторых изменений, внесенных два года спустя, мы все еще живем и делаем сайты по стандартам, утвержденным в середине 90-х. За это время произошло просто колоссальное количество событий. Я не имею ввиду кризис в Югославии, отставку Ельцина, войну в Ираке и прочие мелочи. У нас, в веб-индустрии, прошла целая эпоха – закончилась и снова началась война браузеров, серверы заполонила архитектура LAMP, php стал самой популярной и самой презираемой технологией в веб, пришел ajax, а с ним и web 2.0 (который, правда, уже понемногу забывают), появился этот кошмар под общим названием социальные сети, higload, REST, COMET, NoSQL... Наконец, веб проник на мобильные устройства, причем на такие, которые в 90-х можно было увидеть разве что в сериале Star Track. И вот при всем при этом мы все еще живем при HTML 4.0. Ну, хорошо, на самом деле на 4.01, но кто-нибудь с ходу вспомнит разницу?

XHTML – стандарт для стандартизаторов

Впрочем, нельзя сказать, что все это время люди, отвечающие за стандарты, сидели сложа руки. Строго говоря, HTML является одной из реализаций SGML (стандартного обобщенного языка разметки – стандарт ISO 8879), причем реализацией добровольно при-

митивной – одно только описание стандарта SGML представляет собой 40-мегабайтный PDF-документ. И первое, что пришло в голову разработчикам консорциума W3C, – представить язык разметки веб-страниц в более упорядоченном и структурированном виде, приведя его к другому производному от SGML стандарту, получившему к тому времени широкое распространение расширяемому языку разметки XML. В результате на свет появился стандарт XHTML (***Ex**tensible **H**yper**t**ext **M**arkup **L**anguage* – расширяемый язык разметки гипертекста). Основные отличия его от HTML можно перечислить в нескольких пунктах:

- все теги (основные элементы HTML/XHTML) должны быть закрыты. Даже не имеющие закрывающего тега изначально. В XHTML, например, элемент `<img>` станет таким: `<img />`;
- все имена тегов и атрибутов должны быть записаны строчными буквами (никаких `<BODY><HEAD></HEAD>`, только так: `<body>`);
- все атрибуты обязательно заключаются в кавычки;
- булевы атрибуты записываются в развернутой форме. Например:

```
<input type ="checkbox" checked="checked" />;
```

- все служебные символы, не относящиеся к разметке, должны быть заменены HTML-сущностями. Например: < на `&lt;`; а & на `&amp`.

Кроме того, XHTML-документ должен подчиняться правилам валидации обычного XML: допустимо существование только одного корневого элемента, не принимается нарушение вложенности тегов (например, конструкции вида `<a><i>Text</a></i>`, вполне позволительные в HTML).

Впрочем, самое главное отличие заключалось не в синтаксисе а в отображении XHTML-документа браузером. При встрече браузером значения поля content-type в заголовке http паркета, равного application/xhtml+xml, документ обрабатывается xhtml-парсером, аналогично обработке XML-документа. При этом ошибки в документе не исправляются. Согласно рекомендациям W3C, браузеры, встретив ошибку в XHTML, должны прекратить обрабатывать документ, выдав соответствующее сообщение.

Спецификация XHTML 1.0 была одобрена в качестве рекомендации консорциума Всемирной паутины в январе 2000 года. В августе 2002 года была опубликована вторая редакция специфика-

ции – XHTML 1.1. Параллельно полным ходом началась разработка XHTML 2.0, призванного стать новым уровнем представления документов во Всемирной сети. Разработчики пошли на довольно смелый шаг – нарушение обратной совместимости, но нововведения, которые они собирались внести, стоили того. XHTML 2.0 содержит спецификации Xforms, Xframes, призванные заменить стандартные HTML-формы и фреймы соответственно, ML Events – API для управления DOM-структурой документа, встроенную поддержку модулей Ruby character и многое другое. Работа шла полным ходом, но было несколько обстоятельств, совершенно не радующих авторов спецификаций. Если коротко, XHTML просто не получил должного распространения.

Во-первых, огорчали веб-разработчики, которые после вольницы HTML никак не хотели принимать новые правила в полном объеме. Расставлять в нужном месте кавычки и сущности оказалось просто непосильной задачей. Что там говорить про XHTML, если и со стандартами HTML4 веб-верстальщики обходились достаточно вольно. И что самое возмутительное – производители браузеров активно им в этом потакали!

И именно в этом заключалась вторая проблема. На самом деле все довольно понятно – те, кто делали браузеры, просто не могли допустить, чтобы какой-либо значимый контент в них был не доступен пользователю из-за каких-то неясных принципиальных соображений, и, надо сказать, они были по-своему правы (ну в самом деле, веб нам нужен для общения с миром, а не для того, чтобы все атрибуты были снабжены кавычками!). В результате наиболее популярные браузеры имели два режима отображения XHTML-документов, причем по умолчанию обычно работал «нестрогий» режим, при котором огрехи в разметке милосердно прощались. Хуже того, безусловно, самый распространенный на тот момент браузер Internet Explorer вообще не реагировал на MIME-тип application/xhtml и не имел в своем составе парсера обработки XHTML-документов вплоть до восьмой версии.

Главная причина неудачи повсеместного внедрения XHTML довольно проста. Строгие правила валидации, атрибуты, взятые в кавычки, закрытые одиночные теги... все это, может, и хорошо, но нужны эти тонкости в основном самим разработчикам и блюстителям стандарта, но никак не пользователям, которым, строго говоря, дел нет до всех этих тонкостей. И никак не создателям веб-контента, которым эти правила попросту ничего не дают, кроме несильной

головной боли. В общем, сложилось что-то вроде революционной ситуации – создателям стандарт не нужен, а потребителям он не нужен тем более. А что всем им было нужно? Живой интерактивный веб-контент, воплощающий социальные потребности современного человека. Плоский мир HTML с этим справлялся плохо, и к концу 90-х на веб-странице появились не относящиеся к языку разметки компоненты.

За пределы разметки – ActiveX, Java, Flash

Попытки выйти за пределы возможностей HTML начались всего через два года после появления браузера Mosaic. На следующий год после появления W3C – в 1995 г. – Sun Microsystem вместе с первой версией платформы Java представила технологию Java Applets – прикладных программ, чаще всего написанных на языке Java в форме байт-кода и выполняемых в браузере посредством виртуальной Java-машины (JVM).

Технологически это работает следующий образом: апплет (скомпилированная в байт-код Java-программа) встраивается в HTML-разметку с помощью специального тега `<applet>` (в настоящее время он признан устаревшим) или более современного `<object>`. Код апплета загружается с веб-сервера и исполняется браузером в «песочнице». Такой подход позволяет привнести в браузер значительную часть мультимедийных, интерактивных и коммуникационных возможностей Java.

К достоинствам апплетов можно отнести кроссплатформенность (они будут исполняться везде, где установлена JVM).

Недостатков у технологии довольно много, прежде всего это необходимость Java-плагина для браузера. Они прочно завязаны на JVM и страдают от связанных с подобными приложениями проблем с совместимостью версий и безопасности. Впрочем, причина, по которой Java Applets так и не получили большого распространения, скорее, в другом – для их работы необходим запуск JVM, а это совсем не добавляет скорости исполнения и производительности.

В 1996 году компания Microsoft представила свое расширение для возможностей веб-страниц – технологию ActiveX. Это было развитие Component Object Model (COM) и Object Linking and Embedding (OLE). Компонент ActiveX встраивается в веб-страницу с помощью тега `<object>`, он исполняется операционной системой, и вся модель

работы, основанная на COM, диктует их применение только на операционных системах семейства Windows. Благодаря этой же модели сами компоненты могут быть разработаны на любом языке программирования, поддерживающем Component Object Model.

ActiveX позволяет браузеру Internet Explorer запускать другие приложения – например, Media Player или Quicktime. В ограниченном объеме компонент имеет доступ к другим возможностям операционной системы. Но все это, разумеется, только на платформе Microsoft Windows и только с помощью обозревателя от той же компании.

Самое применяемое и самое успешное на настоящий момент расширение возможностей веб-страниц началось с разработки небольшой компании FutureWave, FutureSplash Animator, представляющий собой пакет анимации в векторном формате. В 1996 году FutureWave была приобретена компанией Macromedia, и продукт под названием Macromedia Flash начал завоевывать Интернет.

Для работы Macromedia Flash браузеру требовался специально устанавливаемый плагин, но технология оказалась настолько удачной, что это не стало препятствием к ее распространению.

Наверное, нет нужды рассказывать, где применяется флэш сегодня, – это рекламные баннеры, анимация, игры, а также воспроизведения на веб-страницах видео- и аудиоконтента. В некоторых областях до недавнего времени у флэша просто не было альтернатив.

Браузерный плагин Flash Player представляет собой виртуальную машину, на которой выполняется загружаемый из Интернета код flash-программы.

Анимация во Flash реализована через векторный морфинг, то есть плавное «перетекание» одного ключевого кадра в другой. Это позволяет разрабатывать сложные мультипликационные сцены через отрисовку нескольких ключевых кадров. Для реализации логики Flash использует язык программирования, основанный на ECMAScript.

Впрочем, технология тоже не свободна от недостатков. Прежде всего это значительная нагрузка на процессор, связанная с реализацией флэш-плеера, разные затруднения при воспроизведении флэш-контента на платформах, отличных от Windows. Большую проблему представляет недостаточный контроль ошибок, приводящий зачастую к праху браузера при сбое в приложении.

Кроме того, содержимое флэш-ролика недоступно для индексирования поисковыми системами, что в наше время может послужить приговором любому ресурсу.

Тем не менее Macromedia Flash в ряде случаев является единственным реальным способом воплотить в вебе разнообразное интерактивное, мультимедийное. Вернее, являлся. До HTML5.

Рождение HTML5

И вот мы подошли к моменту, с которого начинали свое повествование, – историческое голосование в июне 2004 года, о содержании и итоге которого уже говорилось. Резюме семинара гласило: «В настоящее время W3C не намерен предоставлять любые ресурсы сторонней теме неофициального опроса: расширение HTML и CSS для веб-приложений, помимо технологий, разрабатываемых в соответствии с уставом текущей Рабочей Группы W3C». После этого World Wide Web Consortium мог, не отвлекаясь, сосредоточиться на будущих разработках XHTML 2.0, а представители веб-сообщества... Нет, они не опустили руки и не смирились с такой ситуацией. Уже в этом месяце был зарегистрирован домен whatwg.org, так и родилась организация WHAT Working Group, которую основали уже упомянутые производители браузеров: Apple, Mozilla Foundation и Opera Software. WHATWG – рабочая группа по разработке гипертекстовых приложений для веб (*Web Hypertext Application Technology Working Group*). Это свободное, неофициальное и открытое сотрудничество производителей браузеров и заинтересованных сторон. Направление работы данной организации – разработка спецификаций на основе HTML и связанных с ним технологий. Предполагалось, что работы WHATWG по формальному расширению HTML должны стать основой новых стандартов. Причина создания этой организации была обозначена вполне откровенно – пренебрежение W3C к реальным потребностям пользователей. HTML уважаемый консорциум уже не интересовал, выбор был сделан в пользу XML. Вместо укладывания веба в это прокрустово ложе WHAT Working Group применила другой подход, уже практиковавшийся в браузеростроительстве, – узаконивание «нестрогих» алгоритмов отображения разметки, щадяще подходящих к обработке ошибок.

В рамках WHATWG было разработано несколько спецификаций, объединенных в проект Web Applications 1.0. Первый черновик WA был выпущен в сентябре 2006 года и включал такие интересные расширения html, как возможность рисования (canvas), реакция на события сервера, встроенную поддержку аудио- и видеоконтента и многое другое. Кроме того, была доведена до стандартизации

другая разработка – развитие идеи Web Forms 2.0 (изначально разрабатываемых в рамках XHTM2), добавляющая новые типы полей в HTML-формы.

Два с половиной года между W3C и WHATWG продолжалось если не противостояние, то что-то вроде холодной войны. WHATWG работала на HTML, W3C трудилась над XHTML 2.0. И вот к октябрю 2006 года сложилась вполне ясная ситуация – стало понятно, что первые достигли серьезных результатов, которые уже вполне видны, в то время как XHTML 2 представляет собой кучу недоработанных черновиков, не имеющих реального воплощения ни в одном браузере. Игнорировать WHATWG со стороны консорциума далее было бы просто нелепо, и в октябре 2006 года сам основатель W3C Тим Бернерс-Ли заявил, что W3C будет работать вместе с WHAT Working Group над развитием HTML. Надо сказать, почти ко всеобщей радости. Одним из первых решений организованной W3C HTML Working Group было решение переименовать «Web Applications 1.0» в «HTML5».

Закономерным итогом стало объявление 2 июля 2009 года W3C о том, что по истечении в конце 2009 года срока действия Устава рабочей группы XHTML 2 он (устав) продлен не будет. Все ресурсы переводятся в Рабочую группу по разработке HTML5. Этим решением W3C прояснил свою новую позицию относительно будущего HTML.

Для окончательного разрешения ситуации надо отметить, что в начале 2011 года WHATWG приняла решение отказаться от упоминания версии HTML5, заменив ее простым названием HTML, под которое теперь попадают все последующие версии стандарта. То есть как раз версий больше не предусмотрено – предлагается постоянное развитие. Это, в частности, обозначает, что, строго говоря, книга эта вовсе не про HTML5, а про современное состояние HTML. Просто HTML.

Впрочем, это еще не все, и в своем изложении я упустил достаточно интересный и конкурентный класс технологий, без которых для понимания современной стратегической ситуации в мире www никак нельзя!

RIA

Растущий разрыв между потребностями пользователей www и скудными возможностями HTML, который, пусть даже расширенным JavaScript и CSS, не был способен на многое, породил целый класс веб-приложений, которые одно время всерьез претендовали на то, чтобы стать будущим веба. Впрочем, и сейчас претензии на это все еще остались. Я говорю о RIA – Rich Internet Applications, термин, который, наверное, лучше переводить как «насыщенные» интернет-приложения.

Основное отличие работы приложений RIA от традиционного веба состоит в уходе от четкой клиент-серверной архитектуры, при которой браузер являлся тонким клиентом. Постоянная необходимость отправки данных на сервер и ожидания получения ответа сильно сужала рамки дозволенного в веб-технологиях. Ну, представьте себе, например, простую стрелялку, где результатов вашего выстрела приходится ждать десятки секунд, по плохим каналам с далекого заокеанского сервера... В случае же с RIA в браузере запускается полноценное приложение, для которого взаимодействие с сервером носит только вспомогательный характер. По сути, RIA – это приложения, работающие через сеть и предоставляющие клиенту ресурсы веб-сервера, но обладающие функциональностью полноценных настольных приложений.

При всех различиях RIA имеют ряд общих черт. Перечислим их:

- RIA включают в себя программную «прослойку» между пользовательской частью веб-приложения и сервером, представляющую собой программный движок, надстройку к браузеру, запускающемуся в начале работы с приложением;
- работа с RIA требует единовременной установки дополнительного ПО в виде плагина к браузеру;
- приложения запускаются локально в среде безопасности, называемой «песочница» (sandbox).

Необходимость последнего обстоятельства совершенно понятна: если раньше программам, запускаемым в браузере, позволялось очень немногое – установить куки, иногда закэшировать содержи-

мое, то теперь в распоряжении RIA файловая система, память видеокарты и прочие ресурсы вашего компьютера, которые необходимы для нормальной работы полноценного приложения. Как правило, любое RIA выполняется в локальной, изолированной среде и, хотя использует ресурсы компьютера-клиента, не может фатально влиять на его систему.

Давайте кратко осмотрим современные RIA, ставшие заметными в www.

Microsoft Silverlight

Наверное, самой успешной и получившей наибольшее распространение технологией этого класса стала разработка компании, привычно именуемой «софтверным гигантом», Microsoft Silverlight. Она представляет собой классическое Rich Internet Application, включая в себя плагин для браузеров, воспроизводящий различный мультимедиа-контент.

Microsoft Silverlight родилась как часть, или, вернее, как версия Windows Presentation Foundation (WPF – графическая подсистема для построения пользовательских интерфейсов клиентских Windows-приложений) для веб-среды.

Техническая реализация включает построение пользовательского интерфейса на основе языка XAML (*e**X**tensible **A**pplication **M**arkup **L**anguage* – расширяемый язык разметки приложений) и модуля расширения браузера, который обрабатывает XAML-конструкции и отображает итоговый контент в поле обозревателя. Модуль предоставляет разработчикам доступ к объектам XAML-страницы посредством JavaScript, делая возможным создание полноценных графических и мультимедийных приложений. Модуль разработан для всех распространенных браузеров и требует установки (рис. 1).

Первая бета-версия Silverlight была выпущена в декабре 2006 года, а релиз Silverlight 1.0 состоялся в мае 2007 года. Эта версия имела базовые графические возможности, в частности анимацию и базовые элементы пользовательского интерфейса.

Основной особенностью Silverlight 2, вышедшего в октябре 2008 года, стала интеграция технологии с платформой .NET Framework. В ней появился большой набор новых элементов управления (например, DataGrid, TreeView), новые возможности для работы с видео и другие возможности. Тогда же появился инструмент разработки – Microsoft Silverlight Tools for Visual Studio 2008, включаю-

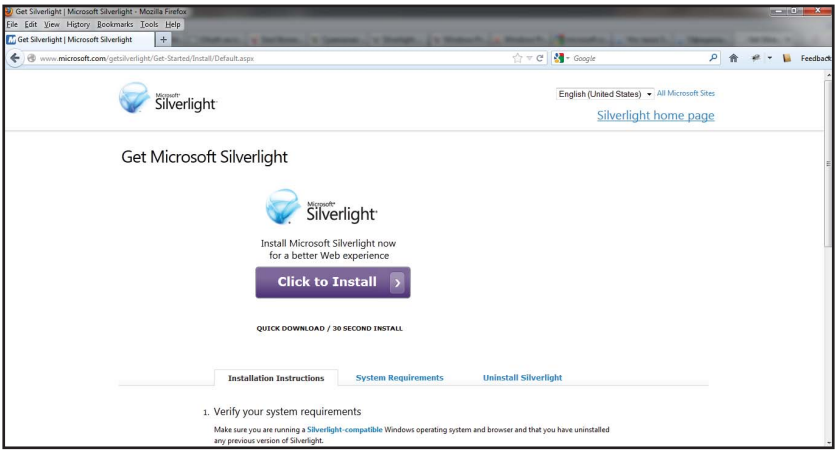

Рис. 1. Устанавливаем Sillverlight

щий в себя генераторы кода для XAML, шаблоны для проектов Visual Basic и C#, средства отладки Silverlight-приложений и прочие необходимые для создания программ вещи.

Silverlight 3 вышел в июне 2009 года. В новую версию были добавлены такие инструменты работы с графикой, как пиксельные шейдеры, обеспечивающие псевдо-3D-рендеринг (так называемый «2,5D»), плавную анимацию (вплоть до реалистичного видео), поддержка аппаратного ускорения при работе с трехмерной графикой, поддержка устройств с multitouch-интерфейсом.

Еще в Silverlight 3 был впервые представлен функционал OOB (*out-of-the-browser – вне браузера*), дающий некоторые возможности по исполнению Silverlight-приложений вне обозревателя.

Версия Silverlight 4 появилась в апреле 2010 года. В ней было добавлено много просто революционных возможностей. Прежде всего это поддержка веб-камеры и микрофона и возможность передачи видеопотока от клиента на сервер в любых приложениях, поддержка буфера обмена, drag&drop и показа оповещений, поддержка Managed Extensibility Framework и многое другое.

Реализована полноценная поддержка офлайн-приложений с доступом к локальным папкам и внешним COM-объектам, а также с возможностью отображения в них HTML через встроенный браузер. Введена поддержка сервисов WCF и механизмов DRM (да-да, я знаю, именно их российским разработчикам и не хватало!).

Финальный релиз Silverlight 5 стал доступен в декабре 2011-го. Среди новшеств: поддержка графического процессорного 3D, а также ускорение декодирования видео.

Поддержка технологии XNA, поддержка 64-битных браузеров, изменяемая скорость воспроизведения медиаконтента с автоматической коррекцией звука, поддержка ускорения запуска приложений.

Впечатляет? Но не все так хорошо.

К недостаткам технологии, безусловно, относится закономерное отсутствие поддержки платформ, отличных от Microsoft Windows, и если раньше подобный факт был бы просто проблемой для этих самых платформ, то теперь это не так. Приходится считаться с распространением широкого спектра различных мобильных устройств, для которых Windows – не только не незаменимая, но и не самая популярная среда.

Правда, Silverlight поддерживается для Mac OS 10.4/10.5 для браузеров Firefox и Safari на платформе Intel, но в данном случае это мало меняет ситуацию. В рамках Mono (проект по реализации функционала фрэймворка .NET Framework на базе свободного программного обеспечения) существует разработка под созвучным названием Moonlight, открытая программная реализация Microsoft Silverlight. Первая стабильная версия Moonlight 1.0 была выпущена в январе 2009 года. Она поддерживала Silverlight 1.0. Moonlight 2.0 появилась 17 декабря 2009 года. В ней декларировались полная поддержка Silverlight 2.0 и реализация некоторых возможностей Silverlight 3.

В любом случае, судьба технологии вызывает законные опасения – в новом графическом интерфейсе от Microsoft (metro) от нее отказались в пользу HTML5, и это, без сомнения, тревожный сигнал для Silverlight.

Adobe Flex

Технология Macromedia Flash, ставшая собственностью компании Adobe, во многом предвосхитила концепцию RIA. Когда наше время потребовало от Flash нечто существенно большего, чем красивые элементы управления и надоедливые баннеры, родилось воплощение Rich Internet Application от Adobe – платформа Adobe Flex.

Flex расширяет возможности Flash, позволяя описывать интерфейс приложения на XML. Логика приложения пишется на языке ActionScript 3. Результатом компиляции является файл формата SWF.

Скомпилированный файл может выполняться как в браузере, в среде Flash Player, так и в виде самостоятельного приложения платформы Adobe AIR. Это является и основным преимуществом Flex перед Microsoft Silverlight – он «условно кроссплатформен», может исполняться в любом браузере, для которого существует Flash-проигрыватель или соответствующие библиотеки.

Физически Flex представляет собой framework, набор классов, расширяющих возможности Flash. Среди базовых возможностей – локализация, валидация вводимых данных, форматоры текстовых полей и прочие возможности, позволяющие вести RAD-разработку. Кроме этого, Flex предоставляет богатые мультимедийные возможности, включая потоковое мультимедиа, доступ к веб-камере и микрофону пользователя.

Сетевые возможности среды включают HTTP-запросы, интерфейс к веб-сервисам, бинарные сокеты (это возможность передачи RealTime-данных). Flex может взаимодействовать с сервером, получая данные через XML, SOAP, Sockets, ZLIB и т. д.

Еще встроенный формат сериализации AMF, операции с координатами трехмерного пространства, встроенные графические фильтры (такие как расфокусировка, падающая тень), возможность расширения функционала путем написания собственных модулей.

В основе построения интерфейса, так же как и в Silverlight, лежит XML-язык разметки – MXML.

Для создания приложений с использованием технологии Flex Adobe System создана мощная среда разработки.

К недостаткам технологии можно отнести некоторую избыточность, заложенную в самой архитектуре Flex-framework. В каждое приложение необходимо включать стандартный набор классов, занимающий более 700 Кб в итоговом swf-файле. Естественно, это не лучшее решение для веб-среды, особенно если речь идет о мобильных устройствах с ограниченными ресурсами. Правда, в более поздних версиях флеш-плеера реализована возможность подгружать только необходимые классы flex, не включая их в каждый отдельный исполняемый swf-файл. Но один раз в кэш плеера среда загрузиться должна, да и загрузку самого плеера никто не отменял.

В конце 2007 года компания Adobe решила открыть исходный код среды Flex и начать его распространение на условиях Mozilla Public License (MPL).

Последняя версия среды Adobe Flex – Flex 4.5 Hero Release – была выпущена в октябре 2010 года.

В 2011 году компания Adobe приняла решение о передаче Flex в состав Apache Software Foundation. В январе 2012-го Apache Foundation утвердила принятие разработок в свой инкубатор, который, к сожалению, имеет нехорошую репутацию «кладбища проектов». Хотя в данном случае вряд ли будет все так плохо – Flex давно активно применяется во многих решениях.

JavaFX

Если вспомнить историю создания интернет-приложений, придется признать за компанией Sun первенство в деле создания Rich Internet Application. Первые Java-апплеты, продемонстрированные Гослингом при презентации браузера WebRunner в далеком 1994 году, вполне подходили под это определение. Правда, с тех пор прошло много всяких событий, интернет-приложения росли и изменялись, появилась технология Flash, использование клиентских возможностей браузера (Javascript, DOM) вылилось в термин WEB-2, а апплеты как технология, в общем, не сильно изменились. Заняв прочное положение на мобильных устройствах, на десктопах они так и не получили заметного распространения.

Прорыв на рынок RIA компания Sun совершила во второй половине 2000-х, представив новую платформу для веб-приложений – JavaFX. Впервые технология была показана на конференции JavaOne в мае 2007-го. В декабре 2008 года вышла JavaFX 1.0, включающая в себя средства разработки – JavaFX 1.0 SDK, плагин для NetBeans IDE 6.5 и JavaFX 1.0 Production Suite – набор инструментов для экспорта графических объектов в приложения JavaFX. Была представлена также бета-версия эмулятора JavaFX 1.0 Mobile для разработки JavaFX-приложений для мобильных платформ. JavaFX TV – среда для запуска приложений на телевизионной платформе, планировалась к запуску в начале 2010 года.

Все чуть не закончилось в апреле 2009-го с утерей Sun Microsystem самостоятельности и переходом всех ее разработок под крыло Oracle. Но после некоторого затишья вскоре стало ясно, что технология заброшена не будет.

Что конкретно представляет собой JavaFX-приложение? Это прежде всего интерфейс и логика, написанные на декларативном языке JavaFX Script. Он имеет простой синтаксис, коллекцию встроенных объектов, а самое главное – может обращаться к любым библиотекам платформы Java. JavaFX использует для работы Java-машину

(см. рис. 2) и, по сути, является частью платформы. В этом ключевое преимущество RIA от Sun/Oracle – за ней вся мощь Java.

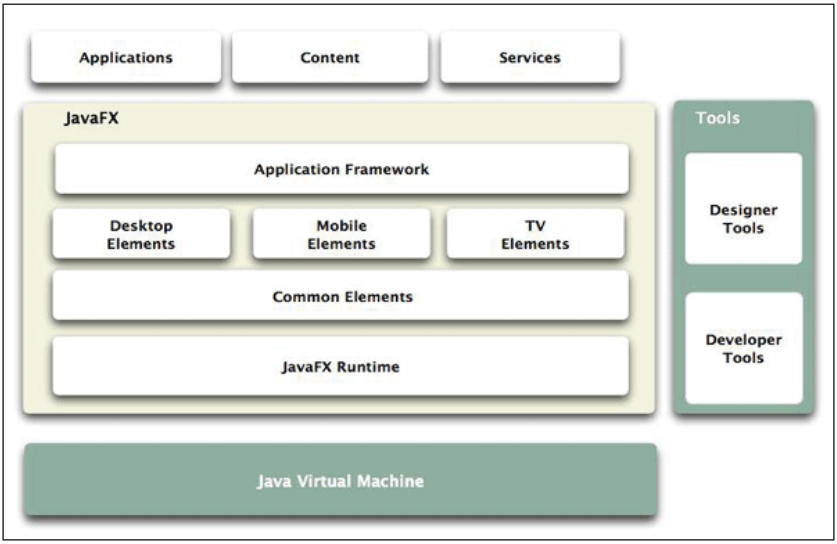

Рис. 2. Архитектура JavaFX

JavaFX-приложение может функционировать как «апплет нового образца», а также в виде автономного приложения (через Java Web Start – сам апплет можно вытащить из содержащей его страницы, закрыть последнюю и продолжить работу). Такой продукт является интернет-приложением только потому, что через Глобальную сеть происходит доставка его потребителю, а также по причине наличия возможности активного взаимодействия через Интернет с сервером. В то же время в несетевой ипостаси такие приложения почти ничем не отличаются от обычных настольных программ (на рис. 3 – калькулятор, написанный на JavaFX).

Рис. 3. Веб-калькулятор, написанный на JavaFX

Особенности технологии позволяют легко встраивать в приложения мульти-

медиа-данные, анимацию и различные визуальные эффекты, а также использовать собственные визуальные примитивы.

На JavaOne 2011 было объявлено о выпуске финальной версии JavaFX 2.0. Среди особенностей новой версии среды – отказ JavaFX Script в пользу стандартного Java API, что позволяет писать JavaFX-приложения на Clojure, Scala или Groovy, в общем, на языках, базирующихся на JVM. Новый движок рендеринга, использующий аппаратную акселерацию для работы с 3D-графикой, новый компонент WebView для встраивания веб-контента в JavaFX-приложения (в том числе использование HTML5 API!).

Основная проблема использования JavaFX – это необходимость установленного у клиента Java Runtime Environment (JRE). Помимо чисто технических, это в наше время еще и лицензионные проблемы.

Что касается технических вопросов – несмотря на заявленную кроссплатформенность, до сих пор технология полноценно работает только на операционной системе Windows. Полноценная поддержка Linux обещана в следующем году, правда... это не первый следующий год. Что касается платформы Mac OS X, то поддержка ее заявлена, но, похоже, несколько поздновато.

Google Native Client

Строго говоря, Google не позиционировало свою технологию Native Client как платформу для Rich Internet Applications, но по формальным признакам она вполне вписывается в этот класс ПО.

Ее суть – запуск в браузере модулей, написанных на нативном коде (увы, адекватного перевода на родной язык термина «native code» в голову не приходит) для архитектуры x86.

В отличие от JavaFX или Silverlight, в этой технологии нет компиляции в байт-код и какой-либо виртуальной машины. Была создана среда выполнения, позволяющая запускать обычные, «родные» для этой платформы программы в безопасном для данной системы окружении. Разработчики идеально выдержали модель «песочницы».

Во избежание взаимодействия Native Client непосредственно операционной системой весь код исполняется в отдельном, изолированном контейнере. Это позволяет модулю использовать системные ресурсы, но в то же время ограждает ОС от возможного случайного или злонамеренного повреждения.

В целом Native Client (NaCL) состоит из контейнера, играющего роль песочницы, и среды исполнения (runtime) нативного кода. Третьим элементом выступает плагин для веб-браузера. Для комму-

никации между браузером и NaCL-модулем предоставлены два варианта: simple RPC-интерфейс (SRPC) и давно известный Netscape Plugin Application Programming Interface (NPAPI).

Писать модули для Google Native Client предполагается на любом компилируемом на данной системе языке программирования.

Native Client распространяется под лицензией BSD, имеет реализации для различных браузеров и платформ.

Еще несколько лет назад Native Client многими рассматривался как веб-платформа будущего. Сейчас новости об этой технологии занимают довольно скромное место на фоне известий о новых API HTML5, но Google от него отказываться явно не собирается. Так, начиная с 14-й версии браузера Google Chrome, Natve Client включен в состав обозревателя, и его пользователям больше не требуется устанавливать никаких дополнительных плагинов.

RIA и HTML5

В настоящее время про Rich Internet Application слышно значительно меньше, чем 5–6 лет назад, интерес к ним со стороны IT-общественности если не падает, то уж точно не растет. Может сложиться впечатление, что RIA стремительно сдают свои позиции в будущем веба HTML5 и JavaScript, и, по крайней мере, отчасти это впечатление верно. Тому есть причины.

Все RIA имеют принципиальные недостатки, диктуемые им их архитектурой. В первую очередь это необходимость подгружать/устанавливать дополнительное программное обеспечение, к которому относится как сам RIA-плагин, так и восполняемые им скрипты. Вторая проблема, которая на самом деле гораздо серьезней, состоит в том, что RIA-движок является чужеродной для браузера средой, чаще всего непрозрачной и недоступной для доступа из сценариев. Фактически HTML, DOM, CSS являются в них лишь фронтендом, дополнительным внешним слоем приложения. Таким образом, однородность веб-среды принципиально нарушается. В то же время HTML5 способен предоставить единую прозрачную среду выполнения, с доступными компонентами.

Все это так, но они обладают и массой интересных возможностей, аналогов которых в HTML5 в настоящий момент нет. И хотя с развитием суммы веб-технологий, входящих в определение HTML5, назвать таковые все труднее, архитектура Rich Internet Application все равно представляет интерес, и вполне возможно, что RIA будут значимой частью будущего веб на новом витке его развития.