Сухов К.

HTML5

путеводитель по технологии



Обновленное издание

ДМК ПРЕСС
ИЗДАТЕЛЬСТВО

УДК 004.738.5:004.4HTML5
ББК 32.973.202-018.2
С89

Сухов К.

С89 HTML5 – путеводитель по технологии. – М.: ДМК Пресс, 2014. – 352 с.: ил.

ISBN 978-5-94074-997-4

Книга посвящена знакомству и незамедлительному началу использования на практике HTML5 – новому стандарту и флагману современных интернет-технологий. Все новые API (Canvas, Geolocation API, WebStorage, WebSockets, WebRTC, WebGL IndexedDB и многое, многое другое) рассмотрены на основе практических примеров, и большую часть из них можно использовать прямо здесь и сейчас. Книга адресована веб-программистам, веб-верстальщикам, ведущим веб-проектов и вообще всем, кто имеет отношение к интернет-разработке.

Мы можем относиться к новому стандарту как угодно, это не важно – важно понимать: HTML5 – это уже не будущее, это настоящее. И не приняв его, мы рискуем застрять в XX веке. Это, может, и не самое плохое время, но если мы работаем с информационными технологиями – давайте жить и творить сегодня!

Для хорошего понимания материла желательны знания HTML/DHTML/JavaScript и общее представление об устройстве Всемирной сети.

УДК 004.738.5:004.4HTML5
ББК 32.973.202-018.2

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

Материал, изложенный в данной книге, многократно проверен. Но, поскольку вероятность технических ошибок все равно существует, издательство не может гарантировать абсолютную точность и правильность приводимых сведений. В связи с этим издательство не несет ответственности за возможные ошибки, связанные с использованием книги.

© Сухов К., 2014
© Оформление, ДМК Пресс, 2014

ISBN 978-5-94074-997-4

Содержание

Введение 7

Будущее www – какое оно? 8

История вопроса 9

Актуальность стандарта 9
XHTML – стандарт для стандартизаторов 9
За пределы разметки – ActiveX, Java, Flash 12
Рождение HTML5 14

RIA 16

Microsoft Silverlight 17
Adobe Flex 19
JavaFX 21
Google Native Client 23
RIA и HTML5 24

HTML5 сегодня 25

О браузерах 26
Проверять возможности, а не версии 26
Moz-, webkit- и прочие vender-грабли 27
Modernizr – бархатный путь в HTML5 29

HTML – это теги 35

Структура страницы 35
Воплощение концепции семантической разметки 37
Всякие полезности 42
Атрибуты и аксессуары 46
Атрибуты data-* 50
Отречемся от старого мира (что выбросили, что изменили) 51

HTML5-формы – о чем мы мечтали 55

Новые поля ввода 55
INPUT... а OUTPUT? 62
Не только разметка – объект ValidityState и другие 64

HTML5 Accessibility – всего лишь доступность (ARIA, WCAG) 68

WCAG – рекомендации, которые никто не слушал 68
WAI-ARIA – перманентно временное решение, которое работает 71
Проблемы доступности 72

ARIA-роли ... 73
Абстрактные роли (Abstract Roles) ... 73
Роли – виджеты (Widget Roles) ... 74
Роли структуры документа (Document Structure Roles) ... 75
Роли разметки (Landmark Roles) ... 76
Состояния и свойства объектов – ARIA-атрибуты ... 77
Атрибуты виджетов ... 77
Атрибуты для Live Region ... 77
Атрибуты перетаскивания (Drag-and-Drop) ... 78
Атрибуты отношений ... 78
Применение WAI-ARIA ... 80

Web с нечеловеческим лицом (микроформаты и микроданные) ... 81
Когда тегов не хватает ... 81
Микроформаты ... 82
Технология RDFa ... 85
Микроданные ... 89
Microdata DOM API ... 96

Canvas – холст для рисования на веб-странице ... 97
Черный ректангл ... 97
Использование примитивов ... 99
Начинаем рисовать ... 103
Работа с изображениями ... 111
За каждый пиксель! ... 117
Трансформации ... 119
Интерактивность и анимация ... 122
Свой Paint ... 122
Как нам организовать анимацию? ... 124
Play the Game! ... 127
Библиотеки для работы с Canvas ... 130

SVG – векторная графика в www ... 133
Рисуем тегами ... 133
Кривая вывезет ... 138
Группируй и властвуй ... 143
Усложняем жизнь – элементы symbol и image ... 145
SMIL – язык анимации SVG ... 147
Библиотеки для работы с SVG ... 150
Canvas vs SVG ... 151

WebGL – врываемся в третье измерение ... 153
Браузеры и драйверы ... 153
Шейдеры ... 159
Наконец-то 3D ... 164

Теперь в движении ... 166
Объем ... 168
Текстура и освещение ... 171
Инструментарий для работы с WebGL ... 179

Храним данные на клиенте – WebStorage/WebSQL/WebNoSQL ... 184
WebStorage – хранилище «ключ/значение» в браузере ... 184
WebSQL – реляционная база данных на веб-странице ... 185
IndexedDB – NoSQL в вебе ... 187

AppCache – управляем кэшированием вплоть до полного offline! ... 193

File, FileSystem и полный drag'n'drop ... 196
File API – Ура! Свершилось! ... 196
FileSystem API ... 200
Все это drag'n'drop! ... 205

Сервер, я здесь ... 215
Server-Sent Events – сервер тебя не оставит ... 216
Web Messaging – легальный XSS ... 219
XMLHttpRequest 2 ... 223

Звуки audio ... 227
MediaElement – медиаплеер на HTML ... 229
WebAudioAPI ... 234

Video ... 240

WebRTC – коммуникации через веб-страницу ... 243

Geolocation API. Непростой вопрос собственного местонахождения ... 250
Где это я? ... 250
Позиционируем с помощью Google Maps ... 252
Откуда? ... 253
Вглубь Geolocaton API ... 254

WebWorkers – судьба сетевого пролетариата ... 257
Параллельные вычисления на веб-странице ... 257
Sharedworker'ы – надо делиться ... 263

WebSockets – забудем про HTTP? ... 267
Web Sockets – TCP для веба ... 267
WebSocket-серверы ... 269
Работаем с phpDaemon ... 270

Web Intents – средство общения приложений 278

Web Speech API – счастье, когда тебя понимают (распознавание речи на веб-странице) 282

Поговорим с веб-интерфейсом 282
Ничего не получается! Ошибки 288
Родная речь 290
А поговорить? SpeechSynthesis 291

MathML – история с математикой 294

Введение в MathML 294
Presentation MathML – разметка представления 298
Content MathML – «содержательная» разметка 305
А теперь все вместе! (Смешанная разметка) 309

Прочие полезные вещи 312

События колесика мыши 312
Full-Screen API 313
Состояние onLine 314
Page VisibilityAPI 314
History Api 315
RequestAnimationFrame – решение проблем JavaScript-анимации 317
Prerendering – отрисовываем страницы заранее 318
Selectors API – простой синтаксис доступа к DOM-элементам 319
Расширения DOM 320
Web Notifications API – настоящие pop-up'ы 320
Mouse Lock/Pointer Lock API 323

Vibration API есть? А если найду? HTML5 для мобильных устройств 325

Battery Status – API, продиктованный жизнью 325
А как насчет ориентации? Device Orientation API 327
Ориентация экрана – объект Screen и его Orientation API 332
«I'm pickin' up good vibrations» – Vibration API 334
Теплый ламповый API – Ambient Light Events 334
Proximity Events – они уже рядом 335
Network Information API 336

Mozilla WebAPI – будущее наступило? 338

Приложение 1. Ресурсы для работы с HTML5-технологиями 342

Приложение 2. Спецификации W3C, имеющие отношение к HTML5-технологиям 343

Предметный указатель 345

HTML – это теги

Вопреки тому, что говорилось во введении, все-таки давайте сначала вспомним, что HTML, по крайней мере когда-то, был языком разметки и состоял из тегов и их атрибутов. Само собой, он и сейчас из них состоит, просто не они теперь играют главную роль в представлении и функционировании веб-приложений. Тем не менее именно теги были и остаются основой построения HTML-документов. В новом стандарте их число несколько изменилось, в основном в сторону увеличения, но это не самое главное. Впрочем, обо всем по порядку.

Структура страницы

Итак, начнем погружение в HTML5, и начнем его с самого верха страницы. С тега HTML? Нет, еще выше, с DOCTYPE! Именно здесь нас ждет самое радикальное и, наверное, самое приятное изменение, ради одного которого уже стоило начинать революцию. DOCTYPE теперь выглядит так:

```
<!DOCTYPE html>
```

Это все, и мне кажется это прекрасно! Про PUBLIC «-//W3C//DTD XHTML 1.0 Strict//EN», xhtml1-strict.dtd и прочее можно забыть как про страшный сон. Показательно отсутствие версии html – согласно концепции WHATWG, отказывавшейся от упоминания версии, определяет стандарт по мере его развития, то есть такой тип включает все любые типы html- и xhtml-документов, в том числе и будущих версий.

Продолжим строить страницу:

```
<!DOCTYPE html>
<html lang="ru" dir="ltr" >
  <head>
    <title>HTML5 - путеводитель по технологии</title>
  </head>
```

```
  <meta charset = utf-8>
  Hello HTML!
</html>
```

Следующий приятный сюрприз – синтаксис тега <meta>. Прежний синтаксис, включающий http-equiv = "..." content = "...", все еще правомерен, но писать по-новому немного приятней, правда? И номер версии HTML указывать нет необходимости – есть просто HTML, и все!

Да, наверное, все заметили, что сам вышеупомянутый тег расположен в непривычном месте (внутри контейнера <head></head>). Это не ошибка – новый стандарт позволяет размещать метаинформацию где угодно. Конечно, лучше ее поместить в наиболее удобном и читаемом месте, но не всегда это будет секция заголовка (в дальнейшем нам будут встречаться примеры таких решений). Кстати, секция <head> вовсе не обязательна для валидности, равно как и опущенная в нашем примере секция <body></body>.

Впрочем, тег именно «опущен», за отсутствием необходимости. Браузер считает, что он все равно незримо существует, в чем нетрудно убедиться, проанализировав данную разметку в Mozilla Firebug или в Google Chrome Inspector (рис. 7). Этот и некоторые другие (<head>, <html>) всегда «подразумеваются», хотя для использования в CSS- или в JavaScript-сценариях их надо прописывать в явном виде.

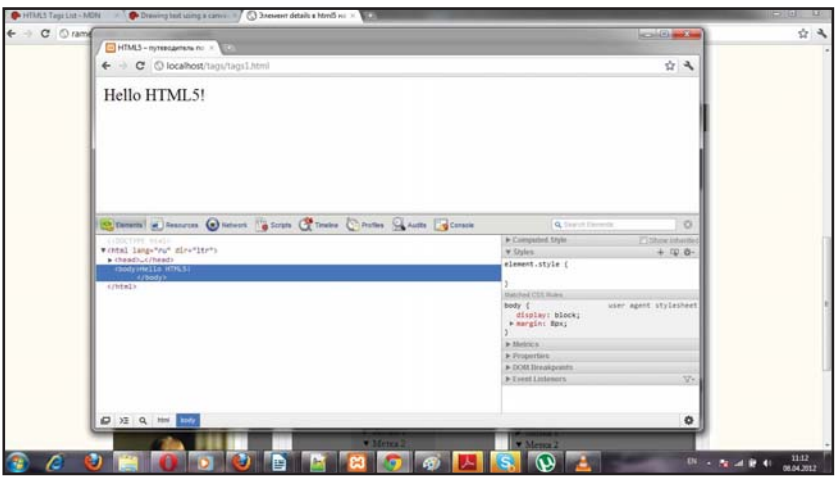

Рис. 7. Браузер дорисовывает необходимые элементы

Да, новый стандарт предполагает довольно большую свободу – я уже не буду говорить об отсутствии необходимости кавычек у атрибутов и закрывающего слэша у одиночных тегов. Правда, это не освобождает нормального разработчика придерживаться определенных стандартов кодирования. Постараемся избежать эклектичности в написании и мы в дальнейших примерах. А пока еще приятная мелочь – тег <a> теперь означает блочный элемент и может выполнять функции группировки:

```
<a href="/item.php&id=5">
  <h2>Швабра</h2>
  <p>Необходимый инструмент для дома и самообороны</p>
  <img src = "shvabra.jpg">
</a>
```

(если вы верстали именно так, то теперь вы можете делать это на законном основании).

Воплощение концепции семантической разметки

В соответствии с подзаголовком я не собираюсь тут излагать теорию семантики в web вообще и семантической верстки в частности, с ней каждому заинтересованному разработчику предлагаю ознакомиться самостоятельно, а мы займемся практикой.

Известно, что веб-страницу любой сложности можно сверстать, используя почти исключительно элементы и . С помощью их можно представить любое содержимое (разумеется, за исключением специфичных мультимедиа-элементов). В реальной жизни стремление к такому подходу выливается в написание кода, подобного этому:

```
<div class="header">
    <div class="title">
      <span class = "Name">
      HTML5 – путеводитель по технологии
      </span>
      <span class = "subName">
      введение в HTML5
      </span>
    </div>
    <div class="logo">
</div>
```

```
<div class="content">
  <div class = "nav" >....</div>
  <div class = articles>
    <div class="article" id ="1">
      <span class = "title">HTML - это теги</span>
      <span class = "datetime">2012-01-08</span>
      <div class = "article_text">
      ......
      </div>
    </div>
    <div class="article" id ="2">
      <span class = "title">Формы HTML5</span>
      <span class = "datetime">2012-02-02</span>
      <div class = "article_text">
      ......
      </div>
    </div>
  </div>
</div>
<div class="footer">
  <span class="copyright">curveSoft inc</span>
<div>
```

Знакомая картина? Тут все по-человечески понятно – верстальщик, вынужденный делать различие между элементами разметки по их содержимому, присваивает им классы с чепловекопонятными и, возможно, в пределах некоего рабочего коллектива унифицированными именами (селекторами). Этот способ позволяет ориентироваться в разметке человеку, но вот браузеру узнать, что в `<div class="nav">` скрывается именно меню сайта, не представляется возможным. А кроме браузеров, существуют еще поисковые роботы, различные контент-анализаторы и программы экранного доступа, для правильного понимания которыми содержимого веб-страницы очень важна понятная, семантическая разметка.

Явно назрела необходимость в переменах, и они произошли. Причем совсем не внезапно, а довольно планомерно.

Компания Opera Software с 2004 года ведет разработку интересного средства исследования веб-среды – поисковой системы, индексирующей структуру и элементы объектной модели сайтов. Она называется Metadata Analysis and Mining Application (MAMA). С помощью этого инструмента было проведено немало интересных исследований, но нас сейчас интересует простой параметр – наиболее употребляемые наименования классов в html-разметке.

В настоящий момент эта таблица выглядит следующим образом (http://devfiles.myopera.com/articles/572/):

1	footer	179528	21	Left	47822
2	menu	146673	22	style5	47645
3	style1	138308	23	right	45855
4	msonormal	123374	24	date	44613
5	text	122911	25	contentpaneopen	44395
6	content1	13951	26	moduletable	44188
7	title	91957	27	link	43629
8	style2	89851	28	blog	42469
9	header	89274	29	bodytext	40450
10	copyright	86979	30	style6	39496
11	button	81503	31	mainlevel	38993
12	main	69620	32	contentheading	38982
13	style3	69349	33	top	37720
14	small	68995	34	normal	37101
15	nav	68634	35	inputbox	36342
16	clear	68571	36	article_seperator	35366
17	search	59802	37	style7	34710
18	style4	56032	38	news	34543
19	logo	48831	39	navbar	33912
20	body	48052	40	links	33830

Тут показаны результаты по частоте используемых значений атрибута `class` для 2 148 723 случайным образов выбранных url.

Для полной картины посмотрим еще результаты по атрибуту `id`:

1	footer	288061	21	layer3	42825
2	content	228661	22	form1	42119
3	header	223726	23	autonumber2	41960
4	logo	121352	24	table3	41504
5	container	119877	25	home	41040
6	main	106327	26	copyright	38893
7	table1	101677	27	page	37274
8	menu	96161	28	layer4	35327
9	layer1	93920	29	image2	35215
10	autonumber1	77350	30	left	34953
11	search	74887	31	searchform	33184
12	nav	72057	32	__viewstate	32714
13	wrapper	66730	33	table_01	32540
14	top	66615	34	table4	31583
15	table2	57934	35	map	30269
16	layer2	56823	36	active_menu	30243
17	sidebar	52416	37	right	30206
18	image1	48922	38	image3	29759
19	banner	44592	39	news	29078
20	navigation	43664	40	body	29037

Тут данные собраны с 1 806 424 url-адресов.

Результаты, по-моему, довольно интересны по нескольким показателям, но главное тут то, что названия используемых нами в верстке примеров (header, content, footer) имеют довольно высокий рейтинг – в первую двадцатку вошли различные вариации на темы навигационного меню (menu, nav, navigation, active_menu, navbar) и специфических элементов, часто встречающихся на веб-сайтах (search, copyright, sidebar). Руководствовались ли этими данными разработчики HTML5, неизвестно (впрочем, скорее да), но новые теги, появившиеся в языке разметки, во многом совпадают с этими классами – если не по названию, то по назначению. Переверстаем наш пример с использованием HTML5:

```
<header>
  <hgroup>
    <h1>HTML5 - путеводитель по технологии</h1>
    <h2>Ведение в HTML5</h2>
  </hgroup>
  <div class="logo"></div>
</header>
<nav>...</nav>
  <section id="articles">
    <article id = 1 >
      <span class = "title">HTML - это теги</span>
      <time pubdata datetime="2012-01-08">8 января</time>

      <div class = "article_text">
      ......
      </div>
    </article>
    <article id = 2 >
      <span class = "title">Формы HTML5</span>
      <time pubdata datetime="2012-02-02">2 февраля</time>
      <div class = "article_text">
      ......
      </div>
    </article>
  </section>
<footer>
  <span class="copyright">curveSoft inc</span>
</footer>
```

Как видите, не все удалось заменить новыми элементами, разговор о семантике мы продолжим в главе, посвященной микроданным, но и сейчас верстка стала более читаемой и логичной.

Давайте рассмотрим новые теги подробнее.

Прежде всего бросаются в глаза контейнеры <header></header> и <footer></footer>. Их назначение понятно по названию – они содержат в себе «шапку» и «подвал» страницы (мне такие определения кажутся более уместны, чем верхний и нижний колонтитулы). Возникает вопрос: как содержимое, ограниченное этими тегами, должно отображаться в браузере? И вот тут важно понять (если вы хотите заниматься разработкой на HTML5) одну простую и важную вещь. Какого-либо отображения внешнего вида элементов HTML5 в стандарте не предусмотрено! То есть разметке важно сообщить, что ограниченная <header> область – это шапка страницы. Что делать с этой ценной информацией браузеру – дело исключительно браузера, а точнее, его производителей. Такой подход вполне оправдан – при современном разнообразии устройств для просмотра веб-страниц даже самые каноничные элементы разметки нужно отображать очень по-разному, скажем, на экране плазменной панели или коммуникатора.

Теги <header> и <footer> могут быть не только верхнего уровня, допустимо, когда эти элементы встречаются у каждой статьи или, например, цитаты.

Вооружившись новым пониманием сути разметки, идем дальше.

Элемент <nav> предназначен для отображения на странице навигационного меню. Как? – см. два абзаца выше. Автоматически его содержание в список преобразовываться, скорее всего, не будет, и потребуются теги <ul> и <li>. Можно также включать внутрь этого контейнера заголовки, картинки и придавать ему любой внешний вид, это не меняет сути – это навигационный блок.

Тег <hgroup> создан для обозначения группы заголовков. Он нужен для того, чтобы все подзаголовки, возможно, картинки и гиперссылки внутри этого контейнера, воспринимались как единая логическая единица.

Стоп! По-моему, мы что-то пропустили. Где тег <content>? Ну или <main>, <page>, в общем, контейнер, заключающий основное содержание страницы? Его нет. И это не упущение разработчиков, а их продуманная позиция. Суть ее в том, что content – это просто то, что не занято контейнерами <header> и <footer>.

В нашем примере таким тегом, несущим основное содержание страницы, служит тег <section>, но предназначен он совершенно не для этого. Использование данного элемента на самом деле обычно вызывает вопросы и даже ошибки, хотя тут все просто – он дей-

ствительно делит на секции. Что именно? В данном случае страницу (за секцией статей может следовать секция анонсов, например, затем секция новостей). Но ничто не мешает использовать его же для порционирования содержимого внутри статьи.

Для самих статей вводится специальный тег <article>. Он обрамляет независимые фрагменты текста, способные существовать и вне контекста страницы. Например, это могут быть записи в блоге, видеоролики и прочее содержимое, которое может быть объединено, как новости в новостную ленту.

Тег <time>, ответственный за разметку указания даты/времени, приносит дополнительные возможности их отображения. Фактически сама временная отметка может быть обозначена в каком угодно формате, а вот в атрибуте datetime время должно быть указано в ISO-стандарте, обеспечивая ее правильное понимание разными роботами (например, Atom/RSS-агрегатором). Булевый атрибут pubdata означает, что указанное время является временем публикации контента.

Кстати, <time> – это тот тег, на котором разработчики начали задумываться, не стоит ли остановиться? Подробнее о проблеме сказано в главе о микроформатах, а тут ограничимся только тем, что на момент написания этих строк данный элемент был на грани исключения из спецификации.

За пределами нашего примера остался такой немаловажный элемент, как <aside>, предназначенный для так называемых «врезок» – текстовых фрагментов, имеющих отношение к основному контенту страницы, но не вписывающихся в ее структуру. Например, это может быть биографическая справка об упомянутом в статье персонаже или список ссылок на статьи, близкие по тематике, помещенный сбоку от основного материала.

Всякие полезности

Теги <figure> и <figcaption> приняты для разметки изображения на веб-странице. Нет, старый добрый <img> никуда не делся, но теперь устраняется досадное недоразумение, заключающееся в том, что рисунок и подпись к нему представляют два совершенно различных и не связанных по умолчанию элемента разметки. Теперь их можно поместить в один контейнер, не сдерживая форматирования в подписи, а само изображение может состоять из нескольких частей. Пример использования <figure>:

```
<figure>
  <img src="logo.png" alt="HTML5 logo" />
  <figcaption>
    Логотип HTML5<br>
    <a href="http://upload.wikimedia.org/wikipedia/commons/6/6e/HTML5-logo.svg">
      Вариант в svg формате
    </a>
  </figcaption>
</figure>
```

Тег <details> создан для отображения скрытого, а вернее разворачиваемого контента. Он представляет из себя простейший виджет-«аккордеон», конструкцию, обычно реализуемую на javaScript. Пример такой конструкции показан в листинге ниже:

```
<details>
  <summary>Ссылки по теме</summary>
    <a href ="http://whatwg.org/htm" >WHATWG</a><br>
      <a href ="http://www.w3.org/TR/html5" >W3C/HTML5</a><br>
      <a href ="http://dev.w3.org/html5/spec/" >Стандарты</a><br> </details>
<details>
  <summary open >Логотип</summary>
  <img src = "http://upload.wikimedia.org/wikipedia/commons/thumb/6/6e/HTML5-
logo.svg/120px-HTML5-logo.svg.png" />
</details>
<details>
  <summary>Описание</summary>
  Введение в HTML5<br>
  <details>
  <summary>Новые теги</summary>
  article<br>
  aside<br>
  audio<br>
  canvas<br>
  command<br>
  details<br>
<details>
  Geolocation API<br>
</details>
```

Результат можно видеть на рис. 8. Элемент <summary> – еще один новый тег, он используется для задания заголовка содержимому details. Булевый атрибут open отвечает за начальное раскрытое по-

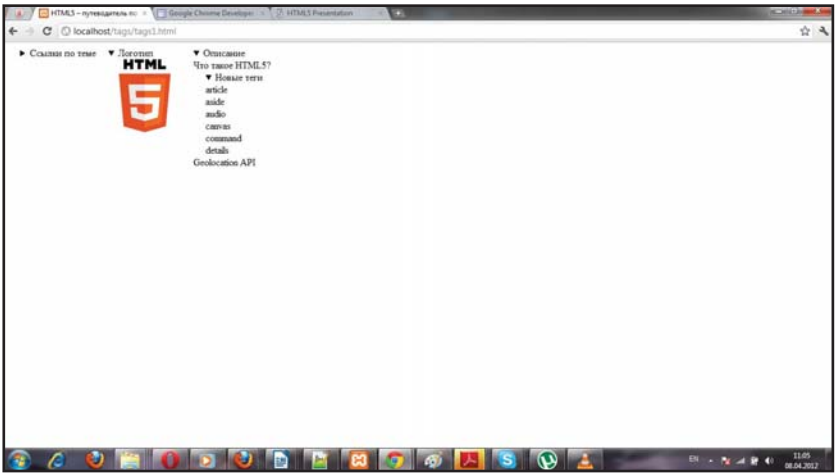

Рис. 8. Работа элементов details и summary

ложение «аккордеона». Из примера видно, что внутри контейнера можно использовать различного рода содержимое, а теги `<details>` могут быть вложены друг в друга.

Использование тега `<embed>` для flash-контента практиковалось давно, правда, не совсем правомерно. Теперь это легализовано – этот элемент в HTML5 служит для представления содержимого нативно не поддерживаемых браузером форматов, обычно требующих установки дополнительных плагинов. Адрес такого плагина уточняется значением необязательного атрибута `pluginspage`:

```
<embed src= "stray_cats.swf"
  width="200"
  height="100"
  type="application/x-shockwave-flash"
  pluginspage="http://www.macromedia.com/go/getflashplayer">
```

Тег `<menu>`, строго говоря, не нов – он присутствовал ранее в спецификации HTML, но в четвертой версии был признан устаревшим, уступив место списку (`<lu>`). Теперь он получил новую жизнь и используется совместно с новым тегом – `<command>`, создающим команду в виде переключателя, флажка или обычной кнопки:

```
<menu>
  <command onclick="alert('Старт')"  label="Старт">Старт</command>
```

```
  <command onclick="alert('Стоп')" label="Стоп">Стоп</command>
  <command onclick="alert('Пауза')" label="Пауза">Пауза</command>
</menu>
```

К сожалению, эта конструкция работает только в браузере Internet Explorer 9.0. Следующий новый элемент, <mark>, поддерживается всеми, и его полезность, в отличие от предыдущего, не вызывает вопросов. Он предназначен для выделения фрагмента текста:

```
поддерживается <mark>всеми</mark> и его полезность
```

В общем случае такое обрамление не должно никак проявляться внешне, предоставив менять свой вид и поведение JavaScript и CSS, но в браузерах Google Chrome и Mozilla Firefox такие фрагменты подсвечены желтым фоном. Должно быть, на всякий случай.

Еще один полезный элемент, <ruby>, в русскоязычном сегменте Сети вряд ли будет сильно востребован. Но современный мир со страшной скоростью глобализуется, и ко всему надо быть готовым. Вообще, тег предназначен для добавления небольшой аннотации сверху (как правило) от нужного текста. Сам элемент обрамляет анотируемый текст (обычно один или два символа) и контейнер <tr></tr>, содержащий саму аннотацию. Зачем это все нужно? В русском языке незачем, а вот в японском есть такое понятие, как фуригана – фонетические подсказки, поясняющие произношение. Выглядит это так:

```
<p lang="zh-CN">
<ruby>
  <rt>hàn</rt>
  <rt>zi </rt>
    <rt>tokyo</rt><rp>tokyo</rp>
</ruby>
</p>
```

Результат – на рис. 9.

Необязательный тег <rp> показывает текст, отображаемый браузером в случае отсутствия поддержки <ruby>.

Наконец, последний новый тег общего назначения – <wbr>. Он несет довольно простую функциональность – указывает браузеру место, где допускается делать перенос строки в тексте, если тот не помещается в родительском элементе.

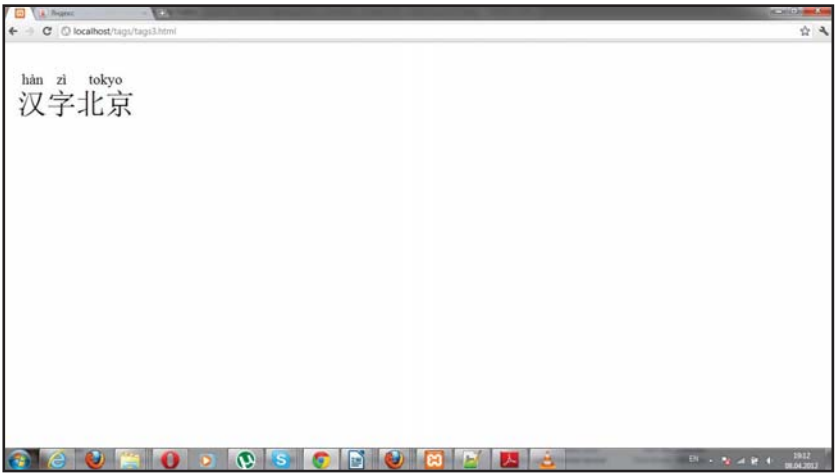

Рис. 9. Загадочная фуригана

На этом с новыми тегами HTML5 мы закончили. Правда, осталось еще более десятка новых элементов, имеющих свое специальное назначение, мы познакомимся с ними по мере освоения различных аспектов технологии.

Впрочем, и эта глава еще не закончена – у тегов, как правило, есть атрибуты, и там тоже появилось много нового.

Атрибуты и аксессуары

Прежде всего о глобальных атрибутах. Хотя, конечно, этот термин довольно нечеткий и неофициальный, но тут под ним принимаются атрибуты, присутствующие у большего числа элементов. Это, например, атрибуты `class` или `id`. В HTML5 их стало несколько больше.

Официально введен в спецификацию довольно давно предлагаемый Microsoft атрибут contenteditable. На самом деле достаточно симпатичная идея, поначалу встретила некоторые трудности в кроссбраузерной реализации, но теперь она поддержана всеми ведущими производителями. Суть очень проста – при установке этого атрибута содержимое элемента становится доступным для редактирования в браузере:

```
<div contenteditable >
  The idea of these new types is that the user agent can provide the
user interface, such as a calendar date picker or integration with
```

```
the user's address book, and submit a defined format to the <span
style="color:green">server</span>. It gives the user a better experience as his
input is checked before sending it to the server meaning there is less time to
wait for feedback.
</div>
```

Результат – на рис. 10: я выделил слово «server» и применил copy/paste.

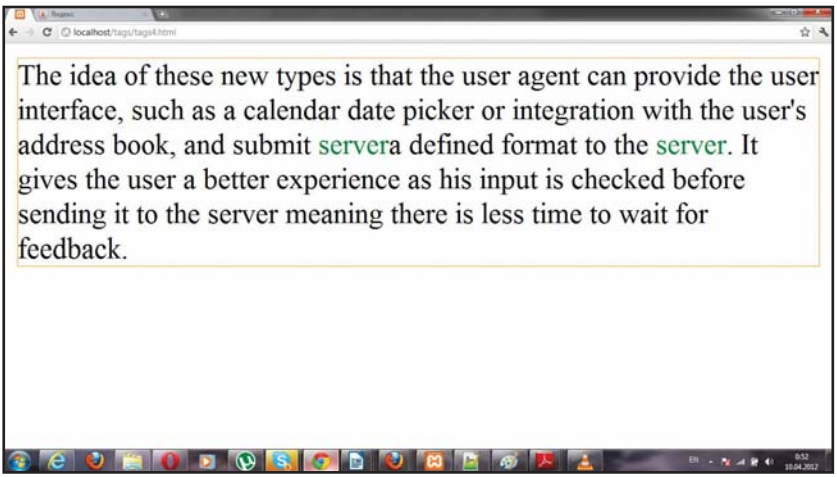

Рис. 10. Contentediteble в действии

Со следующим глобальным атрибутом, contextmenu, добавляющим контекстные меню к любым элементам на странице, ситуация в настоящее время не очень понятна. Он также выдвинут Microsoft и, судя по спецификации, должен был применяться следующим образом:

```
<p>
  <img src="http://html5.png"     alt="" contextmenu="edit">
</p>
<menu type="context" id="edit">
  <li>Вырезать</li>
  <li>Копировать</li>
  <li>Вставить</li>
  <li>Редактировать</li>
  <li>Выравнивание</li>
</menu>
```

Но не работает. Ни в Internet Explorer, ни в любом другом браузере. Реализация контекстного меню HTML-разметкой существует только для Mozilla Firefox с помощью нестандартизированного тега <menuitem>. Естественно, нигде больше это не работает, но сам подход, по-моему, настолько удачен, что имеет все шансы стать стандартом, судите сами (чуть модифицированный пример с http://www.whatwg.org):

```
<!DOCTYPE html>
<html  dir="ltr" >
  <head>
    <title >HTML5 – путеводитель по технологии</title>
    <style>
      img { -moz-transition: 0.2s; }
      .rotate { -moz-transform: rotate(90deg); }
      .resize { -moz-transform: scale(0.7); }
      .resize.rotate { -moz-transform: scale(0.7) rotate(90deg); }
    </style>
    <script>
      function rotate() { document.querySelector("img").classList.
toggle("rotate"); }
      function resize() { document.querySelector("img").classList.
toggle("resize"); }
    </script>
  </head>
  <body>
    <div contextmenu="supermenu">
      <img src="http://html5.png">
    </div>
    <menu type="context" id="supermenu">
      <menuitem label="rotate" onclick="rotate()" icon="http://cdn1.iconfinder.
com/data/icons/silk2/arrow_rotate_clockwise.png"></menuitem>
      <menuitem label="resize" onclick="resize()" icon="http://cdn3.iconfinder.
com/data/icons/fugue/icon/image-resize.png"></menuitem>
      <menu label="share">
        <menuitem label="twitter" onclick="alert('foo')"></menuitem>
        <menuitem label="facebook" onclick="alert('bar')"></menuitem>
      </menu>
    </menu>
  </body>
</html>
```

Результат – на рис. 11.