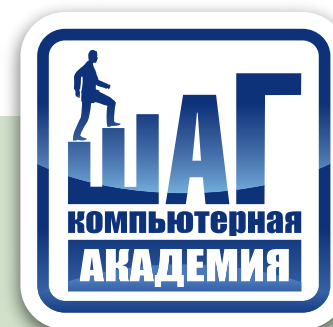


Создание статичных веб-страниц
с помощью языков XHTML и CSS

HTML5. Что нового?



Принципы HTML5

Новая разметка

Аудио и видео

Формы 2.0

Содержание

Введение	3	Аудио.	18
Предыстория	4	Видео.	20
Принципы HTML5.	8	Формы 2.0.	21
Новая разметка	9	Новые атрибуты	21
DOCTYPE	9	Новые значения атрибута type для тега <input>	22
Другие упрощения в заголовке	9	Другие возможности	24
Устаревшие теги и атрибуты	10	Дополнительные материалы	25
Новости для тега <a>	11		
Новые теги	12		
Canvas	14		
Основы рисования.	15		
Другие возможности	17		



Введение

Самым популярным сервисом Интернет является Всемирная паутина (World Wide Web). В основе WWW лежит *гипертекст* — текст, содержащий специальные активные участки, способные ссылаться на другие документы, т.е. гиперссылки. Для форматирования гипертекста используются языки разметки (Markup Languages). Главным и наиболее используемым из них является HTML (Hyper Text Markup Languages). Этот язык существует уже почти 20 лет и, за это время, претерпел немало изменений. Последний действующий стандарт (спецификация) носит название HTML 4.01 и ему уже более 10 лет.

Тем не менее, уже несколько лет ведется разработка пятой версии языка — HTML5. Для того, чтобы понять суть предлагаемых в этой версии нововведений, нужно немного разобраться в истории.



Предыстория

Итак, автором языка HTML является сэр Тим Бернерс-Ли (Tim Berners-Lee). Началось развитие языка с его работы «HTML Tags», опубликованной в 1991 году. Именно там были изложены первые два десятка тегов, которые стали использоваться для создания первых сайтов.

Эти первые теги были созданы Бернерсом-Ли в рамках выполнения задачи, поставленной ему в CERN: **создать единое информационное пространство для навигации по накопленным в организации научным документам.**

Ключевое слово здесь — *научным*. Бернерсом-Ли был создан механизм форматирования сухих академических статей, рефератов, отчетов на основе структуры заголовков, абзацев, списков и т.д., не нуждавшихся в богатом визуальном оформлении.

Дальнейшая история привела к тому, что проект Тима Бернерса-Ли получил широкую мировую известность и стал использоваться не совсем по назначению. Уже через пару лет стал очевиден коммерческий потенциал изобретения. Основную массу сайтов составляли отнюдь не собрания научных трудов, а рекламные коммерческие объявления, предложения и т.д.

Стандарта HTML 1.0 никогда не существовало. Информация о синтаксисе языка передавалась из уст в уста.

Тем не менее, после того, как образовался рынок браузеров — после выпуска браузера NCSA Mosaic (1993, Марк Андриессен), стало ясно, что тех тегов, которые предложил сам Бернерс-Ли, мало. Тогда разработчики браузеров стали разрабатывать теги самостоятельно. Ведь, по сути, дело не столько в языке разметки, а в том, как реагирует браузер на тот или иной тег.



Тим Бернерс-Ли



Марк Андриессен

Поэтому в 1994 году IETF (Internet Engineering Task Force, Специальная комиссия интернет-разработок), по заказу новообразованного W3C (World Wide Web Consortium), выпустила спецификацию HTML 2.0. Она «узаконивала» некоторые новые теги, внедренные усилиями разработчиков браузеров; например, тег ``, придуманный в Netscape.

С этого момента началось довольно бурное развитие языка. В течение следующих 4 лет он пережил выпуск 4 версий:

- HTML 3.0 — март 1995
- HTML 3.2 — январь 1997
- HTML 4.0 — декабрь 1997
- HTML 4.01 — декабрь 1999

Такая активность вызвана конкурентной борьбой на рынке браузеров, которая вошла в историю как Война Браузеров.

А далее в этой истории произошёл ключевой поворот.

В 2000 году вышла спецификация XHTML 1.0, которая призвана была стать новой версией языка разметки.

Сама по себе спецификация для XHTML 1.0 практически ничем не отличалась от HTML 4.01. Не добавлялись никакие новые теги. Суть изменений была только в синтаксисе — в XHTML требовалось соблюдать правила языка XML, — куда более жесткого и требовательного к оформлению. Цель была в том, чтобы в будущем предусмотреть совместимость веб-сайтов с программным обеспечением, т.е. веб-сервисами.

Жесткие правила, впрочем, сказались весьма позитивно. Повысилось качество разметки. Сопровождать сайты стало легче. Выход XHTML 1.0 совпал с возросшим уровнем поддержки браузерами CSS. Синтаксис XHTML получил репутацию лучшего способа разметки и укрепил W3C в правильности выбранного пути.

В 2001 году вышла версия XHTML 1.1, которая приблизила HTML к XML еще сильнее. Проблемой стало то, что Internet Explorer его практически не поддерживал. Таким образом, наметилось серьезное противоречие между идеями Консорциума и реальным положением вещей.



Логотип W3C

За последние 9 лет в этом направлении серьезных сдвигов не произошло.

Следующим шагом в разработке стандартов должен был стать XHTML 2, работа над которым началась еще в 2003 году. Новости от разработчиков шокировали: **XHTML 2 предлагалось сделать обратно-несовместимым со своими предшественниками и старыми версиями HTML** (а значит — со всем существующим содержимым Интернета).

Это привело к расколу в рядах идеологов в W3C.

В 2004 разработчики браузеров — Opera, Apple, Mozilla, предложившие, в лице Яна Хиксона (Ian Hickson), вернуться к разработке языка, удовлетворяющего нуждам разработчиков, и получившие отказ, основали WHATWG (Web Hypertext Application Technology Working Group, www.whatwg.org) для разработки своей версии стандарта, которую они называли «HTML5».

В октябре 2006-го Тим Бернерс-Ли в своем блоге признал, что идея перевести сеть на XML была неверной. После этого, Консорциум принял решение все дальнейшие наработки по

языкам разметки основывать на наработках WHATWG.

Возникла слегка запутанная ситуация. Какое-то время W3C одновременно работал над двумя совершенно несовместимыми языками разметки — XHTML 2 и HTML 5 (обратите внимание: аббревиатура с пробелом), — в то время как WHATWG, отдельная организация, занималась спецификацией HTML5 (без пробела), которая должна была стать основой для другой спецификации в W3C. Конфликт был разрешен в 2009, когда W3C объявил, что проект XHTML 2 закрыт.

Подведем итог этой истории.

За почти 20-летнюю историю язык разметки HTML пережил многое, но все это время он продолжал оставаться тем, чем был изначально — языком описания научных документов. Современный WWW это уже очень давно совсем не научные документы. Если сайт образца 1993 года — это набор связанных между собой тестовых документов с картинками, то современный сайт — это полнофункциональное программное обеспечение. Другими словами



Логотип WHATWG

— веб-сервис (web application). Возможности HTML 4 слишком тесны для современных веб-разработчиков.

Работа WHATWG по разработке новой спецификации направлена на то, чтобы создать язык разметки веб-сервисов, а не академических текстов.

Давайте посмотрим, как решается эта задача.



Принципы HTML5

Главный принцип можно сформулировать как: «Мостить натоптанные тропинки» или «Поддерживать то, что уже существует»

Работа W3C всегда строилась по принципу «сначала проверяем, потом запускаем» и была ориентирована на «демократические» подходы в обсуждениях. Т.е. ставится вопрос, долго обсуждается и поднимается на голосование. В WHATWG все почти наоборот. Есть главный редактор — Ян Хиксон (ранее — сотрудник Opera, ныне — Google). Он принимает или отклоняет решения. Работа строится по принципу «запускаем, затем проверяем».

Немалое влияние на разработку спецификации оказывает список рассылки WHATWG и отзывы производителей браузеров о положениях спецификации. Кстати, Microsoft, очень слабо, по словам Хиксона, участвует в обсуждении спецификации.

Нужно отметить 2 ключевых момента:

1. HTML5 включает (в отличие от HTML 4) в себя указания веб-дизайнерам (авторам) и производителям браузеров (разработчикам). Т.е., отныне, в спецификации будет задокументировано поведение браузера при попытке отображения несуществующего тега, синтаксической ошибки и т.д.
2. HTML5 включает в себя, помимо описания тегов, также спецификацию DOM API, для реализации в JavaScript. Это — большой шаг в сторону унификации и кроссбраузерности.

Таким образом, достойными внимания частями спецификации можно назвать следующие:

1. Новый DOCTYPE.
2. Новые теги и придание нового смысла старым.
3. Введение тега `<canvas>`.
4. Работа с аудио и видео.
5. Новые виды полей в формах.

Новая разметка

В первую очередь, HTML5 — это язык разметки, поэтому рассмотрим изменения в этой сфере.

DOCTYPE

HTML5 вводит новый, более простой, Doctype (Document Type Declaration) — указание языка разметки, используемого на странице.

Задача HTML5 — поддерживать существующие HTML 4.01 и XHTML 1.0 страницы. Новые версии HTML, в свою очередь, должны будут поддерживать HTML5. Таким образом, какое-то детальное указание DTD становится явно излишним. Тем более, браузеры никогда особо и не пользовались этим страшным URL для загрузки корректной DTD. Doctype, по сути, нужен вообще только для валидации

Doctype для HTML 4.01:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3c.org/TR/html4/strict.dtd">
```

Doctype для XHTML 1.0:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict //EN" "http://www.w3c.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

А это – Doctype для HTML5:

```
<!DOCTYPE html>
```

Указание Doctype для разных версий языка HTML

страницы. И еще, поскольку HTML5 описывает отдельно требования для авторов и отдельно для разработчиков — нет нужды в версиях Strict или Transitional.

Другие упрощения в заголовке

Вопросы задания кодировки встали в HTML не сразу и всегда решались с помощью специального meta-тега. Так выглядело указание кодировки раньше:

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

То есть не было даже отдельного специального тега для кодировки. Задание кодировки



шло как дополнение к заданию MIME-типа документа.

В HTML5 задание кодировки выглядит так:

```
<meta charset="UTF-8">
```

Как видим, необходимость в указании MIME-типа отпала совсем.

И не только в `<meta>`! Разрешено не уточнять MIME-тип и в остальных случаях, т.е. при подключении скриптов и стилей.

Подключение скриптов и стилей в HTML5:

```
<script src="file.js"></script>
<link rel="stylesheet" href="file.css">
```

Обязательно нужно упомянуть также, что строгость синтаксиса XHTML теперь не требуется. Авторы вольны форматировать свою разметку в обычном HTML стиле. Разработчики просто рассчитывают на то, что за последние 10 лет уже сформировалось новое поколение авторов, воспитанных на требованиях XHTML и они не будут культивировать небрежную разметку в HTML5.

Устаревшие теги и атрибуты

В HTML5 избавились от термина *устаревший* (deprecated). Напомню, в HTML 4 и последующих версиях так назывались теги (или атрибуты), которые не рекомендовалось использовать и планировалось исключить их следующей спецификации. Кроме того, были еще и *исключённые* (obsolete), использовать которые было прямо запрещено.

Отныне, поскольку есть часть спецификации адресованная авторам и часть — для разработчиков браузеров, то имеются только устаревшие теги (атрибуты), о которых не говорится в авторской части, зато описывается поведение браузера для этих тегов (атрибутов) в части для разработчиков.

Итак в HTML5 признаны устаревшими такие теги:

- `<frame>`, `<frameset>` и `<noframes>`.
- `<acronym>`; вместо него используется тег `<abbr>`.
- оформляющие теги ``, `<big>` и `<center>`.
- атрибуты `bgscolor`, `cellspacing`, `cellpadding` и `valign`.



Интересно, однако, что не все из оформляющих тегов и атрибутов были записаны в устаревшие. К примеру, элемент `<big>` устарел, но вот `<small>` — нет. Его предназначение изменилось, теперь это не физическое уменьшение размера шрифта, а семантический «мелкий шрифт» (fine print).

Оставлен также тег ``, который раньше означал просто «полужирный текст». Теперь это — «текст, стилистически выделенный из основного потока, но не несущий дополнительной смысловой значимости». Когда смысловая значимость таки присутствует, следует использовать тэг ``.

Так же, тег `<i>` теперь не означает «курсивный». Это — «произнесенное с другой интонацией или с другим настроением». Опять же, без особого значения или ударения. Для того, что подчеркнуть значимость (логическое ударение), используется тэг ``.

Тег `<s>` теперь означает данные, которые устарели или потеряли актуальность, а не просто «зачеркнутый».

Значение элемента `<cite>` тоже было слегка изменено. Если раньше он означал «ссылка на источник», то теперь это — «заголовок

источника». Очень часто при цитировании источником будет как раз заголовок книги, фильма или чего угодно, на что мы ссылаемся. Сама цитата по-прежнему размечается тегом `<blockquote>`.

Новости для тега `<a>`

Тег `<a>` требует нескольких слов отдельно. Этот элемент перестаёт быть просто строчным (линейным). Теперь вовнутрь `<a>` можно поместить целые блоки:

```
<a href="http://ad.example.com/?adid=375&pubid=1422">
  <section>
    <h1>The Mellblom Browser</h1>
    <p>Web browsing at the speed of
      light.</p>
    <p>No other browser goes faster!</p>
  </section>
</a>
```

Естественно, в этом случае тег `<a>` приобретает свойства блочного. Единственное

ограничение — нельзя вкладывать теги `<a>` друг в друга.

Новые теги

HTML5 не только упраздняет старые теги, но и вводит новые. Наконец предложена новая модель семантического деления документа на части. Она основана на накопленном опыте, а не поддержке устаревшей схемы устройства научных документов.

Вот некоторые из добавленных тегов.

`<section>` — тематический раздел документа, обычно сопровождаемый заголовком. Например, глава статьи, или вкладка в диалоговом окне. Одним словом то, что раньше делалось при помощи просто абстрактного тега `<div>`. Рекомендуется все же не злоупотреблять этим тегом и применять его в том случае, если эти разделы упоминаются в оглавлении выше. В противном случае нужно использовать обычный `<div>`.

`<nav>` — раздел страницы, связанный с другими страницами или частями данной. Раздел с навигационными ссылками.

`<article>` — представляет собой автономную часть документа, страницы, приложения или сайта, как независимую, так и часть синдикации. Это может быть сообщение форума, журнальная или газетная статья, запись в блоге, пользовательский комментарий, интерактивный виджет или гаджет, или любой другой независимый элемент содержания. Статьи могут вкладываться друг в друга. Предполагается, что они, в принципе, связаны с содержанием внешней статьи.

`<aside>` — раздел страницы, который состоит из содержимого, имеющего косвенное отношение к основному содержанию, и которое может быть рассмотрено отдельно от основного содержимого. Элемент может быть использован для типографских эффектов, таких как цитаты или врезки, для рекламы, для групп `<nav>` элементов.

`<header>` — заголовочная часть какой-либо страницы. Не путать с `<head>`, который предназначен для решения технических вопросов и размещения указаний браузеру. Здесь может располагаться глобальная навигация, аннотация, введение, логотип и т.д.

`<footer>` — «подвальная» часть какой-либо страницы или статьи, предназначенная для вывода копирайта, автора, ссылок, меток, даты публикации и т.д. Может применяться внутри элемента `<article>` или `<section>`.

Конечно, же элементы страницы, размеченные этими новыми тегами, не будут иметь никакого специального стилевого оформления. Все отдано на откуп CSS.

По сути своей эти элементы являются усовершенствованными тегами `<div>`, получившими семантическое наполнение.

является «родным» для браузера, имеет бинарное устройство и, как следствие, зависит от версии и наличия Flash-плеера, как дополнения к браузеру.

Canvas

HTML5 позволяет преодолеть ограничение, связанное с наличием статических изображений на странице. Да, безусловно, есть возможность вставить анимированное изображение в формате GIF, но ведь им невозможно управлять.

До нынешнего момента прерогатива по управлению динамической графикой в браузере, читай анимацией, принадлежала технологии Adobe Flash, но у неё есть два основных недостатка:

- **Закрытость.** Adobe Flash — это проприетарная технология, авторские права на которую принадлежат компании Adobe, и использование технологии сопряжено с ограничениями.
- **Замещаемость.** Имеется в виду то, что анимационный файл в формате SWF не

HTML5 опирается на технологию SVG (Scalable Vector Graphics), которая предусматривает возможность программной генерации векторных изображений при помощи какого-либо API.

Спецификация HTML5 вводит тег `<canvas>` (разработчик — Apple), который — как следует из его названия — представляет собой холст для программного рисования изображений. В качестве инструментария рисования выступает специальный JavaScript API, который документируется спецификацией в разделе 4.8.11.

Сам по себе тег `<canvas>` очень прост. В атрибутах — размеры холста и идентификатор `id` для адресации из скрипта:

```
<canvas id="my-first-canvas" width="360"
        height="240"></canvas>
```

API (Application Programming Interface) — интерфейс прикладного программирования (иногда *интерфейс программирования приложений*) — набор готовых классов, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) для использования во внешних программных продуктах. Используется программистами для написания всевозможных приложений.

API определяет функциональность, которую предоставляет программа (модуль, библиотека), при этом API позволяет абстрагироваться от того, как именно эта функциональность реализована. Если программу (модуль, библиотеку) рассматривать как чёрный ящик, то API — это множество «ручек», которые доступны пользователю данного ящика, которые он может вертеть и дёргать, не зная, что происходит внутри..



Как видно, тег этот парный. Но его содержимое предназначено только для браузеров, которые этот элемент не поддерживают:

```
<canvas id="my-first-canvas" width="360"
      height="240">
  <p>No canvas support? Have an old-
    fashioned image instead:</p>
  
</canvas>
```

К примеру, в браузере Microsoft Internet Explorer вместо «холста» мы увидим фото щенка, как показано на иллюстрации, поскольку этот браузер не поддерживает `<canvas>`.

Элемент `<canvas>` является обыкновенным inline-block элементом и может быть оформлен с помощью стилей CSS.

Основы рисования

Как уже упоминалось, спецификация HTML5 в полной мере содержит все JavaScript методы и свойства, необходимые для использования.

Рассмотрим буквально несколько из них для того, чтобы проиллюстрировать принцип работы с `<canvas>`.

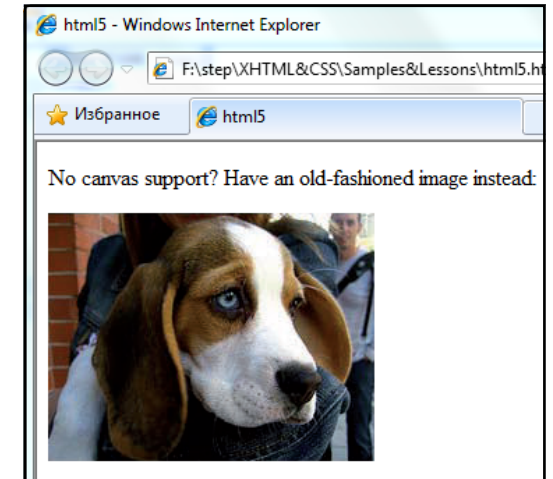
В настоящее время в Mozilla Labs ведется разработка проекта Canvas 3D, который позволит реализовывать 3D сцены в веб-страницах. В случае его готовности, появится второй контекст рисования — '3d'. На данный момент реализован только один — '2d'. Перед началом работы его нужно указать явным образом:

```
var canvas = document.getElementById('my-
  first-canvas');
var context = canvas.getContext('2d');
```

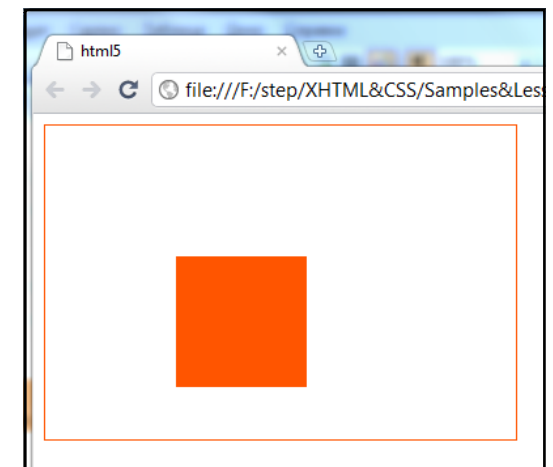
Рисуем закрашенный прямоугольник:

```
context.fillStyle = "#f50";
context.fillRect
  (100, 100, 100, 100);
```

В отличие от SVG, `<canvas>` в чистом виде поддерживает только один примитив — прямоугольник (rectangle). Все дру-



Просмотр примера кода с использованием тега `<canvas>` в Microsoft Internet Explorer.



Прямоугольник, созданный с помощью тега `<canvas>`.

гие формы могут быть созданы комбинацией одной или нескольких кривых, для создания которых имеется солидный набор функций.

Рисование кривых (и просто отрезков) опирается на такие управляющие функции:

- `beginPath()` — создаёт новый список линий, который пополняется с помощью функций рисования, изложенных ниже и в конце может быть выведен либо в виде обводки либо в режиме заливки;
- `closePath()` — замыкает фигуру, рисуя прямую линию от текущей точки до начальной;
- `stroke()` — создает фигуру на основе списка в виде обводки;
- `fill()` — создает фигуру на основе списка в виде заливки.

Они не принимают параметров, а управляют списком линий, хранящихся в оперативной памяти браузера. Этот список пополняется при вызове рисующих функций:

- `lineTo(x, y)` — отрезок прямой
- `arc(x, y, radius, startAngle, endAngle, anticlockwise)` — дуга окружности
- `quadraticCurveTo(cp1x, cp1y, x, y)` — квадратичная кривая Безье

- `bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y)` — кривая Безье
- `rect(x, y, width, height)` — сегмент в виде прямоугольника
- `moveTo(x, y)` — перенос курсора в другую точку без следа на холсте.

Вот пример интерактивного рисования:

```
var canvas = document.getElementById('my-first-canvas');
var context = canvas.getContext('2d');
var interval, r0=1, r=r0, x=0, y=0;
canvas.onmousedown=function(event) {
    function rndColor()
    {return Math.round(Math.random()*255);}
    context.fillStyle=
    "rgb("+rndColor()+","+rndColor()+","+rndColor()+")";
    x=event.clientX-this.offsetLeft;
    y=event.clientY-this.offsetTop;
    interval=setInterval(function() {
        context.beginPath();
        context.arc(x, y, r++, 0, Math.PI*2, true);
        context.fill();
    },10);
}
canvas.onmouseup=function(event) {
    clearInterval(interval);
    r=r0;
}
```


В месте щелчка мышью появляется окружность случайного цвета. При удерживании кнопки мыши, радиус окружности растёт.

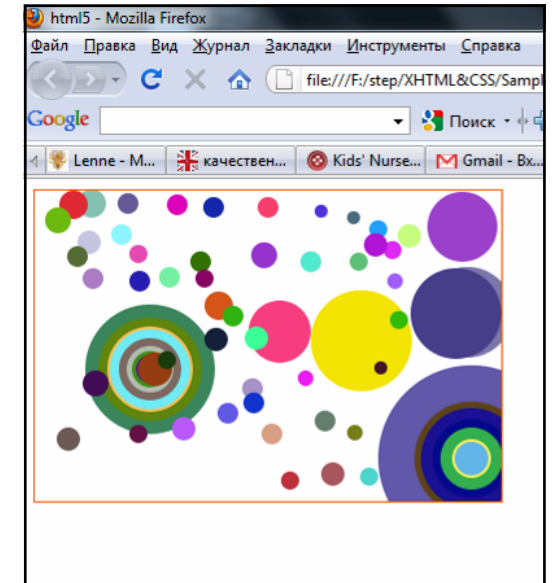
Вы можете опробовать приведённый код в своём браузере, открыв документ *canvas.html*, прилагающийся к этому документу.

Другие возможности

Замечательная возможность тега `<canvas>` — это возможность импорта изображения. Её можно применять для создания динамических фотокомпозиций, фонов графиков и т.п. Внешние изображения могут быть в форматах PNG, GIF или JPEG. Другие элементы `<canvas>` той же страницы HTML также могут быть использованы как источники изображений.

Изображения можно рисовать с нуля, можно изменять их размер, можно разрезать их на части. Причем все это делается при помощи одного и того же метода `drawImage()`. При рисовании можно управлять цветом и прозрачностью при помощи свойств `globalAlpha`, `strokeStyle`, `fillStyle`.

Естественно, поддерживает-ся обработка всех событий, обработка интервалов и задержек, что, в принципе, позволяет в ряде случаев обходиться без использования технологии Adobe Flash на страницах.



Рисование окружностей с помощью сценария JavaScript и тега `<canvas>`.

Вот некоторые примеры того, на что способен язык HTML5 и тэг `<canvas>` в частности:

<http://www.benjoffe.com/code/demos/canvascape/> — 3D-«бродилка», созданная с помощью языка HTML5 и сценариев JavaScript.

<http://www.blobsallad.se/> — несложный пример анимации графических объектов и имитации примитивной «физики».

<http://arapehlivanian.com/wp-content/uploads/2007/02/canvas.html> — анимированный «полет сквозь космос» без использования технологии Adobe Flash.

<http://igrapher.com/> — полноценное и достаточно сложное интерактивное приложение, созданное без применения Flash.

Аудио

Веб-дизайнеры давно мечтали о возможности воспроизведения звуковых файлов на страницах сайтов. Такую возможность давал Adobe Flash, который предоставлял swf-файл с аудио-плеером, внедряемом в страницу. Именно через него и можно про-слушать аудио-файл.

HTML5 предлагает встроить аудио-плеер прямо в браузер и управлять им с помощью тега `<audio>` и его атрибутов.

Вставка аудио-файла на страницу в HTML5 выглядит просто:

```
<audio src="bear.mp3"></audio>
```

Правда именно в таком виде толку не много, поскольку нет плеера и включить звук не получится. Можно, например, добавить атрибут `autoplay` и звук начнет проигрываться сразу после загрузки, но это, конечно, никак не согласуется с соображениями юзабилити. Гораздо лучше применить атрибут `controls`,

который выведет вполне функциональный аудио-плеер. Атрибут `loop` сделает воспроизведение бесконечным.

У элемента `<audio>` имеется небольшой API, представленный методами `play()`, `pause()`, свойством `volume` и др. С помощью JavaScript можно создать свой собственный дизайн аудио-плеера.

Используя этот код, мы должны иметь в виду то, что формат mp3 — проприетарный. За возможность с ним работать требуется платить определенную сумму обладателям патента. Для больших корпораций вроде Apple или Adobe это явно не проблема, в отличие от более мелких компаний и open-source групп. Оттого mp3 прекрасно работает в Safari или Google Chrome, но не проигрывается в Firefox.

Существуют, конечно, и другие форматы. Например, кодек Vorbis (<http://www.vorbis.com/>) — обычно дающий на выходе файлы формата OGG — не ограничивается патентами и лицензиями. Вот он как раз работает в Firefox (но, в свою очередь, по каким-то при-



Аудиоплеер
в Mozilla Firefox 3.10 (вверху)
и в Google Chrome (внизу)



Placeholder вместо аудио-плеера в Mozilla Firefox при попытке воспроизведения mp3 файла

чинам не поддерживается браузером Safari в данный момент).

Для этой проблемы существует решение, которое заключается в применении вместо параметра `src` в теге `<audio>` нескольких заключенных в него элементов `source` для разных файлов:

```
<audio controls>
    <source src="bear.ogg">
    <source src="bear.mp3">
</audio>
```

Браузер попытается загрузить первый файл и, если получится, на этом и остановится. Браузер, который не понимает Ogg Vorbis, пропустит первый файл и загрузит второй. Если сможет.

Видео

Видео в WWW в последние годы приобрело серьёзную популярность. Для его воспроизведения, также, как и для аудио, используется, в основном, Adobe Flash.

В HTML5 вводится тег `<video>`, который очень похож на рассмотренный выше `<audio>`. У него определены те же атрибуты: `autoplay`, `controls`, `loop`, `src`. Также можно создавать форматные вариации при помощи нескольких элементов `<source>`. Аналогично, присутствует возможность создания собственного пользовательского интерфейса.

Главное отличие видео от аудио — размеры. Поэтому для видео, в атрибутах `width` и `height` можно задать размеры ограничивающего прямоугольника:

```
<video src="movie.mp4" controls
      width="360" height="240"></video>
```

Нужно сказать и о недостатках. Главная проблема — в борьбе форматов, она еще более жесткая, чем среди аудиоформатов. Свободным форматом также является OGG, но на сей раз Ogg Theora Video. Ему противостоят форматы MPEG4, DivX и другие. Такие проблемы решаются аналогично аудио:

```
<video controls width="360" height="240">
  <source src="movie.ogv"
    type="video/ogg">
  <source src="movie.mp4"
    type="video/mp4">
</video>
```

Главной проблемой в использовании браузерных мультимедийных плагинов (дополнений) — Flash Player, Silverlight, QuickTime и других — всегда была их изолированность от остальной части документа. Теперь же, когда все эти элементы — часть общей системы, они легко доступны для скриптов и стилей.



Отображение видеоролика в браузере Mozilla Firefox.

Формы 2.0

Одним из мотивов образования WHATWG была работа на спецификацией WebForms 2.0. Сегодня она является неотъемлемой частью HTML5.

Практически сразу после появления в браузерах, JavaScript стал использоваться для улучшения форм и для создания эффектов при работе с мышью.

Направление, касающееся мыши, на данный момент, наверное, является юрисдикцией CSS. С тех пор, как в CSS2.1 псевдокласс `:hover` стало возможно использовать не только для ссылок, но и для других объектов, отпала необходимость в JavaScript для организации несложных анимационных эффектов. CSS3 развивает эту тенденцию. А вот HTML5 занялся совершенствованием форм.

Новые атрибуты

Вот несколько атрибутов, вводимых в HTML5 для формы и её элементов:

Свойство `placeholder` — текст-заглушка. Это то, что всегда делалось с помощью JavaScript.

```
<label for="search">Search</label>
  <input id="search" name="search"
    type="text" placeholder="Поиск...">
```

Теперь текст «Поиск...» будет отображаться в поле, пока в него не будет установлен курсор.

`autofocus` — поле формы с этим атрибутом принимает фокус при загрузке документа. Этот атрибут можно применять к любому полю формы, но только единожды на странице.

`required` — поле, обязательное для заполнения. Предполагается, что браузеры не позволят отправку формы на сервер без заполнения таких полей.

`autocomplete`. Имеет два значения — `on` (по умолчанию) и `off`. В последнем случае, поле (или форма целиком) не позволит браузеру применить автозаполнение, т.е. запоминание

введенных ранее в эти поля значений, что браузеры часто используют, чтобы облегчить работу с формами.

Стоит упомянуть о новом элементе для форм — `<datalist>`. С его помощью можно обычное текстовое поле превратить в `combobox`. Добавив к полю параметр `list`, можно затем создать список предустановленных вариантов выбора:

```
<label for="homeworld">Ваша родная
    планета</label>
<input type="text" name="homeworld"
    id="homeworld" list="planets">
<datalist id="planets">
    <option value="Меркурий">
    <option value="Венера">
    <option value="Земля">
    <option value="Марс">
</datalist>
```

Это позволяет пользователям выбрать вариант из списка или добавить свой, если его там нет. Очень удобно для ситуаций, в которых обычно требуется вставлять дополнительное поле типа «Другое, укажите ниже».

Новые значения атрибута `type` для тега `<input>`

Перемены и дополнения в этой сфере, в принципе, назрели.

Итак новые значения:

`search` — поисковое поле. Браузеры должны будут отображать это поле так как у них принято отображать поля ввода встроенных поисковых систем.


`email`, `url`, `tel` — Опять же, вести себя они будут аналогично обычным полям, с той разницей, что теперь браузеры располагают немного большей информацией относительно данных, которые туда надо вводить, что может быть полезным, к примеру, чтобы проверить правильность ввода.


`range` — ползунок (!). Т.е. тот элемент, который раньше можно было организовать только при помощи JavaScript. Диапазон по-умолчанию — 0..100, но с помощью атрибутов `min` и `max`, его можно настроить.

`amount` — числовое поле со стрелочками для увеличения или уменьшения значений. Полезно для ввода точных значений


Ползунок типа `range` в Opera, Google Chrome и Safari (сверху вниз).

Сколько хотите? 

Сколько хотите? 

Сколько хотите? 

Поле типа `amount` в Opera и Google Chrome (сверху вниз).

Сколько хотите? 

Сколько хотите? 



без клавиатуры. Граничные значения этого поля также можно настроить атрибутами `min` и `max`.

`date` — поле для ввода года, месяца и числа.

`datetime` — поле для ввода даты в полном формате: год, месяц, число, плюс часы, минуты, секунды и указание часового пояса.

`datetime-local` — то же самое, но без указания часового пояса.

`time` — поле для ввода времени: часы, минуты, секунды.

`month` — поле для ввода года и месяца (без числа).

`color` — пипетка для выбора шестнадцатеричного цвета а-ля Photoshop. Пока не поддерживается ни одним браузером.

Несмотря на все это великолепие, говорить о готовности технологии еще рановато. Браузеры поддерживают новые формы достаточно слабо, но готовиться к их использованию нужно уже сейчас.

Start date 2010-10-21 ▾

◀	Октябрь	▶	2010	▲	▼			
Недел	Пн	Вт	Ср	Чт	Пт	Сб	Вс	
39		27	28	29	30	1	2	3
40	4	5	6	7	8	9	10	
41	11	12	13	14	15	16	17	
42	18	19	20	21	22	23	24	
43	25	26	27	28	29	30	31	
44	1	2	3	4	5	6	7	
Сегодня		Отсутствует						

Поле типа `date` в Opera

Другие возможности

В качестве послесловия, можно перечислить то, что еще обещает нам HTML5:

- drag'n'drop — набор событий и их API, при помощи которых можно реализовать это популярное и полезное действие. Да, сейчас для этого есть отличный плагин jQuery UI, но поддержка на уровне браузера — это однозначно лучше.
- storage — возможность хранения браузером информации между сессиями и в автономном режиме. Похоже на cookies, но имеют другой механизм и другой API
- работа с браузерной базой данных WebSQL.
- Работа с сетью. Открытие соединений, управление header'ами и т.д.

- Дополнения в DOM API. В частности, долгожданная функция `getElementsByClassName()`.
- geolocation, т.е. выяснение местоположения пользователя и обмен этой информацией.
- Микроформаты и новые значения атрибута `rel` для ссылок.

Таким образом, видно, что спецификация HTML5 становится более профессиональной и прослеживается еще более тесная связь веб-дизайна с технологиями программирования.

Протестировать свой браузер на поддержку им технологий HTML5 можно онлайн на сайте <http://www.html5test.com/>



Дополнительные материалы

<http://slides.html5rocks.com>

слайды-презентация HTML5 от Google

<http://www.whatwg.org/specs/web-apps/current-work/multipage/>

текущий черновик спецификации

<http://w3pro.ru/article/otlichiya-html-5-ot-html-4>

отличия между HTML5 и HTML4 (перевод официальной статьи W3C)

<http://www.html5test.com/>

онлайн тест браузеров на совместимость с HTML5

<http://www.html5rocks.com/>

сайт поддержки развития HTML5 от Google

https://developer.mozilla.org/ru/%D0%9E%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5_canvas

проект Mozilla Foundation по поддержке проекта Canvas

<http://www.w3.org/html/wg/>

страница рабочей группы W3C по разработке HTML5

http://vremenno.net/html-css/html_5-review/

обзор спецификации

<http://habrahabr.ru/blogs/webstandards/103256/>

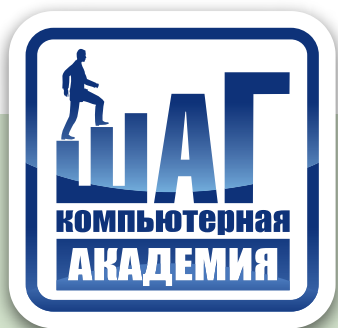
<http://habrahabr.ru/blogs/webstandards/104458/>

<http://habrahabr.ru/blogs/webstandards/104502/>

<http://habrahabr.ru/blogs/webstandards/103575/>

перевод книги «HTML5 для дизайнеров». Дж. Кейт.





А. Пилипенко

СОЗДАНИЕ СТАТИЧНЫХ ВЕБ-СТРАНИЦ
С ПОМОЩЬЮ ЯЗЫКОВ XHTML И CSS.

HTML5. ЧТО НОВОГО?

Верстка: Е. Украинская