

# Контуры HTML 5: Больше тегов, хороших и разных

## Обзор предварительной спецификации нового стандарта языка гипертекстовой разметки

**Кирилл Сухов**

При современных темпах развития веб-технологий, основа основ, стандарт HTML (Hypertext Markup Language), оставался почти неизменен с 1997 года. Появление HTML 4.1 отразило в 1999 году лишь незначительные изменения спецификации. С тех времён, если не считать не совсем успешное продвижение стандарта XHTML, о сколь-нибудь значимых событиях в области разработки базового языка разметки слышно не было. Появился повод огорчить тех, кого данное положение устраивало, – грядёт HTML 5.



7 марта 2007 года W3C официально сообщил о возобновлении разработки спецификации HTML. В ноябре 2007 года консорциум опубликовал первую, предварительную (draft) спецификацию языка гипертекстовой разметки HTML 5. Последний документ консорциума, касающийся HTML, был датирован 24 декабря 1999 года (HTML 4.1).

За прошедшие восемь лет произошло множество изменений в работе и восприятии WWW. Разработчики давно осознали все недостатки HTML 4, и давно назрела необходимость перемен. Новая спецификация вполне претендует на попытку революции в стандарте. Так ли это? Давайте попробуем разобраться.

Само известие о разработке нового стандарта языка гипертекстовой разметки, HTML 5, было встречено в среде веб-разработчиков с немалым удивлением. HTML 4 утвердился давно, и все были уверены – больше версий не будет. Не то чтобы четвертая версия языка была идеальной, просто развитие WWW шло несколькими иным путём.

Спецификация XHTML, разработанная в 2001 году, казалось решила проблему языка гипертекстовой разметки для WWW, дав возможность разработчикам заняться более интересными задачами. Чем же были вызваны перемены?

## Недостатки HTML 4

Какими недостатками может обладать язык разметки, на котором написано подавляющее большинство веб-страниц? На самом деле их много. С моей точки зрения, основной из них – необязательность стандарта.

На каждой странице должны присутствовать теги <html>, <head>, <body>, но... всё прекрасно отображается и без них.

Теги <p> вроде должны иметь закрывающий тег (</p>), но, в общем, можно и без него.

Атрибуты должны быть заключены в кавычки, но если не охота...

Может, вам кажется, что все эти вольности создают большую свободу для разработчика? Это в корне неверно. Точный термин для этой свободы – неопределённое поведение, создающее только проблемы. А со значени-

ями атрибутов по умолчанию и вовсе беда.

К примеру, изображение (тег <img>), у которого не указан атрибут border, отображается с рамкой нулевого размера. Если его же заключить в гиперссылку, толщина рамки станет равна единице.

Впрочем, это только проявление основного недостатка – отсутствия единой концепции языка. Можно сколько угодно пинать производителей браузеров за отступление от стандартов, но по большому счёту они не совсем виноваты – проблема существует в самом стандарте.

## XHTML

Появление XHTML 1.0 несколько изменило ситуацию, но отнюдь не радикальным образом. Кроме того, XHTML не очень прижился. Уже прошло немало времени с его утверждения, но большинство сайтов всё ещё далеко до этого стандарта. Более того, большая часть контента, написанного на XHTML, выдаётся и обрабатывается, как <text/html>, то есть как набор обычных тегов, а не как XML-документ.

Причины – это явления на поверхности. Сам стандарт XML предполагает очень строгий механизм обработки ошибок. Разбор XML-документа прекращается при первой же встреченной ошибке, что означает недоступность всего документа с единственной синтаксической погрешностью. Естественно, такой подход не приемлем для производителей браузеров и разработчиков, заинтересованных в том, чтобы содержимое веб-страницы, пусть в искажённом виде, но было донесено до пользователя.

Другая, широко известная проблема, тесно связанная с первой, заключается в том, что «Самый Распространённый Браузер» (около 70% пользователей на 25 февраля 2008 года по статистике моего собственного веб-сервера) вообще не поддерживает XHTML в виде «application/xhtml+xml», то есть не применяет какие-либо правила XHTML, не имеет понятия об их синематике. Конечно, можно с важным видом говорить, что это проблемы Microsoft Internet Explorer, а не XHTML, но, увы, башен из слоновой кости на всех не хватает, а вне оной – это проблемы разработчика.

Последний аргумент в крышку стандарта XHTML 1.0 – поисковые системы не индексируют XHTML в виде XML-документов. Для XML создано очень мало инструментов для CMS и тому подобных систем. Как следствие – низкий интерес у рекламодателей и производителей контента.

## XHTML 2.0

Вторая версия стандарта XHTML вроде бы была призвана решить вышеупомянутые проблемы, но на самом деле этот проект, разрабатываемый довольно давно, не нашёл поддержки ни у одного из ведущих производителей браузеров, да и вообще не вызывает особого восторга у рядовых разработчиков. И это вполне понятно – создавали стандарта, по-видимому, в целях идеологической чистоты не интересовались мнением его будущих пользователей. Если XHTML 1.0 привели к XML и HTML, то XHTML 2.0 – это новый язык, пока никак не присутствующий во Всемирной сети. Он добавляет дополнительный уровень сложности, в нём изменена синематика для многих элементов.

W3C уже довольно давно работает над XHTML 2.0, но последний пока существует только в качестве рабочего проекта. Ни один из крупных разработчиков браузеров его не поддерживает, и есть подозрение, что реализация данного стандарта не выйдет из академической среды.

## Web Applications

Спецификация Web Applications 1.0 разрабатывалась рабочей группой WHATWG (Web Hypertext Application Technology Working Group), куда вошли трое из четырёх ведущих поставщиков браузеров, а именно Mozilla Foundation, Opera Software и Apple. По словам одного из непосредственных участников разработки, эти компании противостояли инициативе W3C полностью перейти на XML и оставить HTML в прошлом.

Изначально разработка представляла собой набор поэтапных спецификаций, добавляющих в HTML самые необходимые (с точки зрения этой рабочей группы) функции. Кроме того, WHATWG, не затрагивая XHTML 2.0, предлагает набор обновлений XHTML 1.0 под именем XHTML 5

и обновление HTML DOM (Document Object Model) как DOM5 HTML.

Еще не запутались? Чтобы добить окончательно, добавлю, что WA 1.0 определяет несколько новых программных интерфейсов, улучшающих использования WWW как платформу для приложений.

Добавлены такие важные вещи, как локальное хранение информации, её автономный просмотр и работа с историей (предполагается возможность обработки кнопки back!).

Кроме того, добавлены функции, приближающие потенциальный интерфейс веб-браузера к стандартному для любого оконного приложения виду. Это функции undo/redo, copy/paste, drag and drop и другие.

Некоторые функции WHATWG уже реализованы в браузере Safari для Apple и в Mozilla Firefox 1.5 (в частности, элемент для работы с 2D-графикой – canvas).

В общем, можно сказать, что WA – это основа будущей спецификации HTML 5, в рабочей группе W3C, созданной отдельно от разработки XHTML 2.0, работают и представители четвертого крупного производителя браузеров. Крис Уилсон, архитектор Microsoft Internet Explorer является её сопредседателем.

Впрочем, у HTML 5 есть ещё одна немаленькая составная часть.

## Web Forms 2

Спецификация Web Forms 2.0, также разработанная группой Web Hypertext Application Technology Working Group, пытается решить многие из проблем, которые разработчики находят в текущем состоянии HTML-форм. Сегод-

нящие формы не используют многие основные функции обычных приложений для настольных компьютеров, таких как проверка корректности вводимых данных, соответствие их ожидаемому типу. Не используют они и функциональные виджеты.

Сама спецификация призвана решить подобные проблемы. Изначально она вырабатывалась как для HTML, так и для XHTML, но в конце концов её положения нашли отражение в предварительной версии HTML 5. Следует уточнить, что с частью стандарта XHTML 2.0, XForm, Web Forms 2.0 не имеет ничего общего.

## Отличия от HTML 4

Основная задача HTML 5 обеспечить однозначное определение процесса разбора HTML-страницы любой программой (в частности, браузером). Причём если ранее производители браузеров сами «вытягивали» отображение неправильно сформированных и невалидных документов, то теперь стандарт должен указать на этот счет четкие и однозначные правила поведения. Что было сделано на этом пути? С тегами, однозначно указывающими свою обработку, поступили радикально – их удалили из спецификации.

По сравнению с HTML 4 стандарт HTML 5 лишается около 15 тегов.

Самой заметной потерей здесь станут <font>, использование его, а также тегов basfont, big, center, s, strike, tt и u предполагается заменить применением каскадных таблиц стилей.

По давно назревшим идеологическим соображениям за бортом спецификации остались теги frame, frameset,

noframes – разговор о вредоносности использования фреймов не стихал, с самого, наверное, их появления. Заметим – тег iframe оставлен.

Следующая группа элементов не вошла в новый стандарт ввиду их редкого использования, неоднозначности трактовки, либо дублированием их функций другими элементами: acronym, applet (теперь применяем только object), isindex, dir.

Чистка рядов коснулась и атрибутов. Далее перечисляются все, не вошедшие в спецификацию:

- Атрибут align элементов caption, iframe, img, input, object, legend, table, hr, div, h1, h2, h3, h4, h5, h6, p, col, colgroup, tbody, td, tfoot, th, thead, tr и body.
- Атрибуты alink, link, text и vlink элемента body.
- Атрибуты background и bgcolor элемента body. Атрибут bgcolor упраздняется также и для элементов table, tr, td и th.
- Атрибут border для элементов table, img и object.
- Атрибуты cellpadding и cellspacing для элемента table.
- Атрибуты char и charoff для col, colgroup, tbody, td, tfoot, th, thead и tr.
- Атрибут clear элемента br.
- Атрибут compact элементов dl, menu, ol и ul.
- Атрибут frame для элементов table.
- Атрибут height для элементов iframe, td и th.
- Атрибуты hspace и vspace для элементов img и object.
- Атрибуты marginheight и marginwidth для iframe. Для этого элемента отменены также атрибуты frameborder и scrolling.
- Атрибут noshade для hr.
- Атрибут nowrap для td и th.
- Атрибут rules для table.
- Атрибут size для элементов hr, input и select.
- Атрибут style для всех элементов(!).
- Атрибут type для элементов li, ol и ul.
- Атрибут valign для элементов tol, colgroup, tbody, td, tfoot, th, thead и tr.
- Атрибут width для элементов hr, table, td, th, col, colgroup, iframe и pre.

Атрибуты элементов медиаконтента и их значения

Fnhb,en	audio	video	
src	+	+	Источник медиаконтента
autoplay	+	+	Флаг автоматического запуска медиаресурса
start	+	+	Стартовая позиция медиаресурса
loopstart	+	+	Начальная позиция циклического воспроизведения медиаресурса
loopeend	+	+	Конечная позиция циклического воспроизведения медиаресурса
end	+	+	Финишная позиция медиаресурса
playcount	+	+	Задаёт очерёдность проигрывания медиаресурсов
controls	+	+	Флаг, включающий интерфейс для пользователя
poster	+	+	URL или URI файла-заставки
width	+	+	Ширина демонстрируемого видеофрагмента
height	+	+	Высота демонстрируемого видеофрагмента

Уф, вроде всё. Возможно ли после такой чистки называть стандарт обратно совместимым? Да, собственно, почему бы и нет.

На мой взгляд, в веб-строительство привнесено немного порядка, только и всего. Тем более что главное требование обратной совместимости будет выполнено – приложения, рассчитанные на работу с документами HTML 5, могут без изменений работать с документами HTML 4. Впрочем, выкидыванием элементов дело, понятно, не ограничилось.

Посему идём дальше.

## Новые элементы

Какие новые элементы нам предлагает HTML 5? Если коротко – разные.

Основой спецификации является выделение на странице логических блоков, без указания конкретики отображения (типичный пример такой конкретики – упразднённый в ней тег `<centre>` и аналогичные ему).

Ниже представлен фрагмент страницы, сверстанной с использованием новых элементов.

```
<body>
<header>...</header>
<nav>...</nav>
<article id="1">
<section>

...

</section>
</article>
<article id="2">...</article>
<aside>...</aside>
<footer>...</footer>
</body>
```

Тут, я думаю, назначение любого элемента вполне понятно из его названия:

- **<header>** – отмечает заглавную часть секции (не только и не обязательно заголовок и не обязательно страницы);
- **<footer>** – представляет нижнюю часть секции (например, с информацией об авторстве и копирайтом);
- **<article>** – представляет независимый блок контента (вроде газетной статьи);
- **<section>** – секция документа или приложения. Совместно с тегами заголовков (`<h1>`–`<h6>`) может организовывать структуру документа:

```
<section>
  <h1>Первый уровень</h1>
  <section>
    <h1>Второй уровень</h1>
    <section>
      <h1>Третий уровень</h1>
    </section>
  </section>
</section>
```

- **<nav>** – представляет блок навигации документа;
- **<aside>** – боковой сайдбар, например, классическое боковое (левое или правое) меню.

Что нам обещают ещё?

Для выделения участка текста взамен упразднённых тегов предлагается один, универсальный – `<m>`:

```
<p>The highlighted part below is where the error lies:</p>
<pre>
<code>var i: Integer;
begin
  i := <m>1.1</m>;
end.</code>
</pre>
<p>I also have some <m>kitten</m>s who are visiting me
these days. They're really cute. I think they like my
garden!</p>
```

Следующий тег обещает быть крайне полезным... Ну, наверное...

```
<dialog>
<dt> Pyatachok
<dd> Вини, а куда это мы идём?
<dt> Vinny Puh
<dd> За мёдом!
<dt> Pyatachok
<dd> А где мы его возьмём?
<dt> Vinny Puh
<dd> У пчёл.
</dialog>
```

А вот по-настоящему полезные теги: `<audio>` и `<video>`. Предназначены для отображения мультимедиа-контента. Они предоставляют API для различных медиаприложений и имеют следующие атрибуты (см. **таблицу**).

И простенькие примеры использования:

```
<audio src="starik kozlodoev.mp3"
autoplay="autoplay" loop="20000" />

<audio src="hymn.mp3">
<p>Президент Российской Федерации в своей речи...
</p>
</audio>
```

Тег `<figure>` служит для ассоциации разнородного содержимого с заголовком, который в свою очередь представляет тег `<legend>`:

```
<figure>
<video src=...></video>
<legend>caption</legend>
</figure>
```

Ещё один потенциально полезный тег – `<embed>` используется для подключения контента различного типа (подразумевается, что тип будет не html). Он имеет только четыре атрибута, кроме размеров это тип и источник контента.

Элемент `<m>` (от marked) выделяет текст как «помеченный». В полном соответствии с новой идеологией он жестко не предписывает его визуальное выделение (например, подчёркивание), всё зависит от выбранного стиля отображения и приложения.

В качестве примера спецификация приводит помеченные Google закешированные страницы в результатах поиска:

```
The Great <m>Egret</m> (also known as the
American <m>Egret</m>) is a large white wading bird found worldwide.
The Great <m>Egret</m> flies with slow wing beats.
The scientific name of the Great <m>Egret</m> is <i>Casmerodius albus</i>
```

Тег `<canvas>`, пожалуй, одно из самых интересных новшеств стандарта. Он предоставляет API для работы с двумерными графическими объектами, или, проще говоря, поз-

воляет создавать двумерные рисунки средствами JavaScript. Рекомендую статью по его использованию, опубликованную mozilla developer center: [http://developer.mozilla.org/en/docs/Canvas\\_tutorial:Basic\\_usage](http://developer.mozilla.org/en/docs/Canvas_tutorial:Basic_usage).

Пример работы с этим тегом:

```
<html>
<head>
<script type="application/x-javascript">
function draw() {
var canvas = document.getElementById("canvas");
if (canvas.getContext) {
var ctx = canvas.getContext("2d");

ctx.fillStyle = "rgb(200,0,0)";
ctx.fillRect (10, 10, 55, 50);

ctx.fillStyle = "rgba(0, 0, 200, 0.5)";
ctx.fillRect (30, 30, 55, 50);
}
}
</script>
</head>
<body onload="draw();" >
<canvas id="canvas" width="150" height="150"></canvas>
</body>
</html>
```

Результат показан на **рисунке**.

Элемент `<meter>` представляет собой аналог прогресс-бара (например, индикатора загрузки). Естественно, его параметры могут быть заданы динамически. Вот примеры использования этого элемента:

```
<meter>75%</meter>
<meter>750%</meter>
<meter>3/4</meter>
<meter>6 blocks used (out of 8 total)</meter>
<meter>max: 100; current: 75</meter>
<meter><object data="graph75.png">0.75</object></meter>
<meter min="0" max="100" value="75"></meter>
```

Элемент `<time>` позволяет включать дату/время, причём один из его атрибутов (timezone) отвечает за локализацию этих параметров:

```
<p>We stopped talking at ⌵
<time datetime="2006-09-24 05:00 -7">5am ⌵
the next morning</time>.</p>
```

Тег `<command>` открывает серию из нескольких элементов, вносящих на HTML интерактивность. Вот пример его использования:

```
<menu>
<command onclick="alert('first command') " ⌵
label="Do 1st Command"/>
<command onclick="alert('second command') " ⌵
label="Do 2nd Command"/>
<command onclick="alert('third command') " ⌵
label="Do 3rd Command"/>
</menu>
<menu type="popup" label="Edit">
<command onclick="undo() " label="Undo"/>
<command onclick="redo() " label="Redo"/>
<command onclick="cut() " label="Cut"/>
<command onclick="copy() " label="Copy"/>
<command onclick="paste() " label="Paste"/>
<command onclick="delete() " label="Clear"/>
</menu>
```

Используемый здесь тег `<menu>`, известен ещё с HTML 2, а в HTML 4 он был помечен как deprecated (устаревший) и посему не рекомендованный для использования). В дан-

ном случае он служит контейнером для нескольких элементов `<command>`, каждый из которых предоставляет пользователю интерфейс для выполнения какого-либо сценария.

Элемент `<progress>` представляет элемент управления, сходный с GNU прогресс-баром (например, индикатором загрузки или копирования). Приведу пример его использования:

```
<p>Downloaded:
<progress value="333" max="1000">33%</progress>
/p>
```

Элемент `<datagrid>` – представляет иерархически организованную информацию в виде дерева или вкладок-табов.

Для организации комбобокса в HTML-формах теперь предусмотрен новый тег – `<datalist>`. Ниже пример его работы вместе с новым атрибутом тега `<input>` – `list`:

```
<input list="browsers">
<datalist id="browsers">
<option value="Opera">
<option value="Firefox">
<option value="Internet Explorer">
</datalist>
```

Для `<input>` вообще сделано много интересного. Например добавлены следующие возможные значения атрибута `type`:

- `datetime`
- `datetime-local`
- `date`
- `month`
- `week`
- `time`
- `range`
- `email`
- `url`

Смысл этих нововведений не только в контроле данных формы перед отправкой на сервер со стороны клиента. Предполагается, что браузер будет предоставлять свои ресурсы вроде календаря или адресной книги для интеграции с веб-приложением посредством полей форм. Впрочем про атрибуты далее.

## Старые теги о главном (новые атрибуты)

Тег `<input>` не единственный, известный с HTML 4 и ранее элемент, который обзавёлся новыми атрибутами. Попробуем в них разобраться.

Прежде всего тот же `<input>` вместе с `<textarea>` приобрели атрибут `required`, диктующий обязательность заполнения полей формы (разумеется, у таких `input`-элементов, как `image`, `hidden` или `submit/reset` он отсутствует). Эта же пара элементов по новой спецификации теперь имеет атрибут `inputmode`, значение которого призвано подсказать пользователю ожидаемое заполнение полей.

Ещё несколько новых атрибутов для тега `<input>`:

- `Autocomplete`
- `Autofocus`



- Min
- Max
- Pattern
- Step

Как и новые значения type, все новые атрибуты пришли из проекта Web Form 2.0. Об их назначении в большинстве случаев нетрудно догадаться по названию. Ниже приведён образец формы HTML 5:

```
<form>
<label>Date<input name="date" type="datetime" ␣
Autocomplete ␣></label>
<label>Name: <input name="name" required/></label>
<label>E-mail: <input name="email" type="email" ␣
required/></label>
<label>URL: <input name="url" type="url" ␣></label>
<label>Comment: <textarea name="comment" ␣
required/></label>
<label>Search: <input name="search" autofocus/></label>
<input type="submit" value="React!" ␣>
</form>
```

Теги <a> и <area> получили атрибуты media и ping. Первый указывает на то, к каким медиаресурсам происходит обращение (медиаресурс должен относиться к допустимым медиатипам). Значением атрибута ping является список нескольких URI, разделённых пробелами.

Назначение – отображать данную ссылку с возможностью выбора доступного в данный момент ресурса. Честно говоря, не очень представляю, как это будет реализовано, но сама идея довольно заманчива.

Обязательно стоит упомянуть о группе атрибутов, применимой к любому повторяемому элементу, призванной организовать это повторение:

- repeat
- repeat-start
- repeat-min
- repeat-max

А также об атрибутах, каждый из которых реализует отдельный API 9 и достоин подробного разбирательства в отдельной статье):

- **contenteditable** – признак редактируемости содержимого элемента;
- **contextmenu** – возможность задания своего контекстного меню;
- **draggable** – применимость drag & drop API;
- **irrelevant** – признак релевантности элемента.

Соответствуя веяниям времени (как и положено стандарту, изрядно при этом запаздывая), тег <script> обрёл атрибут async, отвечающий за синхронность/асинхронность выполнения сценария. Разумеется, это в первую очередь скажется на более органичном применении AJAX-технологии, но «фоновое» обращение к серверу далеко не единственное применение технологии асинхронного выполнения сценариев.

## Изменённые элементы

Ниже перечислены элементы, подвергнувшиеся в новой спецификации некоторым изменениям. Сразу предупреждаю – не бойтесь, ничего страшного:

- Элемент <a> без атрибута href теперь воспринимается как якорь.

- Возможности элемента <address> теперь расширяемы элементами section.

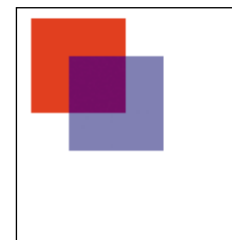
- Элемент <b> теперь обозначает стилистическое выделение, не конкретизируя определённого внешнего вида. То же касается элемента <i>, который передаёт альтернативное начертание. Оба тега перестают означать конкретный вид текста, а становятся элементами логической разметки.

- Элемент <hr> теперь передаёт границу блока контента, уровня параграфа.

- Элемент <menu>, как было показано выше, теперь действительно применяется для организации меню.

- Элемент <small> также становится логическим. Он теперь обозначает мелкий шрифт – например комментария или копирайта.

- Элемент <strong> ... правильно, так же переходит в разряд логической разметки и обозначает важность содержания.



Пример работы тега <canvas>

Я перечислил не все новые теги и атрибуты, с полным списком нововведений можно ознакомиться в официальной документации рабочих групп, ссылки на которую даны в конце статьи.

## Когда?

Думаю, не ошибусь, что, глядя на все изменения и нововведения, рядовой разработчик в первую очередь задастся вопросом: когда данные изменения в стандарте станут повседневной реальностью? Когда HTML 5 будет поддерживаться основными браузерами, когда его возможности станут задействованы в веб-приложениях?

Заявлено, что финальная версия стандарта выйдет не ранее конца 2010 года. В FAQ от WHATWG ([http://wiki.whatwg.org/wiki/FAQ#When\\_will\\_HTML\\_5\\_be\\_finished.3F](http://wiki.whatwg.org/wiki/FAQ#When_will_HTML_5_be_finished.3F)) указан другой срок – 2012 год. За это время производители браузеров должны привести свою продукцию под новый стандарт. Учитывая участие их представителей в рабочей группе, есть надежда, что этот срок и станет началом новой жизни WWW. Правда, остаётся ещё сообщество разработчиков, не всегда на ура принимающее какие-либо изменения, но если крупные игроки форсируют внедрение новых стандартов, куда им (вернее, нам) деваться? В общем, дабы не оказаться на обочине, предлагаю всем причастным к веб-разработке держать нос по ветру. Перемены неизбежны.

Удачи! 🍀

1. Web Forms 2.0. Working Draft. 12 October 2006 – <http://www.whatwg.org/specs/web-forms/current-work>.
2. HTML 5. A vocabulary and associated APIs for HTML and XHTML. W3C Working Draft 4 March 2008 – <http://www.w3.org/html/wg/html5>.
3. HTML 5 differences from HTML 4. W3C Working Draft 22 January 2008 – <http://www.w3.org/TR/2008/WD-html5-diff-20080122>.