HTML

Table of Contents

История (In English)

HTML (HyperText MarkupLanguage) – стандартный язык разметки документов во Всемирной паутине. Большинство веб-страниц содержат описание разметки на языке HTML (или XHTML). Язык HTML интерпретируется браузерами и отображается в виде документа в удобной для человека форме.

Как все развивалось:

- 1993 – Tim Berners-Lee proposes a draft of HTML to the Internet Engineering Task Force (IETF), a standards organization
- 1994 – original HTML draft expires, the IETF creates the first HTML Working Group (HTMLWG), which later creates HTML 2
- 1994 – Tim creates the World Wide Web Consortium (W3C), with a mission To lead the World Wide Web to its full potential by developing protocols and guidelines that ensure long-term growth for the Web
- 1996 – after a series of additions to HTML 2, the IETF HTMLWG closes and further work on HTML moves to the W3C
- 1997 – The W3C publishes HTML 3.2 and HTML 4
- December, 1999 – HTML 4.01 is published
- 1998 - … – the W3C rests on HTML…

And then there was XHTML. Backing up a bit, HTML was always based on SGML, the Standard Generalized Markup Language, an International Standards Organization (ISO) standard. Without getting into the details of SGML, many saw it as too ambiguous for a reliable web. Enter XML. The eXtensible Markup Language, is a stricter subset of SGML, with the goal of clearer communication. It's also, as the name implies, designed for extensibility.

- 2000 – XHTML
- 2002 – XHTML 2

Browser vendors fell all over themselves in their rush to not implement XHTML2, so it never went anywhere. Many attribute XHTML 2's failure to the lack of compatibility with HTML 4 or XHTML 1. Amidst the lack of progress on XHTML 2, there was a workshop in 2004 in which some people decided the W3C was no longer managing the web very well. So they created their own organization, the Web Hypertext Application Technology Working Group (WHATWG).

- 2004 – WHATWG is formed, begins work on a new standard (HTML5)

The WHATWG was founded by individuals of Apple, the Mozilla Foundation, and Opera Software in 2004, after a W3C workshop. Apple, Mozilla and Opera were becoming increasingly concerned about the W3C's direction with XHTML, lack of interest in HTML and apparent disregard for the needs of real-world authors. So, in

response, these organizations set out with a mission to address these concerns and the Web Hypertext Application Technology Working Group was born.

- 2006 — W3C says HTML should be reinvented

That was a significant (complete) change from the W3C's previous position, essentially that HTML was dead, to be non-incrementally replaced by XHTML 2. If the new HTMLWG sounded a lot like the WHATWG, that's because it was.
But unlike the previous transition from the IETF to the W3C, the WHATWG didn't simply hand control of HTML back to the W3C. Even after the WHATWG's HTML 5 draft was formally adopted by the HTMLWG in 2007, the WHATWG continued working on it.

- 2008 – 1st public draft of HTML5 is published
- 2009 – XHTML 2 is dead

HTML5 does include XHTML5, an XML syntax of the language. So XHTML in general isn't dead, but it's not exactly healthy. Given the lack of any other path forward, it does seem very likely that somehow HTML5 will become the recommended web publishing format of both the WHATWG and the W3C's HTMLWG. Somehow.

- 2012 – W3C designates HTML5 as a Candidate Recommendation (recommendation is "two 100% complete and fully interoperable implementations")

Plans:

- End of 2014 – a plan to release a stable HTML5 Recommendation
- End of 2016 – a plan to release HTML 5.1 specification Recommendation

One thing that is entirely clear about the future of HTML5 is that browsers will support it. That's clear because they already support it. One aspect of it, canvas, has been supported, and heavily relied upon, for years. If nothing else, HTML5 has already reversed the relationship between browsers and standards bodies. Instead of the W3C all but begging browsers to implement standards, browsers are now impatiently waiting for the W3C to recommend the standards they've already implemented.

Introduction

Простой HTML-документ

```
<!DOCTYPE html>
 <html lang=“en">
 <head>
   <title>A tiny document</title>
```

```
 </head>
 <body>
   <h1>Main heading in my document</h1>
   <!-- Note that it is "h" + "1", not "h" + the letters "one" --
>
   <p>Look Ma, I am coding
   <abbr title="Hyper Text Markup Language”>HTML</abbr>.</p>
 </body>
 </html>
```

HTML документ состоит из дерева элементов и текста. Каждый элемент имеет открывающий тэг, такой как <body>, и закрывающий, такой как </body>.

Определенные открывающие и закрывающие тэги в некоторых ситуациях можно опустить.

Тэги должны находиться полностью один в другом, не «перекрывая» друг друга:

Плохо: `<h1>This is a <a>link</h1></a>`

Хорошо: `<h1>This is a <a>link</a></h1>`

Тэги

1. Ссылка:
   ```
   <a href=http://www.w3schools.com/ target="_blank”>Bang!</a>
   ```

2. Изображение:
   ```
   <img src="smiley.gif" alt="Smiley face" width="42"
   height="42">
   ```

3. Таблица:
   ```
   <table>
      <thead>
        <tr>
          <th>Header content 1</th>
          <th>Header content 2</th>
        </tr>
      </thead>
      <tfoot>
        <tr>
          <td>Footer content 1</td>
          <td>Footer content 2</td>
        </tr>
      </tfoot>
      <tbody>
   ```

```
    <tr>
      <td>Body content 1</td>
      <td>Body content 2</td>
    </tr>
  </tbody>
</table>
```

Результат:

Header content 1	**Header content 2**
Body content 1	Body content 2
Footer content 1	Footer content 2

Таблица с подписью:

```
<table>
   <caption>Awesome caption</caption>
   <tr>
     <td>Awesome data</td>
   </tr>
</table>
```

Результат:

Awesome caption

Awesome data

4. Комментарий:
   `<!-- This is comment text -->`

5. `<span>`— строчный контейнер. Может использоваться для выделения группы элементов с целью стилизации.
   `<span>Some text</span>`

6. `<div>` — блочный контейнер, предназначенный для различных общих целей. Может использоваться для группировки элементов с целью стилизации.
   ```
   <div>
        <p>Any kind of content here. Such as &lt;p&gt;,
        &lt;table&gt;. You name it!</p>
   ```

```
</div>
```

7. Неупорядоченный список:

```
<ul>
   <li>first item</li>
   <li>second item</li>
   <li>third item</li>
</ul>
```

Результат:

- first item
- second item
- third item

8. Упорядоченный список:

```
<ol>
  <li>first item</li>
  <li>second item
    <ol>
      <li>second item first subitem</li>
      <li>second item second subitem</li>
      <li>second item third subitem</li>
    </ol>
  </li>
  <li>third item</li>
</ol>
```

Результат:

1. first item
2. second item
    1. second item first subitem
    2. second item second subitem
    3. second item third subitem
3. third item

9. Заголовки

Заголовки в HTML предназначены только для заголовков. Некорректно использовать тэги заголовков только для того, чтобы увеличить текст или сделать его жирнее. Поисковые роботы используют заголовки на странице для того, чтобы структурировать содержимое страницы.

```
<h1>Heading level 1</h1>
<h2>Heading level 2</h2>
<h3>Heading level 3</h3>
<h4>Heading level 4</h4>
<h5>Heading level 5</h5>
```

```
<h6>Heading level 6</h6>
```

Heading level 1

Heading level 2

Heading level 3

Heading level 4

Heading level 5

Heading level 6

Формы

HTML формы используются для передачи данных на сервер.

```
<form name="input" action="html_form_action.asp" method="get”>
First name: <input type="text" name="firstname"><br>
Last name: <input type="text" name="lastname"><br>

Password: <input type="password" name="pwd"><br>

<input type="radio" name="sex" value="male">Male<br> <input
type="radio" name="sex" value="female">Female<br>

<input type="checkbox" name="vehicle" value="Bike">I have a
bike<br> <input type="checkbox" name="vehicle" value="Car">I have
a car<br>

<input type="submit" value="Submit">
</form>
```

Некоторые возможные значения атрибута type для элемента `<input>`:

10. button
11. checkbox
12. color (HTML5)
13. email (HTML5)
14. hidden
15. password

И другие. Полный список см. https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input#attr-type

Клиентская валидация (HTML5)

HTML5 предоставляет возможность валидации формы на клиенте. Необходимо помнить, что данная функциональность не заменяет валидации на серверной стороне, а лишь улучшает восприятие сайта пользователем.

Возможности, предоставляемые HTML5:

- атрибут `required` для `<input>`, `<select>`, и `<textarea>` элементов означает, что значение не может быть пустым
- атрибут `pattern` для `<input>` элемента означает, что значение должно попадать под регулярное выражение
- атрибуты `min` и `max` для `<input>` элементов задают минимальное и максимальное значении
- атрибут `step` для `<input>` элемента (в комбинации с `min` и `max` атрибутами) указывает на шаг возможного значения
- атрибут `maxlength` для `<input>` и`<textarea>` элементов указывает на максимальное допустимое число символов
- значения `url` и `email` для атрибута `type` означают, что значение поля будет проверяться на валидный URL или e-mail адрес

Атрибуты

Атрибуты расположены внутри открывающего тэга и состоят из имени и значения, разделенных знаком =. Значение атрибута можно не заключать в кавычки, если оно не содержит пробелов, а также " ' ` = < или >. В таком случае, значение атрибута необходимо обернуть в кавычки (двойные либо одинарные). Если значение атрибута — пустая строка, то знак равенства, а также само значение можно опустить.

```
<!-- empty attributes -->
<input name=address disabled>
<input name=address disabled="">

<!-- attributes with a value -->
<input name=address maxlength=200>
<input name=address maxlength='200'>
<input name=address maxlength="200”>
```

Спецсимволы

По-другому — символы-мнемоники. Чтобы вставить определённый символ в разметку, нужно вставить определенную ссылку-мнемонику в HTML структуру.

Наиболее распространенные:
`>` обозначает знак «больше» (>)
`<` обозначает знак «меньше» (<)
`&` обозначает амперсанд (&)

`&quot;` обозначает двойные кавычки (")

Все спецсимволы перечислены здесь: http://www.w3.org/TR/2011/WD-html5-20110113/named-character-references.html.

Структура документа

```
<!DOCTYPE html>
<html>
<head>
  <title>...</title>
  ...
</head>
<body>...</body>
</html>
```

DOCTYPE

Объявление типа документа, сообщает валидатору, какую именно версию HTML вы используете в своей странице. Этот тег должен всегда находиться в первой строке каждой страницы. Тег DOCTYPE - ключевой компонент web-страниц, претендующих на соответствие стандартам: без него ваш код и CSS не пройдут проверку валидатором.

Если вы будете пользоваться неполным тегом DOCTYPE, устаревшим его видом, или вообще забудете про него, браузер перейдет в "загадочный" режим и будет исходить из предположения, что вы писали код страницы с ошибками и вольно отступали от стандартов, т.е. так, как писали в конце 90-ых годов.

В этом режиме браузер попытается разобрать вашу страницу по правилам обратной совместимости и выведет на экран, например, CSS так, как его вывел бы Internet Explorer 4-ой версии, а DOM будет работать так, как он работал именно в этом браузере (IE переключается в свой старый DOM, а Mozilla и Netscape 6 переключается вообще в бог знает что).

Объявление `<!DOCTYPE>` должно быть самым первым, до тэга `<html>`.

`<!DOCTYPE>` не является тэгом, это инструкция.

В HTML 4.01, `<!DOCTYPE>` ссылается на DTD (Document Type Definition), так как HTML 4.01 был создан на базе SGML. HTML5 не основан на SGML и, следовательно, не требует ссылки на DTD.

Версия `DOCTYPE,` которую нужно использовать: `<!DOCTYPE html>`

<html>
Корневой элемент. Подробнее прочитать про него здесь:
https://developer.mozilla.org/en-US/docs/Web/HTML/Element/html

<head>
Может включать в себя название документа, мета-информацию, скрипты, стили и другое.

```
<html>
   <head>
     <title>Document title</title>
   </head>
 </html>
```

Возможные тэги внутри тэга <head>:
<title>, <style>, <meta>, <link>, <script>, <noscript>, <base>.

<title>
Определяет название документа в строке браузера, в закладках, а также в результатах поиска в поисковиках. Обязательный тэг.

```
<title>Awesome page title</title>
```

<base>
Определяет базовый адрес для ресурсов с относительными адресами в документе.

```
<base href="http://www.example.com/">
<base target="_blank" href="http://www.example.com/">

<img src="images/logo.gif">

// http://www.example.com/images/logo.gif
```

<link>
Данный тэг чаще всего используется для подключения стилей.

```
<link href="style.css" rel="stylesheet" type="text/css"
media="all">
```

<meta>
В данном тэге указывается мета-информация. Метаданные не отображаются на странице, однако роботы имеют доступ к ним.

```
<!-- In HTML5 -->
<meta charset="utf-8">
```

```
<meta name="description" content="Start learning HTML">

<meta name="keywords" content="HTML, CSS, XML, XHTML,
JavaScript">

<meta name="author" content="Vasya">
```

<body>

Содержимое документа. Чаще всего это текст либо встроенный контент.

HTML5

- Новая версия языка с новыми элементами, атрибутами, поведением
- Расширенный набор технологий (Storage, IndexedDB, WebSockets, WebWorkers, Drag/Drop, Geolocation …)

  - Коллекция примеров, демонстрирующих применение новейших технологий HTML5: https://developer.mozilla.org/en-US/demos/tag/tech:html5
  - Помните, что не все нововведения HTML5 поддерживаются во всех браузерах. См. http://caniuse.com/#cats=HTML5

Новое в HTML5:

- *Semantics*: allowing you to describe more precisely what your content is.
- *Connectivity*: allowing you to communicate with the server in new and innovative ways.
- *Offline & Storage*: allowing webpages to store data on the client-side locally and operate offline more efficiently.
- *Multimedia*: making video and audio first-class citizens in the Open Web.
- *2D/3D Graphics & Effects*: allowing a much more diverse range of presentation options.
- *Performance & Integration*: providing greater speed optimization and better usage of computer hardware.
- *Device Access*: allowing for the usage of various input and output devices.
- *Styling*: letting authors write more sophisticated themes.

Семантика

Позволяет более точно описывать предназначение содержимого.

<section>

Рассмотрим пример на HTML4:

```
<div class="section" id="forest-elephants" >
   <h1>Forest elephants</h1>
   <p>In this section, we discuss the lesser known forest
elephants.     ...this section continues...
  <div class="subsection" id="forest-habitat" >
     <h2>Habitat</h2>
     <p>Forest elephants do not live in trees but among them.
...this subsection continues...
   </div>
 </div>
```

Имеем такую структуру:

1. Forest elephants
  1.1 Habitat

`<div>` не обязателен для того, чтобы объявить новую секцию. Опустим его. Получаем следующую разметку:

```
<h1>Forest elephants</h1>
  <p>In this section, we discuss the lesser known forest
elephants.
    ...this section continues...
  <h2>Habitat</h2>
  <p>Forest elephants do not live in trees but among them.
    ...this subsection continues...
  <h2>Diet</h2>
<h1>Mongolian gerbils</h1>
```

Имеем такую структуру:

1. Forest elephants
  1.1 Habitat
  1.2 Diet
2. Mongolian gerbils

`<div class="section" id="forest-elephants" ></div>` Назначение данного дива неизвестно — либо он используется только для стилизации `(class="section")`, либо он участвует в разметке страницы.
В HTML5 — все, что внутри `<body>` является частью секцией. Секции могут быть вложенными. Кроме главной секции (внутри `<body>`), границы секций могут быть объявлены явно или неявно. Неявно объявленные секции — это содержимое тэгов `<body>`, `<section>`, `<article>`, `<aside>`, `<footer>`, `<header>`, `<nav>`.

Как это можно сделать с помощью HTML5:

```
<section>
  <h1>Forest elephants</h1>
  <section>
    <h1>Introduction</h1>
    <p>In this section, we discuss the lesser known forest
elephants.</p>
  </section>
  <section>
    <h1>Habitat</h1>
    <p>Forest elephants do not live in trees but among them.</p>
  </section>
</section>
<footer>
  <p>(c) 2010 The Example company</p>
</footer>
```

Полученная структура:
1. Forest elephants
  1.1 Introduction
  1.2 Habitat

Использование новых семантических тэгов в старых браузерах:

```
section, article, aside, footer, header, nav, hgroup {
display:block;
}
```

Так как неизвестные элементы по умолчанию `display:inline`

IE ниже 9 версии не поддерживает стилизацию неизвестных элементов, поэтому необходим скрипт:

```
<!--[if lt IE 9]>
  <script>
    document.createElement("header" );
    document.createElement("footer" );
    document.createElement("section");
    document.createElement("aside"  );
    document.createElement("nav"    );
    document.createElement("article");
    document.createElement("hgroup" );
    document.createElement("time"   );
  </script>
<![endif]-->
```

или использовать готовый скрипт https://code.google.com/p/html5shiv/

```
<!--[if lt IE 9]>
  <script src="html5shiv.js"></script>
<![endif]-->
```

<article>

Независимый, отдельный кусок содержимого, предназначенный для многократного использования. Пример: пост на форуме, статья в газете или журнале, комментарий пользователя, интерактивный виджет и др.

<nav>

Секция с навигационными линками.

<header>

Внутри хэдэра расположена вводная информация или навигационная система. Может содержать лого, форму поиска и др.

<footer>

Футер для ближайший секции, т.е. для любого элемента `<article>`, `<aside>`, `<nav>`, `<section>`, `<blockquote>`, `<body>`, `<details>`, `<fieldset>`, `<figure>`, `<td>`

<aside>

Секция, содержимое которой, связано с контентом вокруг нее (сайдбар). Пример: глоссарий, реклама, биография автора и др.

Новые глобальные атрибуты

data-*

Дата-атрибуты предназначены для хранения данных в HTML, с целью последующего использования в скрипте.

Дата-атрибуты:

```
<article
      id="electriccars"
      data-columns="3"
      data-index-number="12314"
      data-parent="cars">
       ...
</article>
```

Получить дата-атрибуты с помощью JavaScript:

```
var article = document.querySelector('#electriccars'),
```

```
    data = article.dataset;

// data.columns -> "3"
// data.indexNumber -> "12314"
// data.parent -> "cars"
```

draggable
Означает, может ли элемент быть перемещен с помощью Drag and Drop API.

Взаимодействие с сервером

WebSockets
Открывают сессию между клиентом и сервером. С помощью данного API возможно отправлять сообщения на сервер и получать обратно ответ, слушая `MessageEvent` событие обработчиком событий.
Больше здесь: https://developer.mozilla.org/ru/docs/WebSockets

Server-sent Events
Сервер посылает push-сообщения клиенту формата `text/event-stream`. Клиент подписывается на эти события.
Больше здесь: https://developer.mozilla.org/en-US/docs/Server-sent_events

WebRTC
Технология, позволяющая обмениваться данными, а также аудио/видео потоком между клиентами (браузерами).
Больше здесь: http://www.webrtc.org/

Работа в оффлайне и хранение данных

Application cache
Позволяет веб-приложениям работать в оффлайн-режиме. С помощью AppCache разработчик может указать, какие ресурсы будут закешированы.
Больше здесь: https://developer.mozilla.org/en-US/docs/HTML/Using_the_application_cache

Online и offline события
С помощью свойства `navigator.onLine` есть возможность отслеживать, когда пользователь уходит в оффлайн и когда возвращается в онлайн.

В Firefox 3 появились новые события: `“online”, “offline”.` Эти события происходят, когда браузер переключается между online и offline режимами.
Больше здесь: https://developer.mozilla.org/en-US/docs/Online_and_offline_events

DOM Storage
Механизм, позволяющий хранить пары вида ключ-значение.

sessionStorage
Объект, доступный в течение жизни сессии страницы. Сессия страницы живет, пока браузер открыт, также сохраняется при перегрузки страницы. Если открыть страницу в новой вкладке или новом окне, новая сессия будет создана.

```
// Save data to the current session's store
sessionStorage.setItem("username", "John");
// Access some stored data alert("username = " +
sessionStorage.getItem("username"));
```

localStorage
То же самое, что и `sessionStorage`, однако является постоянным хранилищем.

Больше здесь: https://developer.mozilla.org/en-US/docs/Web/Guide/API/DOM/Storage

Пример: http://codebase.es/test/webstorage.html

IndexedDB

IndexedDB – API для работы с хранением большого объема структурированных данных в браузере с последующим быстрым поиском с использованием индексов.
https://developer.mozilla.org/en-US/docs/IndexedDB

Работа с файлами

File API предоставляет возможность выбрать файл на компьютере, а затем прочитать его содержимое.
Реализуется с помощью `<input>` элемента или с помощью drag'n'drop.
Больше здесь: https://developer.mozilla.org/en-US/docs/Using_files_from_web_applications

Мультимедиа

Аудио и видео

В HTML5 встроена поддержка медиа. Аудио и видео могут быть добавлены на страницу с помощью `<audio>` и `<video>` элементов.

Примеры:

```
<video src="http://v2v.cc/~j/theora_testsuite/320x240.ogg"
controls>
   Your browser does not support the <code>video</code> element.
</video>
```

```
<audio src="/test/audio.ogg" controls autoplay loop>
 <p>Your browser does not support the audio element.</p>
</audio>
```

Видео в разных форматах для разных браузеров:

```
<video controls>
   <source src="foo.ogg" type="video/ogg">
   <source src="foo.mp4" type="video/mp4">
   Your browser does not support the <code>video</code> element.
</video>
```

Подробнее про видео: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/video

Подробнее про аудио: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/audio

Камера

С помощью Camera API есть возможность делать фотографии и загружать на текущую веб страницу.

```
<input type="file" id="take-picture" accept="image/*">
```

Затем с помощью File API можно получить доступ к сделанной фотографии. Подробнее здесь: https://developer.mozilla.org/en-US/docs/Web/Guide/API/Camera

3D, графика, эффекты

<canvas>

Канвас используется для создания растровых изображений с помощью JavaScript.

Пример:

```
<html>
  <head>
   <script type="application/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>
```

Подробнее: https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Canvas_tutorial

WebGL

Позволяет создавать на JavaScript интерактивную 2D- и 3D-графику без использования плагинов.
Подробнее: https://developer.mozilla.org/en-US/docs/Web/WebGL

SVG

XML-разметка для создания векторной графики.
Подробнее: https://developer.mozilla.org/en-US/docs/Web/SVG

Производительность и интеграция

Некоторые нововведения:

Web Workers

Позволяют выполнять скрипты в других потоках.
Подробнее: https://developer.mozilla.org/en-US/docs/Web/Guide/Performance/Using_web_workers

History API

History API позволяет манипулировать историей браузера с помощью объекта history. Методы данного API позволяют перемещаться по истории браузера вперед и назад.

Пример:

```
window.history.back();
window.history.forward();
var numberOfEntries = window.history.length;
```

Подробнее: https://developer.mozilla.org/en-US/docs/Web/Guide/API/DOM/Manipulating_the_browser_history

Drag and drop

Подробнее: https://developer.mozilla.org/en-US/docs/DragDrop/Drag_and_Drop

Работа с устройствами

Некоторые нововведения:

Touch события

Для полноценной работы с устройствами, имеющими touch-экран, разработана поддержка touch событий.

```
el.addEventListener("touchstart", handleStart, false);
el.addEventListener("touchend", handleEnd, false);
el.addEventListener("touchcancel", handleCancel, false);
el.addEventListener("touchleave", handleEnd, false);
el.addEventListener("touchmove", handleMove, false);
```

Подробнее: https://developer.mozilla.org/en-US/docs/Web/Guide/Events/Touch_events

Геолокация

API геолокации позволяет получить доступ к местоположению пользователя. По соображениям конфиденциальности пользователь должен разрешить доступ к данной информации.

```
navigator.geolocation.getCurrentPosition(
      function(position) {
            do_something(position.coords.latitude,
            position.coords.longitude);
      }
);
```

Подробнее: https://developer.mozilla.org/en-US/docs/WebAPI/Using_geolocation

Ориентация устройства

Устройства могут сигнализировать об изменении своей ориентации с горизонтальной на вертикальную и обратно.

```
window.addEventListener("deviceorientation", handleOrientation,
true);
```

Подробнее: https://developer.mozilla.org/en-US/docs/WebAPI/Detecting_device_orientation

Стилизация

Новые возможности с помощью CSS3:

- Поддержка нескольких фонов:
  ```
  .myclass {   background: background1, background 2, ...,
  backgroundN; }
  ```
- Тени элементов
- Закругленные углы, картинки в качестве границ элементов (http://www.w3schools.com/css/css3_borders.asp)
- Анимация (http://www.w3schools.com/css/css3_animations.asp)
- Многоколоночный текст
  ```
  .content-box { column-width: 300px; }
  .content-box { column-count:3; }
  ```
  (http://www.w3schools.com/css/css3_multiple_columns.asp)
  И др.

Доступность контента для людей с ограниченными возможностями

Сайтом могут пользоваться люди с физическими отклонениями (слабовидящие, слабослышащие и т.д.), а также люди, использующие старые устройства, устройства без мыши, клавиатуры и т.д. Некоторые пользователи могут быть ограничены в движении. Все это необходимо учитывать при разработке веб-страниц.

Полное руководство по доступности контента для людей с ограниченными возможностями здесь: http://www.w3.org/TR/WCAG20/19

Некоторые рекомендации:

- Создавать страницы, которые при просмотре их с помощью различных устройств либо людьми с разными отклонениями или ограничениями, могут все равно быть представлены без потери данных или структуры:
  - Отделять содержимое и представление (не использовать устаревшие тэги `font, b, i, u` (заменять css-классами); описывать стили в отдельном файле)
  - Предоставлять текстовую версию любого нетекстового контента (аудио, видео, изображения) для его возможного преобразования в альтернативные формы, удобные для различных пользователей.

```
<img
src="https://developer.cdn.mozilla.net/media/img/mdn-logo-sm.png" alt="MD Logo" />
```

    - Не создавать страницы, которые рассчитаны только на один тип устройств. То есть страницы должны быть доступны и в случае просмотра их с черно-белого экрана, экрана с низким разрешением, в случае его отсутствия, а также отсутствия мыши, клавиатуры и др.
    - Не полагаться на цвет исключительно.
    - Использовать разметку семантически. В случае неправильного использования элементов разметки (таблицы для организования лэйаута, заголовки для увеличения текста и т.д.), данная страница становится трудной для понимания и навигации пользователям со специализированными устройствам (например, устройство чтения для слабовидящих).
    - Убедиться, что страницы, использующие новейшие технологии, доступны и для устройств, их не поддерживающих.
- Разрабатывать контент с понятными элементами управления и навигации. Страницы должны отображаться и функционировать предсказуемым образом.
- Указывать язык, на котором написан документ. Если какая-то часть содержимого написана на другом языке, указывать язык для элемента, окружающего это содержимое. Подробнее здесь: http://www.w3.org/International/questions/qa-html-language-declarations
- Использовать рекомендации W3C (не использовать устаревшие технологии и подходы, использовать наиболее свежие версии языков и технологий)
- Задавать логическую последовательность перехода по управляющим элементам с помощью клавиши tab.
- Предоставлять горячие клавиши для наиболее важных управляющих элементов.

Литература, инструментарий

Литература

Список всех элементов, атрибутов. Примеры для каждого из них:
http://www.w3schools.com/html/

Спецификация HTML5:
http://www.w3.org/TR/html5/

Поддерживается ли какая-либо HTML5 технология в конкретном браузере:
http://caniuse.com/#cats=HTML5

Обучающие примеры, разделенные по уровням:
http://www.htmldog.com/guides/html/

Ресурс со статьями и примерами:
http://www.html5rocks.com/

Справочные материалы, документация, инструменты, полезные ссылки:
http://www.html5rocks.com/en/resources

Видео уроки:
http://www.lynda.com/search?q=html

Инструментарий

Валидатор разметки:
http://validator.w3.org/

Плагин для Firefox, позволяет в браузере инспектировать и изменять HTML и CSS
http://getfirebug.com/

Набор веб-разработчика, встроенный в Google Chrome:
https://developers.google.com/chrome-developer-tools/