Палитра элементов HTML

Содержание

- **Категории контента**
- ** : Элемент разделения контента**
- **<section>**
- **<article>**
- **<nav>Элемент секции навигации**
- **Добавление векторной графики в веб-документ (встроенный контент)**
- **Видео и аудио контент**

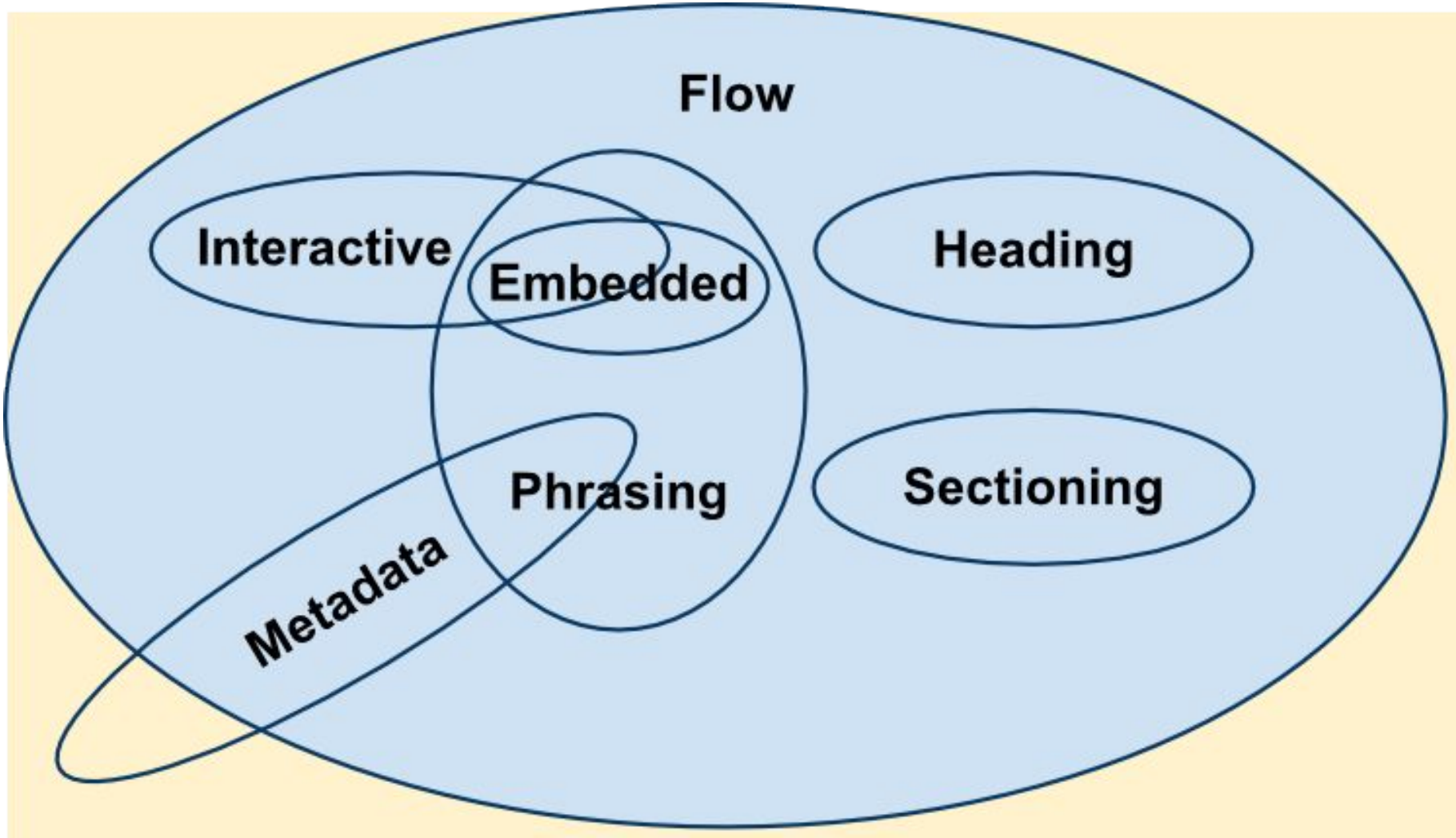
Категории контента

Каждый элемент HTML принадлежит некоторому количеству категорий контента, которые объединяют элементы с общим набором характеристик.

Существует три типа категорий контента:

- Основные категории контента, описывающие общие для многих элементов правила;
- Категории контента для элементов форм, описывающие общие правила для элементов форм;
- Особые категории контента, описывающие редкие категории, актуальные только для нескольких элементов, возможно, лишь в особом контексте.

Категории контента



Основные категории контента

Метаданные

Элементы, принадлежащие к *категории метаданных,* изменяют отображение или поведение HTML-документа, связывают его с другими документами и предоставляют другую *дополнительную* информацию о документе.

Элементами метаданных являются: <base>, <command>, <link>, <meta>, <noscript>, <script>, <style> и <title>.

Основной поток

flow

Элементы основного потока обычно содержат текст и встроенный контент.

<a>, <abbr>, <address>, <article>, <aside>, <audio>, <b>,<bdo>, <bdi>, <block quote>,
, <button>, <canvas>, <cite>, <code>, <command>, <data>, <datalist>, <del>, <details>, <dfn>, , <dl>, , <embed>, <fieldset>, <figure>, <footer>, <form>, <h1> (en-US), <h2> (en-US), <h3> (en-US), <h4> (en-US), <h5> (en-US), <h6> (en-US), <header>, <hgroup>, <hr>, <i> (en-US), <iframe> (en-US), <img>, <input>, <ins>, <kbd>, <keygen> (en-US), <label>, <main>, <map>, <mark>, <math>, <menu>, <meter>, <nav>, <noscript>, <object> (en-US), <ol>, <output>, <p>, <pre>, <progress>, <q> (en-US), <ruby>, <s> (en-US), <samp> (en-US), <script>, <section>, <select>, <small> (en-US), , , <sub> (en-US), <sup> (en-US), <svg>, <table> (en-US), <template>, <textarea> (en-US), <time>, <ul>, <var> (en-US), <video>, <wbr> и Текст.

Секционный контент

Подобные элементы создают секции (блоки) в текущей структуре документа, определяющие область действия **заголовочного контента** и элементов <header> и <footer>

К этой категории принадлежат элементы <article>, <aside>, <nav> и <section>.

Заголовочный контент

Заголовочный контент задаёт заголовок секции, явно отмеченной структурным элементом или неявно – самим заголовочным.

Данной категории принадлежат такие элементы, как <h1> (en-US), <h2> (en-US), <h3> (en-US), <h4> (en-US), <h5> (en-US), <h6> (en-US) и <hgroup>.

Элемент <header> сам по себе не является заголовочным, хотя с большой вероятностью содержит контент такого типа.

Фразовый контент

Фразовый контент определяет текст и его формат. Серии фразового контента образуют параграфы.

К данной категории принадлежат следующие элементы: <abbr>, <audio>, <b>, <bdo>,
, <button>, <canvas>, <cite>, <code>, <command>, <data>, <datalist>, <dfn>, , <embed>, <i> (en-US), <iframe> (en-US), <img>, <input>, <kbd>, <keygen> (en-US), <label>, <mark>, <math>, <meter>, <noscript>, <object> (en-US), <output>, <progress>, <q> (en-US), <ruby>, <samp> (en-US), <script>, <select>, <small> (en-US), , , <sub> (en-US), <sup> (en-US), <svg>, <textarea> (en-US), <time>, <var> (en-US), <video>, <wbr> и обычный текст (не только состоящий из символов пробелов).

Фразовый контент

Ещё несколько элементов входят в данную категорию при соблюдении особых условий:

<a>, если содержит в себе только фразовый контент

<area>, только внутри элемента <map>

<del>, если содержит в себе только фразовый контент

<ins>, если содержит в себе только фразовый контент

<link>, при наличии атрибута itemprop (en-US)

<map>, если содержит в себе только фразовый контент

<meta>, при наличии атрибута itemprop (en-US)

Встроенный контент

Встроенный контент импортирует в документ другой ресурс или вставляет содержимое на другом языке разметки или принадлежащее другому пространству имён. Элементами данной категории являются: <audio>, <canvas>, <embed>, <iframe> (en-US), <img>, <math>, <object> (en-US), <svg>, <video>.

Интерактивный контент

К интерактивному контенту относятся элементы, который специально разработаны для взаимодействия с пользователем. В данную категорию входят <a>, <button>, <details>, <embed>, <iframe> (en-US), <keygen> (en-US), <label>, <select> и <textarea> (en-US). Некоторые элементы считаются интерактивным контентом только при соблюдении определённых условий:

<audio>, если указан атрибут controls

<img>, если указан атрибут usemap

<input>, если атрибут type не скрыт

<menu>, если атрибут type имеет значение toolbar

<object> (en-US), если указан атрибут usemap (en-US)

<video>, если указан атрибут controls

Контент форм

Контент форм включает в себя элементы, у которых есть владелец формы, обозначенный атрибутом **form**. Владельцем формы является либо элемент <form>, либо элемент, id которого указан в атрибуте **form**.

<button>
<fieldset>
<input>
<keygen> (en-US)
<label>
<meter>
<object> (en-US)
<output>
<progress>
<select>
<textarea> (en-US)

Данная категория включает несколько подкатегорий:

Listed Элементы, перечисленные в IDL коллекциях form.elements (en-US) и fieldset.elements. Включают <button>, <fieldset>, <input>, <keygen> (en-US), <object> (en-US), <output>, <select>, и <textarea> (en-US).

Labelable Элементы, которые могут ассоциироваться с элементами <label>. Включают <button>, <input>, <keygen> (en-US), <meter>, <output>, <progress>, <select> и <textarea> (en-US).

Submittable Элементы, которые могут использоваться для построения набора данных формы при отправке на сервер. Включают <button>, <input>, <keygen> (en-US), <object> (en-US), <select> и <textarea> (en-US).

Resettable Элементы, которые могут быть затронуты при сбросе данных формы. Включают <input>, <keygen> (en-US), <output>,<select> и <textarea> (en-US).

Вторичные категории

Существуют некоторые вторичные классификации элементов, о которых тоже полезно знать.

Элементы поддержки скриптов

Элементы поддержки скриптов – это элементы, которые напрямую не влияют на отрисовку документа.

Вместо этого они служат для внедрения скриптов, путём либо содержания кода скрипта напрямую, либо указания данных, которые будут использованы скриптами.

Элементами поддержки скриптов являются:

<script>

<template>

: Элемент разделения контента

Элемент разделения контента HTML (** **) является универсальным контейнером для потокового контента. Он не влияет на контент или макет до тех пор, пока не будет стилизован с помощью CSS.

HTML

```
<div class="warning">
   <img src="/media/examples/leopard.jpg"
        alt="An intimidating leopard.">
   <p>Beware of the leopard</p>
</div>
```

CSS

```
.warning {
   border: 10px ridge #f00;
   background-color: #ff0;
   padding: .5rem;
   display: flex;
   flex-direction: column;}
.warning img {
   width: 100%;}
.warning p {
   font: small-caps bold 1.2rem sans-serif;
   text-align: center;}
```

Элемент

Являясь "чистым" контейнером, элемент , по существу, не представляет ничего. Между тем, он используется для группировки контента, что позволяет легко его стилизовать, используя атрибуты class или id, помечать раздел документа, написанный на разных языках (используя атрибут lang).

Категории

Категории контента	Потоковый контент, явный контент.
Разрешённое содержимое	Потоковый контент или (в WHATWG HTML), если родительским является элемент <dl>: один или несколько элементов <dt>, сопровождаемых одним или более элементами <dd>, в ряде случаев смешанных с элементами <script> и <template>.
Пропуск тега	Ни одного; Оба тега, открывающий и закрывающий, являются обязательными.
Разрешённые родительские элементы	Любой элемент, который разрешает потоковый контент в качестве содержимого.. Или (в WHATWG HTML): элемент <dl>.
Разрешённые роли ARIA	Любые
DOM-интерфейс	HTMLDivElement

<dl>: Элемент списка описаний

HTML-элемент <dl> (*от англ. Description List*) представляет собой список описаний. Этот элемент служит контейнером для списка пар терминов (определяемых элементом <dt>) и их описаний (определяемых элементами <dd>). Этот элемент обычно используют при создании глоссария или для отображения метаданных (списка пар ключ-значение).

```
<dl>
  <dt>Beast of Bodmin</dt>
  <dd>A large feline inhabiting Bodmin Moor.</dd>
```

ARIA

Accessible Rich Internet Applications **(ARIA)** определяет способ сделать веб контент и веб приложения (особенно те, которые разработаны с помощью Ajax и JavaScript) более доступными для людей с ограниченными возможностями.

Например, ARIA делает доступным навигационные маркеры, JavaScript виджеты, подсказки на форме, сообщения об ошибках, автоматические обновления и многое другое.

ARIA - это набор специальных атрибутов, которые могут быть добавлены в любую разметку, но особенно подходят для HTML. Атрибут role определяет тип объекта (такие как статья, оповещение или ползунок). Дополнительные ARIA атрибуты предоставляют другие полезные возможности, такие как описания для форм или текущее значение индикатора выполнения.

Поддержка ARIA реализована в большинстве современных браузеров и программах экранного доступа. Конечно, реализации различаются, и старые технологии не поддерживают их полностью (либо вообще не поддерживают). Используйте постепенно деградирующий "щадящий" ARIA, или просите пользователей использовать новые технологии.

Атрибуты

К этому элементу применимы глобальные атрибуты.

Примечание: Атрибут align устарел и вышел из употребления; не используйте его больше. Вместо этого, вам следует использовать свойства CSS или методы, такие как CSS Grid или CSS Flexbox для выравнивания и изменения положения элементов на странице.

Примечание

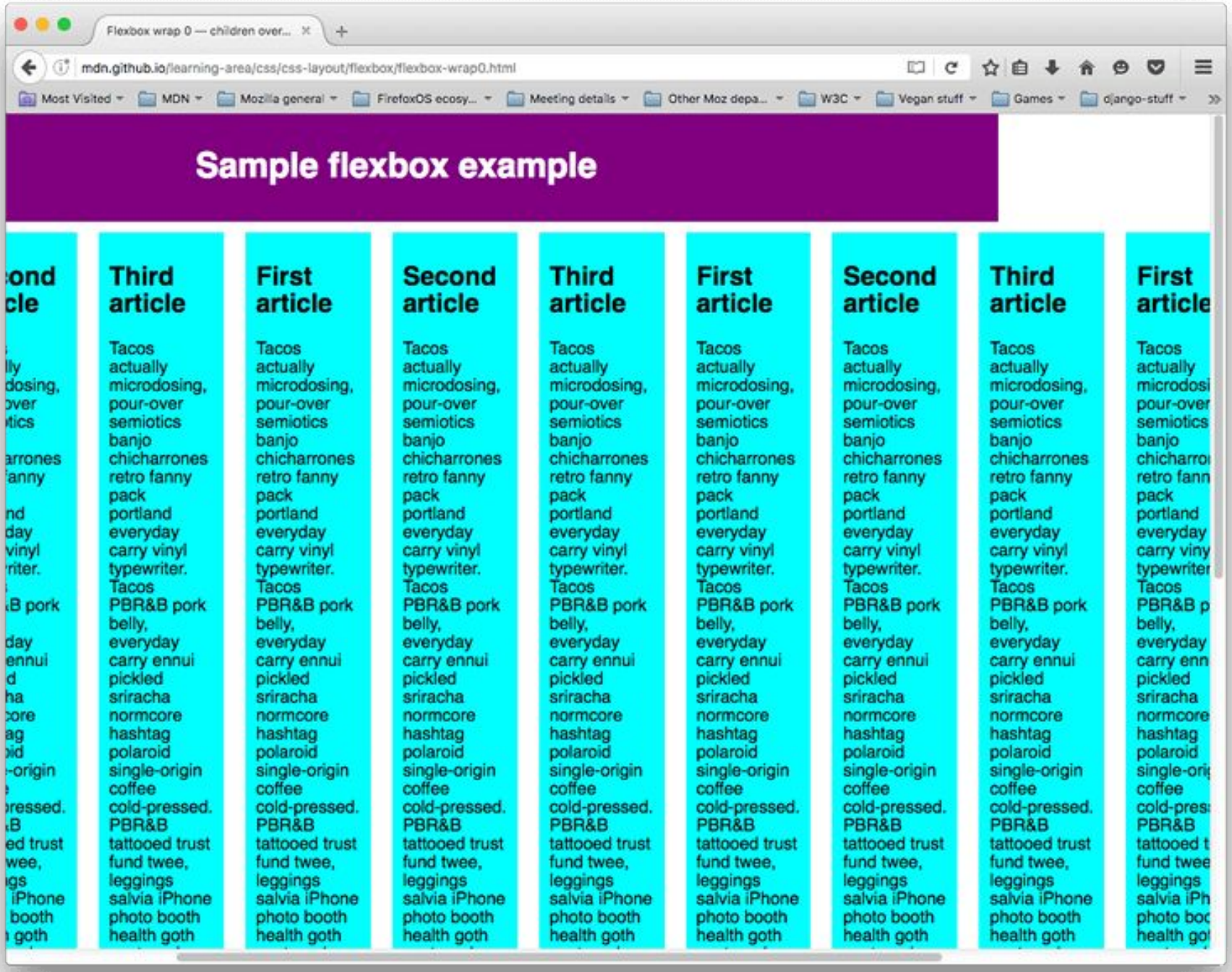
Элемент следует использовать только в том случае, если никакой другой семантический элемент (такой как <article> или <nav>) не подходит.

СМ. Пример audi a7

CSS Grid

- One Two Three Four Five Six

```
.wrapper { display: grid;
grid-template-columns: repeat(3,
1fr);
 grid-gap: 10px; grid-auto-rows:
minmax(100px, auto); }
.one { grid-column: 1 / 3;
grid-row: 1; }
.two { grid-column: 2 / 4;
 grid-row: 1 / 3; }
.three { grid-column: 1;
grid-row: 2 / 5; }
.four { grid-column: 3; grid-row:
3; }
.five { grid-column: 2; grid-row:
4; }
.six { grid-column: 3; grid-row:
4; }
```


Flexbox wrap 0 — children over...
mdn.github.io/learning-area/css/css-layout/flexbox/flexbox-wrap0.html
Most Visited
MDN
Mozilla general
FirefoxOS ecosy...
Meeting details
Other Moz depa...
W3C
Vegan stuff
Games
django-stuff
Sample flexbox example
Third article
Tacos actually microdosing, pour-over semiotics banjo chicharrones retro fanny pack portland everyday carry vinyl typewriter. Tacos PBR&B pork belly, everyday carry ennui pickled sriracha normcore hashtag polaroid single-origin coffee cold-pressed. PBR&B tattooed trust fund twee, leggings salvia iPhone photo booth health goth
First article
Second article
Third article
First article
Second article
Third article
First article

Стилизованный пример

Этот пример создаёт прямоугольник с тенью, применяя стили к с помощью CSS. Заметьте, что использование атрибута class на элементе даёт применение стилей "shadowbox" (в дословном переводе означает "теневая коробка") к элементу.

HTML

```
<div class="shadowbox"> <p>Вот очень интересная заметка в прекрасном прямоугольнике с тенью.</p> </div>
```

CSS

```
.shadowbox { width: 15em; border: 1px solid #333; box-shadow: 8px 8px 5px #444; padding: 8px 12px; background-image: linear-gradient(180deg, #fff, #ddd 40%, #ccc); }
```

<section>

HTML-элемент <section> представляет собой автономный раздел — который не может быть представлен более точным по семантике элементом — внутри HTML-документа. Как правило, но не всегда, разделы имеют заголовок.

```
<h1>Choosing an Apple</h1>
<section>
  <h2>Introduction</h2>
  <p>This document provides a guide to help with the important task of choosing the correct Apple.</p>
</section>
<section>
  <h2>Criteria</h2>
  <p>There are many different criteria to be considered when choosing an Apple — size, color, firmness, sweetness, tartness...</p>
</section>
```

Например, меню навигации должно быть помещено в элемент <nav>, но список результатов поиска и отображение карты с её элементами управления не имеют специфических элементов и могут быть помещены в <section>.

Примечание: Если содержимое элемента имеет смысл объединить как единое целое (в цельный и независимый блок), то элемент <article> может стать лучшим выбором.

категории

категории контента	Основной поток, секционный контент, явный контент.
Разрешённое содержимое	Основной поток
Пропуск тега	Ни одного; Оба тега, открывающий и закрывающий, являются обязательными.
Разрешённые родительские элементы	Любой элемент, который разрешает контент основного потока в качестве содержимого. Обратите внимание, что элемент <section> не должен быть потомком элемента <address>.
Разрешённые роли ARIA	alert, alertdialog, application, banner, complementary, contentinfo, dialog, document, feed, log, main, marquee, navigation, search, status, tabpanel
DOM-интерфейс	HTMLElement

Атрибуты

К этому элементу применимы только глобальные атрибуты.

Примечание

Каждый элемент <section> должен быть идентифицирован, обычно путём добавления заголовка (элементы <h1> (en-US)-<h6> (en-US)) в качестве дочернего элемента.

Если имеет смысл по-особому объединить содержимое элемента <section> (например, сделать цельным и независимым разделом HTML-документа), используйте вместо него элемент <article>.

Не используйте элемент <section> как общий контейнер; для этого есть , особенно когда секционирование применяется только для стилизации. На практике раздел должен логически выделяться в структуре документа.

<article>

HTML-элемент <article> представляет самостоятельную часть документа, страницы, приложения или сайта, предназначенную для независимого распространения или повторного использования.

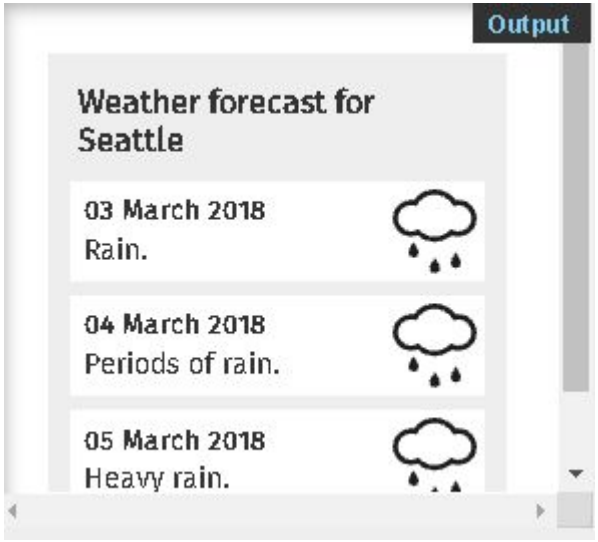
Этот элемент может представлять статью на форуме, статью в журнале или газете, запись в блоге или какой-либо другой самостоятельный фрагмент содержимого.

Пример HTML

```
<article class="forecast">
   <h1>Weather forecast for Seattle</h1>
   <article class="day-forecast">
      <h2>03 March 2018</h2>
      <p>Rain.</p>
   </article>
   <article class="day-forecast">
      <h2>04 March 2018</h2>
      <p>Periods of rain.</p>
   </article>
   <article class="day-forecast">
      <h2>05 March 2018</h2>
      <p>Heavy rain.</p>
   </article>
</article>
```

Пример CSS

```
.forecast {
   margin: 0;
   padding: .3rem;
   background-color: #eee;
   font: 1rem 'Fira Sans', sans-serif;
}
.forecast > h1,
.day-forecast {
   margin: .5rem;
   padding: .3rem;
   font-size: 1.2rem;
}
.day-forecast {
   background: right/contain content-box border-box no-repeat
      url('/media/examples/rain.svg') white;
}
.day-forecast > h2,
.day-forecast > p {
   margin: .2rem;
   font-size: 1rem;
}
```



URL()

Конструктор **URL()** возвращает вновь созданный URL объект, отражающий URL, определяемый параметрами.

Если предоставленный базовый URL-адрес или итоговый URL-адрес не является валидным, то будет сгенерировано JavaScript исключение TypeError.

Синтаксис

const url = new URL(url [, base])

Параметры

urlСтрока типа USVString или любой объект, который может быть преобразован в строку, включая, например, элементы <a> и <area>, представляющие абсолютный и относительный URL-адрес соответственно.

Если url это относительный URL-адрес, то параметр base становится обязательным, так как он будет использован в качестве базового URL-адреса.

Если же url — абсолютный URL-адрес, то значение параметра base будет проигнорировано.

base

НеобязательныйСтрока типа USVString. Используется в качестве базового адреса, когда url это относительный URL-адрес. Значение по умолчанию: undefined.

Атрибуты

К этому элементу применимы только глобальные атрибуты.

Примечание

Каждый элемент <article> должен быть идентифицирован, обычно путём добавления заголовка (элементы <h1>-<h6>) в качестве дочернего элемента.

Когда элемент <article> является вложенным, внутренний элемент представляет собой контент связанный с внешним элементом. Например, комментарии к публикации в блоге могут быть элементами <article>, вложенными в другой <article>, являющийся публикацией в блоге.

Информация об авторе в элементе <article> может быть представлена через элемент <address>, но это не применимо к вложенным элементам <article>.

Дата и время публикации в элементе <article> могут быть описаны с помощью атрибута datetime элемента <time>. *Обратите внимание, что атрибут pubdate элемента <time> больше не является частью стандарта W3C HTML5.*

Пример

```
<article class="film_review">
<header> <h2>Парк Юрского периода</h2>
    </header>
 <section class="main_review">
 <p>Динозавры были величественны!</p>
    </section>
 <section class="user_reviews">
<article class="user_review">
 <p>Слишком страшно для меня.</p>
<footer>
 <p> Опубликовано <time datetime="2015-05-16
    19:00">16 мая</time> Лизой. </p>
</footer>
</article>
```

Элемент секции навигации <nav>

HTML-элемент <nav> определяет отдельную секцию документа, назначение которой обозначение ссылок навигации (как внутри текущего документа, так и ведущих на другую страницу).

В качестве примера такой секции можно привести меню, якорные ссылки.

Атрибуты

Этот элемент поддерживает глобальные атрибуты.

Примечание

Не обязательно все ссылки должны быть обёрнуты в <nav>. <nav> следует использовать лишь для главных навигационных блоков. Например, <footer> часто содержит список ссылок, которые не стоит оборачивать в **<nav>** .

Документ может содержать несколько **<nav>** элементов. Например, один для навигации по сайту, второй для навигации по странице.

Пользовательские агенты, такие как устройства чтения с экрана, предназначенные для людей с плохим зрением, могут использовать этот элемент, чтобы определить следует ли пускать рендеринг содержимого навигации.

Пример nav

```
<nav class="menu">
 <ul>
 <li>
<a href="#">Главная</a>
</li>
<li>
<a href="#">О нас</a>
</li>
 <li>
<a href="#">Контакты</a>
</li>
</ul>
</nav>
```

Пример <aside>

```
<p> Саламандры - это группа
      амфибий с внешностью
      ящерицы, включая короткие
      ноги и хвост как в
      личиночной, так и во
      взрослой формах.
<aside>
<p> Грубокожий тритон защищает
      себя смертельным
      нейротоксином.</p>
 </aside>

<p > Несколько видов саламандр
      обитают в умеренных
      тропических лесах
      Тихоокеанского северо-
      запада, включая Энсатину,
      Северо-западную
      саламандру и тритона с
      грубой кожей. Большинство
      саламандр ведут ночной
      образ жизни и охотятся на
      насекомых, червей и других
      мелких существ. </p>
```

```
aside {
   width: 40%;
   padding-left: .5rem;
   margin-left: .5rem;
   float: right;
   box-shadow: inset 5px 0 5px
-5px #29627e;
   font-style: italic;
   color: #29627e;
}

aside > p {
   margin: .5rem;
}
```

HTML-элемент <aside> представляет собой часть документа, чьё содержимое только косвенно связанно с основным содержимым документа. Чаще всего представлен в виде боковой панели, сносок или меток.

Добавление векторной графики в веб-документ (встроенный контент)

Векторная графика очень полезна во многих случаях. Она имеет малые размеры файла и высокую масштабируемость – при увеличении масштаба пиксели не увеличиваются вместе с графикой.

Более того, файлы векторных изображений намного меньше растровых, т.к. в них содержится алгоритмы построения вместо информации о каждом пикселе.

Что такое SVG?

SVG это язык на базе XML для описания векторных изображений.

По сути это язык разметки, как и HTML, только содержащий множество различных элементов для определения фигур вашего изображения, а также параметров их отображения.

SVG предназначен для разметки графики, а не содержимого.

В простейшем случае, вы можете использовать элементы для создания простых фигур, таких как <circle>(круг) и <rect>(прямоугольник).

Более сложные SVG элементы включают

- <feColorMatrix> (en-US) (разложение цвета с использованием матрицы),
- <animate> (анимация частей вашего векторного изображения) и
- <mask> (en-US) (применение маски к изображению.)

В качестве простого примера, следующий код создаёт круг и прямоугольник:

```
<svg version="1.1" baseProfile="full" width="300"
    height="200" xmlns="http://www.w3.org/2000/svg">
 <rect width="100%" height="100%" fill="black" />
<circle cx="150" cy="100" r="90" fill="blue" />
</svg>
```

Для создания SVG изображений используются редакторы векторной графики, такие как Inkscape или Illustrator.

Данные приложения позволяют создавать различные изображения, используя множество графических инструментов, и создавать приближения фотографий (например опция Trace Bitmap feature приложения Inkscape.)

Дополнительные преимущества SVG:

- Текст в векторном изображении остаётся машинописным (то есть доступным для поисковика, что улучшает SEO).
- SVG легко поддаются стилизации/программированию (scripting), потому что каждый компонент изображения может быть стилизован с помощью CSS или запрограммирован с помощью JavaScript.

Так почему же тогда вообще используют растровые изображения, а не только SVG?

Дело в том, что SVG имеет ряд недостатков:

- SVG может очень быстро стать сложным в том смысле, что размер файла увеличивается; сложные SVG-изображения также создают большую вычислительную нагрузку на браузер.
- SVG может быть сложнее создать, нежели растровое изображение, в зависимости от того, какое изображение необходимо создать.
- не поддерживается старыми версиями браузеров, то есть не подойдёт для сайтов, поддерживающих Internet Explorer 8 или старее.
- В целом, растровая графика лучше подходит для сложных изображений, например, фотографий.

Добавление SVG на страницы

Рассмотрим различные варианты, с помощью которых можно добавить SVG векторную графику на веб-страницу.

Быстрый путь: <img>

Чтобы встроить SVG используя элемент <img>, вам просто нужно сослаться на него в атрибуте src, как и следовало ожидать. Вам понадобится атрибут height или width (или оба, если ваш SVG не имеет собственного соотношения сторон).

```
<img src="equilateral.svg" alt="triangle with all three sides equal" height="87px" width="100px" />
```

Минусы <img>

Вы не можете изменять изображение с помощью JavaScript.

Если вы хотите управлять содержимым SVG с помощью CSS, вы должны использовать встроенные CSS стили в своём SVG коде. (Внешние таблицы стилей, вызываемые из файла SVG, не действуют.)

Вы не можете изменить стиль изображения с помощью псевдоклассов CSS (например :focus).

Как включить SVG в ваш HTML код

Вы можете открыть файл SVG в текстовом редакторе, скопировать этот код и вставить его в ваш HTML документ — такой приём иногда называют встраиванием SVG (**SVG inline** или **inlining SVG**). Убедитесь, что фрагмент вашего SVG кода начинается и заканчивается с тегов <svg></svg> (не включайте ничего, кроме них). Вот очень простой пример того, что вы можете вставить в ваш документ:

```
<svg width="300" height="200"> <rect width="100%" height="100%" fill="green" />
</svg>
```

Плюсы

- Вставка вашего SVG путём **SVG inline** позволяет сохранить HTTP запросы и, следовательно, может уменьшить время загрузки.
- Вы можете присваивать class-ы и id элементам SVG и стилизовать их при помощи CSS, либо в пределах SVG, либо внутри SVG, либо там, где вы размещаете правила стиля CSS для вашего HTML документа.
- По факту вы можете использовать любой атрибут представления SVG как свойство CSS.
- **SVG inline** единственный метод, который позволяет вам использовать CSS-взаимодействия (как :focus) и CSS-анимацию на вашем SVG изображении (даже в вашей обычной таблице стилей).
- Вы можете разметить SVG как гиперссылку, обернув в элемент <a>.

Минусы

- Этот метод подходит, только если вы используете SVG лишь в одном месте. Дублирование делает обслуживание ресурсоёмким.
- Дополнительный SVG код увеличивает размер вашего HTML файла.
- Браузер не может кешировать встроенный SVG, так как он кеширует обычные изображения.
- Вы можете добавить альтернативный вариант в элементе <foreignObject>, но браузеры поддерживающие SVG будут продолжать загружать все альтернативные изображения.

Как встраивать SVG при помощи <iframe>

Вы можете открывать ваши SVG изображения в браузере просто как веб-страницы.

```
<iframe src="triangle.svg" width="500" height="500" sandbox>
 <img src="triangle.png" alt="Triangle with three unequal sides" />
</iframe>
```

Это - определённо не самый лучший метод для выбора:

Минусы

У iframe-ов есть резервный механизм, но браузеры отображают резервный вариант только если они вообще не поддерживают iframe-ы.

Более того, до тех пор пока SVG и ваша текущая веб-страница имеют одинаковый origin(Браузер отправляет серверу **первичные данные** - протокол, *хостинг, домен, порт соединения через* URL. Два объекта одинаковые если протокол, хост, домен и порт одинаковые.),

вы не можете использовать JavaScript на вашей основной веб-странице, чтобы манипулировать SVG.

Видео и аудио контент

В спецификации HTML5 были добавлены функции, с элементами <video> и <audio>, и некоторые новые JavaScript API для их управления.

Прежде всего, вы также должны знать, что есть немало OVPs (провайдеров онлайн-видео) вроде YouTube, Dailymotion и Vimeo, а также онлайн аудио-провайдеров вроде Soundcloud.

Такие компании предлагают удобный и простой способ размещения и потребления видео, поэтому вам не нужно беспокоиться об огромном потреблении трафика.

OVP даже обычно предлагают готовый код для встраивания видео и аудио в ваши веб-страницы

Элемент <video>

Элемент <video> позволяет вам вставлять видео достаточно легко. Очень простой пример выглядит так:

```
<video src="rabbit320.webm" controls>
 <p>Ваш браузер не поддерживает HTML5 видео.
 Используйте <a href="rabbit320.webm">ссылку на видео</a>
 для доступа.</p>
 </video>
```

Описание параметров:

src Точно так же, как для элемента <img> , атрибут src (source — источник) содержит путь к видео, которое вы хотите внедрить. Он работает точно так же.

Controls

Пользователи должны иметь возможность контролировать воспроизведение видео и аудио.

Вы должны либо использовать атрибут controls, чтобы использовать встроенный в браузер интерфейс управления или создать собственный интерфейс, используя соответствующие JavaScript API. Как минимум, интерфейс должен включать способ запуска и остановки медиа-носителя и регулировки громкости.

Поддержка нескольких форматов

Форматы, такие как MP3, MP4 и WebM, называются **форматами контейнеров**. Они содержат различные части, которые составляют всю песню или видео — например, звуковую дорожку, видеодорожку (в случае видео) и метаданные для описания представленного носителя.

Аудио и видео треки также находятся в разных форматах, например:

- Контейнер WebM обычно загружает звук Ogg Vorbis с видео VP8 / VP9. Поддерживается в основном в Firefox и Chrome.
- Контейнер MP4 часто включает аудио AAC или MP3 с видео H.264. Поддерживается в основном в Internet Explorer и Safari.
- Более старый контейнер Ogg имеет тенденцию идти с аудио Ogg Vorbis и видео Ogg Theora. Поддерживалось главным образом в Firefox и Chrome, но было вытеснено более качественным форматом WebM.

Вышеупомянутые форматы существуют для сжатия видео и аудио в управляемые файлы (необработанные видео и аудио очень большие). Браузеры содержат разные кодеки, вроде Vorbis или H.264, которые используются для преобразования сжатого звука и видео в бинарные данные и обратно.

<source>

Здесь мы изъяли атрибут src из нашего тега <video>, и вместо этого включали отдельные элементы <source>, каждый из которых ссылается на собственный источник. В этом случае браузер пройдётся по элементам <source> и начнёт воспроизводить первый из них, который имеет поддерживаемый кодек.

Включение источников WebM и MP4 должно быть достаточно для воспроизведения вашего видео на большинстве платформ и браузеров в наши дни.

```
<video controls>
<source src="rabbit320.mp4" type="video/mp4">
<source src="rabbit320.webm" type="video/webm">
 <p>Ваш браузер не поддерживает HTML5 видео. Вот
 <a href="rabbit320.mp4">ссылка на видео</a> взамен.</p>
</video
```

Литература

- Элемент - HTML | MDN (mozilla.org)
- <video> - HTML | MDN (mozilla.org)