HTML 5

Введение

HTML-документ — это обычный текстовый документ, который состоит из дерева HTML-элементов и текста. Каждый элемент обозначается в исходном документе начальным (открывающим) и конечным (закрывающим) тегом (за редким исключением).

Начальный тег показывает, где начинается элемент, конечный — где заканчивается. **Закрывающий тег** образуется путем добавления слэша `/` перед именем тега: `<имя тега>…</имя тега>`. Между начальным и закрывающим тегами находится содержимое элемента — контент.

Элементы могут вкладываться друг в друга, например, `<p><i>Текст</i></p>`. При вложении следует соблюдать порядок их закрытия **(принцип «матрёшки»)**, например, следующая запись будет неверной: `<p><i>Текст</p></i>`.

HTML-элементы могут иметь атрибуты (глобальные, применяемые для всех HTML-элементов, и собственные). Атрибуты прописываются в открывающем теге элемента и содержат имя и значение, указываемые в формате `имя атрибута="значение"`. Атрибуты позволяют изменять свойства и поведение элемента, для которого они заданы.

```
<p class="description">Текст</p>
```

Структура HTML-документа

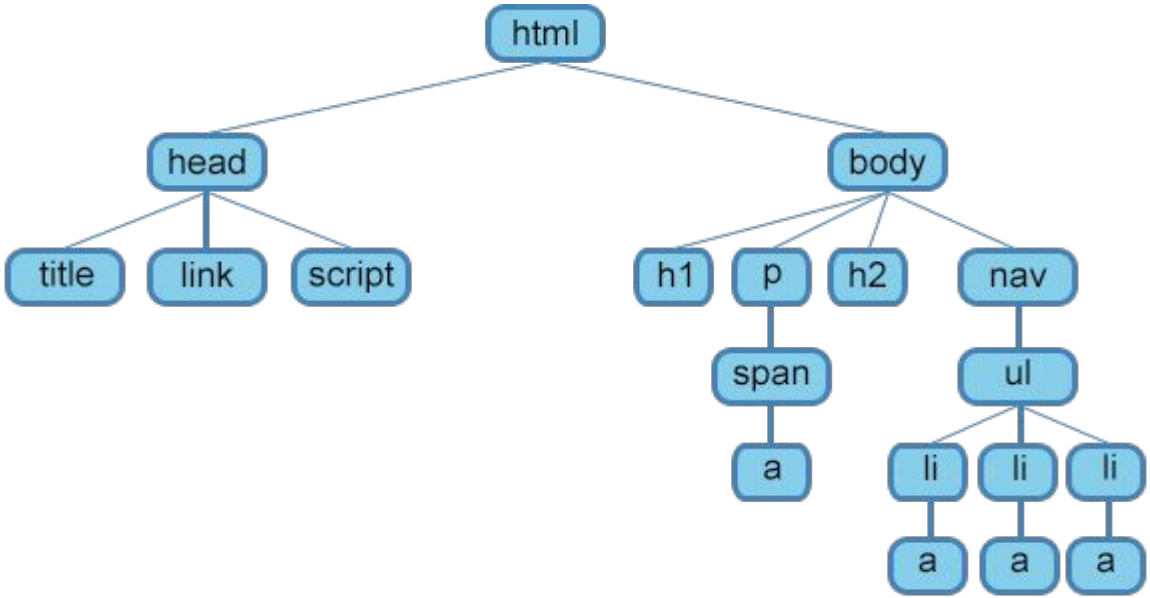
Язык HTML следует правилам, которые содержатся в файле объявления типа документа *(Document Type Definition, или DTD)*. DTD представляет собой XML-документ, определяющий, какие теги, атрибуты и их значения действительны для конкретного типа HTML. Для каждой версии HTML есть свой DTD.

DOCTYPE отвечает за корректное отображение веб-страницы браузером. DOCTYPE определяет не только версию HTML (например, html), но и соответствующий DTD-файл в Интернете.

Для HTML5 `<!DOCTYPE html>` его мы и будем использовать.

Элементы, находящиеся внутри элемента `<html>`, образуют дерево документа, так называемую **объектную модель документа, DOM (document object model)**. При этом элемент `<html>` является корневым элементом.

Структура HTML-документа



Структура HTML-документа

Предок — элемент, который заключает в себе другие элементы. На рисунке 1 предком для всех элементов является `<html>`. В то же время элемент `<body>` является предком для всех содержащихся в нем элементов: `<h1>`, `<p>`, `<span>`, `<nav>` и т.д.

Потомок — элемент, расположенный внутри одного или более типов элементов. Например, `<body>` является потомком `<html>`, а элемент `<p>` является потомком одновременно для `<body>` и `<html>`.

Родительский элемент — элемент, связанный с другими элементами более низкого уровня, и находящийся на дереве выше их. На рисунке 1 `<html>` является родительским только для `<head>` и `<body>`. Элемент `<p>` является родительским только для `<span>`.

Дочерний элемент — элемент, непосредственно подчиненный другому элементу более высокого уровня. На рисунке 1 только элементы `<h1>`, `<h2>`, `<p>` и `<nav>` являются дочерними по отношению к `<body>`.

Соседний элемент(сестринский) — элемент, имеющий общий родительский элемент с рассматриваемым, так называемые элементы одного уровня. На рисунке 1 `<head>` и `<body>` — элементы одного уровня, так же как и элементы `<h1>`, `<h2>` и `<p>` являются между собой соседними.

Структура HTML-документа

Элемент <html>

Является корневым элементом документа. Все остальные элементы содержатся внутри `<html>...</html>`.

Элемент <body>

В разделе `<body>` располагается все содержимое документа.

Элемент <head>

Раздел `<head>...</head>` содержит техническую информацию о странице: заголовок, описание, ключевые слова для поисковых машин, кодировку и т.д. Введенная в нем информация не отображается в окне браузера, однако содержит данные, которые указывают браузеру, как следует обрабатывать страницу.

Элементы раздела <head>

Элемент <meta> Необязательным элементом раздела `<head>` является элемент `<meta>`. С его помощью можно задать описание содержимого страницы и ключевые слова для поисковых машин, автора HTML-документа и прочие свойства метаданных.

Элемент `<head>` может содержать несколько элементов `<meta>`, потому что в зависимости от используемых атрибутов они несут различную информацию.

Указывает кодировку символов.

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

Meta-тег **viewport** сообщает браузеру о том, как именно обрабатывать размеры страницы, и изменять её масштаб.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<meta name="description" content="Описание содержимого страницы">
<meta name="keywords" content="Ключевые слова через запятую">
```

Элементы раздела <head>

Элемент <title>

Обязательным элементом раздела `<head>` является `<title>`. Текст, размещенный внутри элемента `<title>`, отображается в строке заголовка веб-браузера.

Длина заголовка должна быть не более 60 символов, чтобы полностью поместиться в заголовке. Текст заголовка должен содержать максимально полное описание содержимого веб-страницы.

Элемент <style>

Внутри этого элемента задаются стили, которые используются на странице. Для задания стилей в HTML-документе используется язык CSS. Таких элементов на странице может быть несколько. Элемент может содержать код форматирования как самих элементов веб-страницы, так и веб-страницы целиком.

CSS-код можно встраивать непосредственно в элемент разметки в виде значение атрибута `style`, например:

```
<p style="color: #666666; background-color: #ef4444; padding: 20px;">
```

Элементы раздела <head>

Элемент <link>
Задать стили для документа можно также при помощи другого способа — записать их в отдельный файл с расширением `.css`, например, `style.css`. Элемент `<link>` определяет отношение между текущей страницей и другими документами. Таких элементов на странице может быть несколько.

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

`href` — основной атрибут элемента, в качестве значения выступает путь к файлу со стилями;
`type` — определяет MIME-тип документа, на который идет ссылка. В данном случае он принимает значение `text/css`";
`rel` — определяет отношения между текущим документом и документом, на который идет ссылка;
Атрибут `rel` имеет множество значений которые знать наизусть не нужно, и при необходимости можно посмотреть на сайте веб-документации **MDN**.

Те что нам потребуются на первых этапах это:
`icon` — определяет путь к иконке, которая будет использована для текущего документа.
`stylesheet` — ссылка на внешний файл, который будет использоваться в качестве таблицы стилей для данного документа.
`media` — определяет тип устройства, к которым должен быть применен ресурс ссылки.

Элементы раздела <head>

Элемент <script>
Элемент `<script>` позволяет присоединять к документу различные сценарии. Текст сценария может располагаться либо внутри этого элемента, либо во внешнем файле. Если текст сценария расположен во внешнем файле, то он подключается с помощью атрибутов элемента.

Заголовок слайда не совсем корректный, так как для оптимизации скорости загрузки сайта элемент <script> стоит размещать перед закрывающим тегом </body>, но во многих cms они все же размещаются в блоке <head> и шаблонизатор автоматически переносит вниз страницы.

```
<script src="style.css"></script>
```

`src` — указывает на месторасположение файла со сценарием, значение атрибута — это `url` файла, содержащего JavaScript-программу.
`async` — атрибут указывает на то, что сценарий будет выполняться асинхронно с остальной частью страницы (сценарий начнет выполняться одновременно с загрузкой страницы).
`defer` — интерпретация сценариев откладывается до окончания отображения документа на устройстве пользователя.

Элемент <base>
Задает базовый адрес (URL), относительно которого вычисляются все относительные адреса. Это поможет избежать проблем при переносе страницы в другое место, так как все ссылки будут работать, как и прежде.

HTML-теги

HTML-теги — основа языка HTML. Теги используются для разграничения начала и конца элементов в разметке.

Все HTML-элементы делятся на пять типов:

- **пустые элементы** — `<area>`, `<base>`, `<br>`, `<col>`, `<embed>`, `<hr>`, `<img>`, `<input>`, `<link>`, `<menuitem>`, `<meta>`, `<param>`, `<source>`, `<track>`, `<wbr>`;
- **элементы с неформатированным текстом** — `<script>`, `<style>`;
- **элементы, выводящие неформатированный текст** — `<textarea>`, `<title>`;
- **элементы из другого пространства имён** — MathML и SVG;
- **обычные элементы** — все остальные элементы.

Список всех тегов со свойством `display` представлен в таблице: https://html5book.ru/examples/html-tags.html

`<!--...-->` - Используется для добавления комментариев.

Блочные элементы и блочные контейнеры

Блочные элементы — элементы высшего уровня, которые форматируются визуально как блоки, располагаясь на странице в окне браузера вертикально. Значения свойства `display`, такие как `block`, `list-item` и `table` делают элементы блочными. Блочные элементы генерируют основной блок, который содержит только блок элемента. Элементы со значением `display: list-item` генерируют дополнительные блоки для маркеров, которые позиционируются относительно основного блока.

`<address>`, `<article>`, `<aside>`,`<blockquote>`,`<dd>`, `<div>`, `<dl>`, `<dt>`, `<details>`,`<fieldset>`, `<figcaption>`, `<figure>`, `<footer>`, `<form>`,`<h1>-<h6>`, `<header>`, `<hr>`,`<li>`, `<legend>`,`<nav>`, `<noscript>`,`<ol>`, `<output>`, `<optgroup>`, `<option>`,`<p>`, `<pre>`,`<section>`, `<summary>`,`<table>`,`<ul>`

Блочные элементы могут содержать как строчные, так и блочные элементы, но не оба типа элементов сразу. При необходимости, строки текста, принадлежащие блочному контейнеру, могут быть обёрнуты анонимными контейнерами, которые будут вести себя внутри блока как элементы со значением `display: block;`, а строчные элементы обёрнуты элементом `<p>`. Блочные элементы могут содержаться только в пределах блочных элементов.

Элемент `<p>` относится к блочным элементам, но он не должен содержать внутри себя другой элемент `<p>`, а также любой другой блочный элемент.

`<div>` — Тег-контейнер для разделов HTML-документа. Используется для группировки блочных элементов с целью форматирования стилями.

Строчные элементы и строчные контейнеры

Встроенные (строчные) элементы генерируют внутристрочные контейнеры. Они не формируют новые блоки контента. Значения свойства `display`, такие как `inline` и `inline-table` делают элементы строчными.

`<a>`, `<area>`,`<b>`, `<bdo>`, `<bdi>`,`<cite>`, `<code>`,`<dfn>`, `<del>`,`<em>`,`<i>`, `<iframe>`, `<img>`, `<ins>`,`<kbd>`,`<label>`,`<map>`, `<mark>`,`<s>`, `<samp>`, `<small>`, `<span>`, `<strong>`, `<sub>`, `<sup>`,`<time>`,`<q>`,`<ruby>`,`<u>`,`<var>`

Строчные элементы могут содержать только данные и другие строчные элементы. Исключение составляет элемент `<a>`, который согласно спецификации HTML5 может оборачивать целые абзацы, списки, таблицы, заголовки и целые разделы при условии, что они не содержат другие интерактивные элементы — другие ссылки и кнопки.

`<span>` — Контейнер для строчных элементов. Можно использовать для форматирования отрывков текста, например, выделения цветом отдельных слов.

Строчно-блочные элементы

Существует еще одна группа элементов, которые браузер обрабатывает как строчно-блочные `{display: inline-block;}`. Такие элементы являются встроенным, но для них можно задавать поля, отступы, ширину и высоту.

`<audio>`,`<button>`,`<canvas>`,`<embed>`,`<input>`,`<keygen>`,`<meter>`,`<object>`,`<progress>`,`<select>`,`<textarea>`,`<video>`.

HTML-элементы для текста

HTML- текст представлен в спецификации элементами для форматирования и группировки текста. Данные элементы являются контейнерами для текста и не имеют визуального отображения.

Элементы для форматирования текста несут смысловую нагрузку и обычно задают для текста, заключенного внутрь, стилевое оформление, например, выделяют текст жирным начертанием или отображают его шрифтом другого семейства (свойство `font-family`).

Грамотно отформатированный текст дает понять поисковым системам, какие слова несут важную смысловую нагрузку, по каким из них предпочтительно ранжировать веб-страницу в поисковой выдаче.

HTML-элементы для текста

Заголовки являются важными элементами веб-страницы, они упорядочивают текст, формируя его визуальную структуру. Элементы `<h1>...<h6>` должны использоваться только для выделения заголовков нового раздела или подраздела.

При использовании заголовков необходимо учитывать их иерархию, т.е. за `<h1>` должен следовать `<h2>` и т.д. Также не рекомендуется вкладывать в заголовки другие элементы.

Заголовок 1-го уровня

Заголовок 2-го уровня

Заголовок 3-го уровня

Заголовок 4-го уровня

Заголовок 5-го уровня

Заголовок 6-го уровня

HTML-элементы для текста

`<p>` — Параграфы в тексте.

`<br>` — Перенос текста на новую строку.

`<hr>` — Горизонтальная линия для тематического разделения параграфов.

`<b>` — Задает полужирное начертание отрывка текста, не придавая акцент или важность выделенному.

`<strong>` — Расставляет акценты в тексте, выделяя полужирным.

`<i>` — Выделяет отрывок текста курсивом, не придавая ему дополнительный акцент.

`<em>` — Выделяет важные фрагменты текста, отображая их курсивом.

`<small>` — Отображает текст шрифтом меньшего размера.

`<sub>` — Задает подстрочное написание символов, например, индекса элемента в химических формулах.

`<sup>` — Задает надстрочное написание символов.

`<ins>` — Выделяет текст подчеркиванием. Применяется для выделения изменений, вносимых в документ.

`<del>` — Помечает текст как удаленный, перечёркивая его.

`<u>` — Выделяет отрывок текста подчёркиванием, без дополнительного акцента.

HTML-элементы для текста

`<mark>` — Выделяет фрагменты текста, помечая их желтым фоном.

`<abbr>` — Определяет текст как аббревиатуру или акроним. Поясняющий текст задаётся с помощью атрибута `title`.

`<bdo>` — Отображает текст в направлении, указанном в атрибуте `dir`, переопределяя текущее направление написания текста.

`<blockquote>` — Выделяет текст как цитату, применяется для описания больших цитат.

`<q>` — Определяет краткую цитату. Браузерами заключается в кавычки.

`<dfn>` — Определяет слово как термин, выделяя его курсивом. Текст, идущий следом, должен содержать расшифровку этого термина.

`<pre>` — Позволяет вывести текст на экран, сохранив изначальное форматирование. Пробелы и переносы строк при этом не удаляются.

`<code>` — Представляет фрагмент программного кода, отображается шрифтом семейства `monospace`.

HTML-списки

HTML-списки используются для группировки связанных между собой фрагментов информации. Существует три вида списков:

- **маркированный список** — `<ul>` — каждый элемент списка `<li>` отмечается маркером,
- **нумерованный список** — `<ol>` — каждый элемент списка `<li>` отмечается цифрой,
- **список определений** — `<dl>` — состоит из пар термин `<dt>` — `<dd>` определение.

Каждый список представляет собой контейнер, внутри которого располагаются элементы списка или пары термин-определение.

Элементы списка ведут себя как блочные элементы, располагаясь друг под другом и занимая всю ширину блока-контейнера. Каждый элемент списка имеет дополнительный блок, расположенный сбоку, который не участвует в компоновке.

Маркированный список представляет собой неупорядоченный список *(от англ. Unordered List)*. Создаётся с помощью элемента `<ul>`. В качестве маркера элемента списка выступает метка, например, закрашенный кружок.

Каждый элемент списка создаётся с помощью элемента `<li>` *(от англ. List Item)*.

HTML-списки

Нумерованный список создаётся с помощью элемента `<ol>`. Каждый пункт списка также создаётся с помощью элемента `<li>`. Браузер нумерует элементы по порядку автоматически и если удалить один или несколько элементов такого списка, то остальные номера будут автоматически пересчитаны.

Для элемента `<li>` доступен атрибут `value`, который позволяет изменить номер по умолчанию для выбранного элемента списка. Например, если для первого пункта списка задать `<li value="10">`, то остальная нумерация будет пересчитана относительно нового значения.

Для элемента `<ol>` доступны следующие атрибуты:

`reversed` — задает отображение списка в обратном порядке (например, 9, 8, 7…).
`start` — задает начальное значение, от которого пойдет отсчет нумерации, например, конструкция `<ol start="10">` первому пункту присвоит порядковый номер «10». Также можно одновременно задавать тип нумерации, например, `<ol type="I" start="10">`.
`type`
`type` — задает вид маркера для использования в списке (в виде букв или цифр). Принимаемые значения:
`1` — значение по умолчанию, десятичная нумерация.
`A` — нумерация списка в алфавитном порядке, заглавные буквы (A, B, C, D).
`a` — нумерация списка в алфавитном порядке, строчные буквы (a, b, c, d).
`I` — нумерация римскими заглавными цифрами (I, II, III, IV).
`i` — нумерация римскими строчными цифрами (i, ii, iii, iv).

HTML-списки

Списки определений создаются с помощью элемента `<dl>`. Для добавления термина применяется элемент `<dt>`, а для вставки определения — элемент `<dd>`.

Многоуровневый список используется для отображения элементов списка на разных уровнях с различными отступами.

Чтобы сделать вложенную нумерацию, нужно использовать следующие свойства:

- `counter-reset` сбрасывает один или несколько счётчиков, задавая значение для сброса;
- `counter-increment` задаёт значение приращения счётчика, т.е. с каким шагом будет нумероваться каждый последующий пункт;
- `content` — генерируемое содержимое, в данном случае отвечает за вывод номера перед каждым пунктом списка.

HTML-таблицы

HTML-таблицы упорядочивают и выводят на экран данные с помощью строк или столбцов. Таблицы состоят из ячеек, образующихся при пересечении строк и столбцов.

Ячейки таблиц могут содержать любые HTML-элементы, такие как заголовки, списки, текст, изображения, элементы форм, а также другие таблицы. Каждой таблице можно добавить связанный с ней заголовок, расположив его перед таблицей или после неё.

Таблицы больше не используются для вёрстки веб-страниц и компоновки отдельных элементов, потому что такой приём не обеспечив

Таблица создаётся при помощи элемента `<table></table>`, который является контейнером для элементов таблицы и все элементы должны находиться внутри него.

`<table>` — Тег для создания таблицы.
`<thead>` — Определяет заголовок таблицы.
`<tbody>` — Определяет тело таблицы.
`<caption>` — Добавляет подпись к таблице. Вставляется сразу после тега `<table>`.
`<tr>` — Создает строку таблицы.
`<th>` — Создает заголовок ячейки таблицы.
`<td>` — Создает ячейку таблицы.
`<col>` — Выбирает для форматирования один или несколько столбцов таблицы, не содержащих информацию одного типа.
`<colgroup>` — Создает структурную группу столбцов, выделяющую множество логически однородных ячеек.
`<tfoot>` — Определяет нижний колонтитул таблицы.

HTML-таблицы

Группирование строк и столбцов таблицы

Элемент `<colgroup>` создает структурную группу столбцов, выделяя логически однородные ячейки. Группирует один или более столбцов для единого форматирования, позволяя применить стили к столбцам вместо того, чтобы повторять стили для каждой ячейки и для каждой строки.

Добавляется непосредственно после тегов `<table>` и/или `<caption>`.

Элемент `<col>` формирует группы столбцов, которые делят таблицу на разделы, не относящиеся к общей структуре, т.е. не содержащие информацию одного типа. Позволяет задавать свойства столбцов для каждого столбца в пределах элемента `<colgroup>`.

С помощью атрибута `style` можно изменить основной цвет фона ячеек. Для элемента `<col>` доступен атрибут `span`, задающий количество столбцов для объединения.

```
<colgroup>
    <col span="2" style="background:Khaki"><!-- Задаем цвет фона для первых двух столбцов таблицы-->
    <col style="background-color:LightCyan"><!-- Задаем цвет фона для следующего (одного) столбца таблицы-->
</colgroup>
```

Атрибуты `colspan` и `rowspan` объединяют ячейки таблицы. Атрибут `colspan` задает количество ячеек, объединенных по горизонтали, а `rowspan` — по вертикали.

HTML-ссылки

HTML-ссылки создаются с помощью элементов `<a>`, `<area>` и `<link>`. Ссылки представляют собой связь между двумя ресурсами, одним из которых является текущий документ.

Ссылки можно поделить на две категории:

- **ссылки на внешние ресурсы** — создаются с помощью элемента `<link>` и используются для расширения возможностей текущего документа при обработке браузером;
- **гиперссылки** — ссылки на другие ресурсы, которые пользователь может посетить или загрузить.

Гиперссылки создаются с помощью элемента `<a></a>`. Внутрь помещается текст, который будет отображаться на веб-странице. Текст ссылки отображается в браузере с подчёркиванием, цвет шрифта — синий, при наведении на ссылку курсор мыши меняет вид.

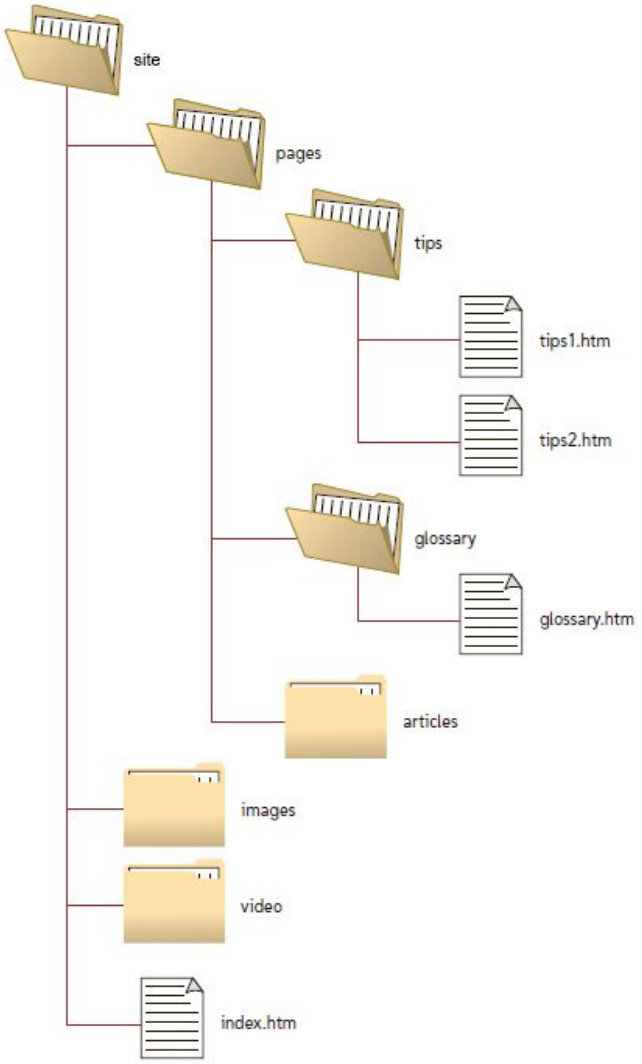
Обязательным параметром элемента `<a>` является атрибут `href`, который задает URl-адрес веб-страницы.

```
<a href="http://site.ru">указатель ссылки</a>
```

HTML-ссылки

Ссылка состоит из двух частей — **указателя** и **адресной части**. **Указатель ссылки** представляет собой фрагмент текста или изображение, видимые для пользователя. **Адресная часть** ссылки пользователю не видна, она представляет собой адрес ресурса, к которому необходимо перейти.

Когда в ссылке указывается только имя файла, браузер предполагает, что файл находится в той же папке, что и документ, содержащий гиперссылку. На практике веб-сайты содержат сотни документов, которые размещают в отдельные папки, чтобы ими было легче управлять. Чтобы создать ссылку на файл, находящийся вне папки, содержащей текущий документ, необходимо указать расположение файла или путь. HTML поддерживает два вида пути: абсолютный и относительный.



HTML-ссылки

Абсолютный путь указывает точное местоположение файла в пределах всей структуры папок на компьютере (сервере). Абсолютный путь к файлу даёт доступ к файлу со сторонних ресурсов и содержит следующие компоненты:

- протокол, например, http (опционально);
- домен (доменное имя или IP-адрес компьютера);
- папка (имя папки, указывающей путь к файлу);
- файл (имя файла).

Относительный путь описывает путь к указанному документу относительно текущего. Путь определяется с учётом местоположения веб-страницы, на которой находится ссылка. Относительные ссылки используются при создании ссылок на другие документы на одном и том же сайте. Когда браузер не находит в ссылке протокол `http://`, он выполняет поиск указанного документа на том же сервере.

Относительный путь содержит следующие компоненты:

- папка (имя папки, указывающей путь к файлу);
- файл (имя файла).

Путь для относительных ссылок имеет три специальных обозначения:

- `/` указывает на корневую директорию и говорит о том, что нужно начать путь от корневого каталога документов и идти вниз до следующей папки
- `./` указывает на текущую папку
- `../` подняться на одну папку (директорию) выше

HTML-ссылки

Якоря, или внутренние ссылки, создают переходы на различные разделы текущей веб-страницы, позволяя быстро перемещаться между разделами. Это оказывается очень удобным в случае, когда на странице слишком много текста. Внутренние ссылки также создаются при помощи элемента `<a>` с разницей в том, что атрибут `href` содержит имя указателя — так называемый **якорь**, а не URl-адрес. Перед именем указателя всегда ставится знак `#`.

Следующая разметка создаст оглавление с быстрыми переходами на соответствующие разделы:

```
<a href="#p1">Лето</a> <!--создаём якорь, указав #id элемента-->
<p id="p1">...</p> <!--добавляем соответствующий id элементу→
```

Если нужно сделать ссылку **с одной страницы** сайта на определенный раздел **другой страницы**, то необходимо задать `id` для этого раздела страницы, а затем добавить его к абсолютному адресу ссылки:

```
<a href="https://html5book.ru/css-shrifty/#about-color" class="site" target="_blank">color</a>
```

HTML-ссылки

Как сделать изображение-ссылку

Чтобы сделать кликабельное изображение, необходимо поместить элемент `<img>` внутрь элемента `<a>`. Чтобы ссылка открывалась в другом окне, нужно добавить атрибут `target="_blank"` для ссылки.

```
<a href="http://www.fast-torrent.ru/film/gran-za-granyu-tv.html" target="_blank">
      <img src="//html5book.ru/wp-content/uploads/2014/07/Fringe.jpg" alt="Fringe">
</a>
```

У ссылок появились новые возможности — по клику можно не только переходить на другие страницы и скачивать файлы, но и совершать звонки на телефоны, отправлять сообщения или звонить по скайпу.

```
ссылка на телефонный номер
<a href="tel:+74951234567">+7 (495) 123-45-67</a>
ссылка на адрес электронной почты
<a href="mailto:example@mail.ru">example@mail.ru</a>
ссылка на скайп (позвонить)
<a href="skype:имя-пользователя?call">Skype</a>
```

HTML-ссылки

Элемент `<a>` поддерживает глобальные атрибуты и собственные.

`href` — Значением атрибута является URL-адрес документа, на который указывается ссылка.

`download` — Дополняет атрибут `href` и сообщает браузеру, что ресурс должен быть загружен в момент, когда пользователь щелкает по ссылке, вместо того, чтобы, например, предварительно открыть его (как PDF-файл). Задавая имя для атрибута, мы таким образом задаем имя загружаемому объекту. Разрешается использовать атрибут без указания его значения.
`target` — Указывает на то, в каком окне должен открываться документ, к которому ведет ссылка. Принимает следующие значения:

- `_self` — страница загружается в текущее окно;
- `_blank` — страница открывается в новом окне браузера;

`rel` — Дополняет атрибут `href` информацией об отношении между текущим и связанным документом. Принимаемые разные значения.

HTML-изображения

HTML-изображения добавляются на веб-страницы с помощью элемента `<img>`. Использование графики делает веб-страницы визуально привлекательнее. Изображения помогают лучше передать суть и содержание веб-документа.

Элементы `<map>` и `<area>` позволяют создавать **карты-изображения** с активными областями.

Элемент `<img>` представляет изображение и его резервный контент, который добавляется с помощью атрибута `alt`. Так как элемент `<img>` является строчным, то рекомендуется располагать его внутри блочного элемента, например, `<p>` или `<div>`.

Элемент `<img>` имеет обязательный атрибут `src`, значением которого является абсолютный или относительный путь к изображению:

```
<img src="image.png" alt="Пример кода">
```

HTML-изображения

Для элемента `<img>` доступны следующие атрибуты:

`src` — задает путь к изображению. Синтаксис: `src="flower.jpg"`.
`alt` — добавляет альтернативный текст для изображения. Выводится на месте появления изображения до его загрузки или при отключенной графике, а также выводится всплывающей подсказкой при наведении курсора мыши на изображение. Синтаксис: `alt="описание изображения"`.
`height` — задает высоту изображения в `px`. Синтаксис: `height="300"`.
`width` — адает ширину изображения в `px`. Синтаксис: `width="500"`.
`usemap` — Атрибут `usemap` определяет изображение в качестве карты-изображения. Значение обязательно должно начинаться с символа `#`. Значение ассоциируется со значением атрибута `name` или `id` элемента `<map>` и создает связь между элементами `<img>` и `<map>`. Атрибут нельзя использовать, если элемент `<map>` является потомком элемента `<a>` или `<button>`.
Синтаксис: `usemap="#mymap"`.
`ismap` — указывает на то, что картинка является частью изображения-карты, расположенного на сервере (изображение-карта — изображение с интерактивными областями). При нажатии на изображение-карту координаты передаются на сервер в виде строки запроса URL-адреса. Атрибут `ismap` допускается только в случае, если элемент `<img>` является потомком элемента `<a>` с действительным атрибутом `href`.
Синтаксис: `ismap`.
`srcset` — Создаёт список источников для изображения, которые будут выбраны, исходя из разрешения экрана. Может использоваться вместе или вместо атрибута `src`. Значением атрибута является одна или несколько строк, разделенных запятой.
`sizes` — Задаёт размер изображения в зависимости от параметров отображения. Работает только при заданном атрибуте `srcset`. Значением атрибута является одна или несколько строк, указанных через запятую.

HTML-изображения

Без задания размеров изображения рисунок отображается на странице в реальном размере. Отредактировать размеры изображения можно с помощью атрибутов `width` и `height`. Если будет задан только один из атрибутов, то второй будет вычисляться автоматически для сохранения пропорций рисунка.

Адрес изображения может быть указан полностью (абсолютный URL), или же через относительный путь от **документа** или **корневого каталога** сайта:

- `url(../images/anyphoto.png)` — относительный путь от документа,
- `url(/images/anyphoto.png)` — относительный путь от корневого каталога.

Это интерпретируется следующим образом:

- `../` — означает подняться вверх на один уровень, к корневому каталогу,
- `images/` — перейти к папке с изображениями,
- `anyphoto.png` — указывает на файл изображения.

HTML-изображения

Форматы графических файлов

- Формат **JPEG**. Изображения JPEG идеальны для фотографий, они могут содержать миллионы различных цветов.
- Формат **GIF**. Идеален для сжатия изображений, в которых есть области со сплошным цветом, например, логотипов. GIF-файлы позволяют установить один из цветов прозрачным, благодаря чему фон веб-страницы может проявляться через часть изображения. Также GIF-файлы могут включать в себя простую анимацию. GIF-изображения содержат всего лишь 256 оттенков, из-за чего изображения выглядят пятнистыми и нереалистичного цвета, как плакаты.
- Формат **PNG**. Включает в себя лучшие черты GIF- и JPEG-форматов. Содержит 256 цветов и дает возможность сделать один из цветов прозрачным, при этом сжимает изображения в меньший размер, чем GIF-файл.
- **SVG**. SVG-рисунок состоит из набора геометрических фигур, описанных в формате XML: линия, эллипс, многоугольник и т.п. Поддерживается как статичная, так и анимированная графика.Изображения в формате SVG могут изменяться в размере без снижения качества.
- Формат **ICO**. Используется как иконка на сайтах в интернете. Именно картинка такого формата отображается рядом с адресом сайта или закладкой в браузере. Размер значка может быть любым, но наиболее употребимы квадратные значки со сторонами 16, 32 и 48 пикселей.

HTML-изображения

Элемент `<map>` служит для представления графического изображения в виде карты с активными областями. Активные области определяются по изменению вида курсора мыши при наведении. Щелкая мышью на активных областях, пользователь может переходить к связанным документам.

Для элемента доступен атрибут `name`, который задает имя карты. Значение атрибут `name` для элемента `<map>` должно соответствовать имени в атрибуте `usemap` элемента `<img>`:

```
<img src="url" usemap="#имя_карты">
<map name="имя_карты">
      ...
</map>
```

Элемент `<map>` содержит ряд элементов `<area>`, определяющих интерактивные области в изображении карты.

HTML-изображения

Элемент `<area>` описывает только одну активную область в составе карты изображений на стороне клиента. Если одна активная область перекрывает другую, то будет реализована первая ссылка из списка областей.

```
<map name="имя карты">
      <area атрибуты>
</map>
```

Имеет большинство атрибутов ссылок таких как: `alt`, `download`, `href`, `rel`, `target`, и дополнительно:

`shape` — Задает форму активной области на карте и ее координаты. Может принимать следующие значения:

- `rect` — активная область прямоугольной формы;
- `circle` — активная область в форме круга;
- `poly` — активная область в форме многоугольника;
- `default` — активная область занимает всю площадь изображения.

`coords` — Определяет позицию области на экране. Координаты задаются в единицах длины и разделяются запятыми:

- для **круга** — координаты центра и радиус круга;
- для **прямоугольника** — координаты верхнего левого и правого нижнего углов;
- для **многоугольника** — координаты вершин многоугольника в нужном порядке, также рекомендуется указывать последние координаты, равные первым, для логического завершения фигуры.

HTML-изображения

`<figure>` — Самодостаточный тег-контейнер для такого контента как иллюстрации, диаграммы, фотографии, примеры кода, обычно с подписью.

`<figcaption>` — Заголовок/подпись для элемента `<figure>`.

```
<figure>
    <img src="picture.jpg" alt="Осень">
    <figcaption>Осенний лес</figcaption>
</figure>
```

Основные семантические теги HTML

Семантическая вёрстка — подход к разметке, который опирается не на содержание сайта, а на смысловое предназначение каждого блока и логическую структуру документа.

Наличие семантической разметки страниц помогает поисковым ботам лучше понимать, что находится на странице, и в зависимости от этого ранжировать сайты в поисковой выдаче.

`<article>` — Раздел контента, который образует независимую часть документа или сайта, например, статья в журнале, запись в блоге, комментарий.

`<section>` — Определяет логическую область (раздел) страницы, обычно с заголовком.

`<aside>` — Представляет контент страницы, который имеет косвенное отношение к основному контенту страницы/сайта.

`<nav>` — Раздел документа, содержащий навигационные ссылки по сайту.

`<header>` — Секция для вводной информации сайта или группы навигационных ссылок. Может содержать один или несколько заголовков, логотип, информацию об авторе.

`<main>` — Контейнер для основного уникального содержимого документа. На одной странице должно быть не более одного элемента `<main>`.

`<footer>` — Определяет завершающую область (нижний колонтитул) документа или раздела.

Как разметить страницу с точки зрения семантики

Процесс разметки можно разделить на несколько шагов с разной степенью детализации.

1. Крупные смысловые блоки на каждой странице сайта. Теги: `<header>, <main>, <footer>`.
2. Крупные смысловые разделы в блоках. Теги: `<nav>, <section>, <article>, <aside>`.
3. Заголовок всего документа и заголовки смысловых разделов. Теги: `<h1>-<h6>`.
4. Мелкие элементы в смысловых разделах. Списки, таблицы, демо-материалы, параграфы и переносы, формы, цитаты, контактная информация и прогресс.
5. Фразовые элементы. Изображения, ссылки, кнопки, видео, время и мелкие текстовые элементы.

Есть простые правила для выбора нужных тегов.

- Получилось найти самый подходящий смысловой тег — использовать его.
- Для потоковых контейнеров — `<div>`.
- Для мелких фразовых элементов (слово или фраза) — `<span>`.

Правило для определения `<article>, <section>` и `<div>`: Можете дать имя разделу и вынести этот раздел на другой сайт? — `<article>`

1. Можете дать имя разделу, но вынести на другой сайт не можете? — `<section>`
2. Не можете дать имя? Получается что-то наподобие «новости и фотогалерея» или «правая колонка»? — `<div>`

HTML-атрибуты

HTML-атрибуты сообщают браузеру, каким образом должен отображаться тот или иной элемент страницы. Атрибуты позволяют сделать более разнообразными внешний вид информации, добавляемой с помощью **одинаковых** тегов.

Значение атрибута заключается в кавычки `""`. Названия и значения атрибутов не чувствительны к регистру, но, тем не менее, рекомендуется набирать их в нижнем регистре.

Глобальные атрибуты, приведенные в таблице, могут быть использованы для любого HTML-элемента, хотя некоторые из них могут не оказывать на элементы никакого влияния.

`class` — Определяет имя класса для элемента (используется для определения класса в таблице стилей).

Принимаемые значения: имя класса.

`id` — Определяет уникальный идентификатор элемента.Принимаемые значения: `id` — идентификатор элемента.

`style` — Указывает на код CSS, применяемую для оформления элемента. Принимаемые значения: код CSS.

`title` — Определяет дополнительную информацию об элементе, задавая всплывающую подсказку для страницы.

Принимаемые значения: текст.

`tabindex` — Определяет порядок перехода к элементу при помощи клавиши TAB. Принимаемые значения: порядковый номер.

`dir` — Определяет направление текста контента в элементах `<bdo>` и `<bdi>`. Принимаемые значения: `ltr/rtl/auto`.

Спецсимволы HTML

Спецсимволы HTML, или символы-мнемоники, представляют собой конструкцию SGML (англ. **Standard Generalized Markup Language** — стандартный обобщённый язык разметки), ссылающуюся на определенные символы из символьного набора документа. В основном они используются для указания символов, которых нет в стандартной компьютерной клавиатуре, либо которые не поддерживает кодировка HTML-страницы (Windows-1251, UTF-8 и т.д.).

Чтобы разместить символ на веб-странице, необходимо указать HTML-код или мнемонику.

Спецсимволы чувствительны к регистру, поэтому их необходимо прописывать точно так, как указано в таблице. Спецсимволы, не имеющие мнемоники, могут не отображаться вовсе или же некорректно отображаться в тех или иных браузерах.

Для вставки символов внутрь тегов воспользуйтесь HTML-кодом символа, а для использования символов в таблицах стилей, например, в качестве значения свойства `content` — CSS-код.

Спецсимвол наследует **цвет** от цвета текста родительского элемента. Чтобы изменить цвет спецсимвола, можно поместить HTML-код внутрь тега `<span>` или задать нужное значение свойства `color` (при вставке спецсимволов через свойство `content`).

HTML-аудио

HTML5-элемент `<audio>` используется для внедрения звукового контента в веб-страницы. В настоящий момент не существует аудио формата, который бы работал во всех браузерах, поэтому для обеспечения доступности контента максимально широкой аудитории рекомендуется включать несколько источников звука, представленных с использованием атрибута `src` элемента `<source>`.

```
<audio controls>
  <source src="name.ogg" type="audio/ogg">
  <source src="name.mp3" type="audio/mpeg">
</audio>
```

`controls` — Указывает браузеру, что нужно отобразить базовые элементы управления воспроизведением (начинать и останавливать воспроизведение, переходить в другое место записи, регулировать громкость).
`autoplay` — Автоматическое воспроизведение аудио файла сразу же после загрузки страницы.
`loop` — Циклическое воспроизведение аудио файла.
`muted` — Выключает звук при воспроизведении аудио файла.
`src` — Содержит абсолютный или относительный URL-адрес аудио файла.
`preload` — Атрибут, отвечающий за предварительную загрузку аудио контента. Не является обязательным, некоторые браузеры игнорируют его. Возможные значения:
`auto` — браузер загружает аудио файл полностью, чтобы он был доступен, когда пользователь начнет его воспроизведение.
`metadata` — браузер загружает первую небольшую часть аудио файла, чтобы определить его основные характеристики.
`none` — отсутствие автоматической загрузки аудио файла.

HTML-видео

HTML5-видео — новый стандарт для размещения мультимедийных файлов в сети с оригинальным программным интерфейсом без привлечения подключаемых модулей.

```
<video controls width="400" height="300">
  <source src="video.mp4" type="video/mp4">
  <source src="video.webm" type="video/webm">
  <source src="video.ogv" type="video/ogg">
</video>
```

Имеет кроме тех же атрибутов что и у `<audio>` дополнительно:

`height` — Задает высоту окна для отображения видеоданных, возможные значения: `px` или `%`
`width` — Задает ширину окна для отображения видеоданных, возможные значения: `px` или `%`
`poster` — URL файла изображения, которое будет отображаться во время загрузки видеофайла или до тех пор, пока пользователь не нажмет на кнопку PLAY. Если атрибут не задан, то будет отображаться первый кадр видеофайла.

HTML-формы

HTML-формы являются элементами управления, которые применяются для сбора информации от посетителей веб-сайта. Веб-формы состоят из набора текстовых полей, кнопок, списков и других элементов управления, которые активизируются щелчком мыши. Технически формы передают данные от пользователя удаленному серверу. Для получения и обработки данных форм используются языки веб-программирования, такие как **PHP**, **Perl**.

`<form>` — Форма для сбора и отправки на сервер информации от пользователей. Не работает без атрибута `action`.
`<fieldset>` — Группирует связанные элементы в форме, рисуя рамку вокруг них.
`<legend>` — Заголовок элементов формы, сгруппированных с помощью элемента `<fieldset>`.
`<input>` — Создает многофункциональные поля формы, в которые пользователь может вводить данные.
`<label>` — Добавляет текстовую метку для элемента `<input>`.
`<textarea>` — Создает большие поля для ввода текста.
`<select>` — Элемент управления, позволяющий выбирать значения из предложенного множества. Варианты значений помещаются в `<option>`.
`<datalist>` — Элемент-контейнер для выпадающего списка элемента `<input>`. Варианты значений помещаются в элементы `<option>`.
`<option>` — Определяет вариант/опцию для выбора в раскрывающемся списке `<select>`, `<optgroup>` или `<datalist>`.
`<button>` — Создает интерактивную кнопку. Внутрь тега можно поместить содержимое — текст или изображение.

HTML-формы

Основу любой формы составляет элемент `<form>...</form>`. Он не предусматривает ввод данных, так как является контейнером, удерживая вместе все элементы управления формы – **поля**. Атрибуты этого элемента содержат информацию, общую для всех полей формы, поэтому в одну форму нужно включать поля, объединенные логически.

`action` — **Обязательный атрибут**, который указывает `url` обработчика формы на сервере, которому передаются данные. Представляет из себя файл (например, `action.php`), в котором описано, что нужно делать с данными формы. Если значение атрибута не будет указано, то после перезагрузки страницы элементы формы примут значения по умолчанию.
В случае, если вся работа будет выполняться на стороне клиента сценариями JavaScript, то для атрибута `action` можно указать значение `#`.

`name` — Задает **имя формы**, которое будет использоваться для доступа к элементам формы через сценарии, например, `name="opros"`.

`method` — Задает способ передачи данных формы.
Метод `get` передает данные на сервер через адресную строку браузера.
Метод `post` применяется для пересылки данных больших объемов, а также конфиденциальной информации и паролей. Данные, отправляемые с помощью этого метода, не видны в заголовке URL, так как они содержатся в теле сообщения.

HTML-формы

Группировка элементов формы

Элемент `<fieldset>...</fieldset>`предназначен для группировки элементов, связанных друг с другом, разделяя таким образом форму на логические фрагменты.

Каждой группе элементов можно присвоить название с помощью элемента `<legend>`, который идет сразу за открывающим тегом элемента `<fieldset>`. Название группы проявляется слева в верхней границе `<fieldset>`. Например, если в элементе `<fieldset>` хранится контактная информация.

`disabled` — Если атрибут присутствует, то группа связанных элементов формы, находящихся внутри контейнера `<fieldset>`, отключены для заполнения и редактирования.
`form` — Значение атрибута должно быть равно атрибуту `id` элемента `<form>` в этом же документе. Указывает на одну или несколько форм, к которым принадлежит данная группа элементов. На данный момент атрибут не поддерживается ни одним браузером.

`name` — Определяет **имя**, которое будет использоваться для ссылки на элементы в JavaScript, или для ссылки на данные формы после заполнения и отправки формы. Является аналогом атрибута `id`.

HTML-формы

Создание полей формы

Элемент `<input>` создает большинство полей формы. Атрибуты элемента отличаются в зависимости от типа поля, для создания которого используется этот элемент.

`button` — создает кнопку.
`checkbox` — превращает поле ввода во флажок, который можно установить или очистить.
`color` — генерирует палитры цветов в поддерживающих браузерах, давая пользователям возможность выбирать значения цветов в шестнадцатеричном формате.
`date` — позволяет вводить дату в формате дд.мм.гггг.
`datetime-local` — позволяет вводить дату и время, разделенные прописной английской буквой `T` по шаблону дд.мм.гггг чч:мм.
`email` — браузеры, поддерживающие данный атрибут, будут ожидать, что пользователь введет данные, соответствующие синтаксису адресов электронной почты.
`file` — позволяет загружать файлы с компьютера пользователя.
`hidden` — скрывает элемент управления, который не отображается браузером и не дает пользователю изменять значения по умолчанию.
`month` — позволяет пользователю вводить год и номер месяца по шаблону гггг-мм.
`number` — предназначено для ввода целочисленных значений. Его атрибуты `min`, `max` и `step` задают верхний, нижний пределы и шаг между значениями соответственно. Эти атрибуты предполагаются у всех элементов, имеющих численные показатели. Их значения по умолчанию зависят от типа элемента.

HTML-формы

Значения атрибута `type`

`password` — создает текстовые поля в форме, при этом вводимые пользователем символы заменяются на звездочки, маркеры, либо другие, установленные браузером значки.
`radio` — создает переключатель — элемент управления в виде небольшого кружка, который можно включить или выключить.
`range` — позволит создать такой элемент интерфейса, как ползунок, min / max — позволят установить диапазон выбора.
`reset` — создает кнопку, которая очищает поля формы от введенных пользователем данных.
`search` — обозначает поле поиска, по умолчанию поле ввода имеет прямоугольную форму.
`submit` — создает стандартную кнопку, активизируемую щелчком мыши. Кнопка собирает информацию с формы и отправляет ее для обработки.
`text` — создает текстовые поля в форме, выводя однострочное текстовое поле для ввода текста.
`time` — позволяет вводить время в 24-часовом формате по шаблону чч:мм. В поддерживающих браузерах оно отображается как элемент управления в виде числового поля ввода со значением, изменяемым с помощью мыши, и допускает ввод только значений времени.
`url` — поле предназначено для указания URL-адресов.
`week` — соответствующий инструмент-указатель позволяет пользователю выбрать одну неделю в году, после чего обеспечит ввод данных в формате нн-гггг. В зависимости от года число недель может быть 52 или 53.

HTML-формы

`autofocus` — Позволяет сделать так, чтобы в загружаемой форме то или иное поле ввода уже имело фокус (было выбрано), являясь готовым к вводу значения.
`disabled` — Отключает возможность редактирования и копирования содержимого поля.
`value` — Определяет текст, отображаемый на кнопке, в поле или связанный текст. Не указывается для полей типа file.
`size` — Задает видимую ширину поля в символах. Значение по умолчанию — 20. Работает со следующими типами полей: `text`, `search`, `tel`, `url`, `email` и `password`.
`placeholder` — Содержит текст, который отображается в поле ввода до заполнения (чаще всего это подсказка).
`readonly` — Не позволяет пользователю изменять значения элементов формы, выделение и копирование текста при этом доступно. Указывается без значения атрибута.
`pattern` — Позволяет определять с помощью **регулярного выражения** синтаксис данных, ввод которых должен быть разрешен в определенном поле. Например, `pattern="[a-z]{3}-[0-9]{3}"` — квадратные скобки устанавливают диапазон допустимых символов, в данном случае — любые строчные буквы, число в фигурных скобках указывает, что нужны три строчные буквы, после которых следует тире, далее — три цифры в диапазоне от 0 до 9.
`maxlength` — Атрибут задает максимальное количество символов, вводимых в поле. Значение по умолчанию 524288 символов.

HTML-формы

`accept` — Определяет тип файла, разрешенных для отправки на сервер. Указывается только для `<input type="file">`.
Возможные значения:
file_extension — разрешает загрузку файлов с указанным расширением, например, `accept=".gif"`, `accept=".pdf"`, `accept=".doc"`
`audio/*` — разрешает загрузку аудиофайлов
`video/*` — разрешает загрузку видеофайлов
`image/*` — разрешает загрузку изображений

`required` — Выводит сообщение о том, что данное поле является обязательным для заполнения. Если пользователь попытается отправить форму, не введя в это поле требуемое значение, то на экране отобразится предупреждающее сообщение. Указывается без значения атрибута.
`list` — Является ссылкой на элемент `<datalist>`, содержит его `id`.Позволяет предоставить пользователю несколько вариантов на выбор, когда он начинает вводить значение в соответствующем поле.

HTML-формы

`max` — Позволяет ограничить допустимый ввод числовых данных максимальным значением, значение атрибута может содержать целое или дробное число. Рекомендуется использовать этот атрибут вместе с атрибутом `min`. Работает со следующими типами полей: `number`, `range`, `date`, `datetime`, `datetime-local`, `month`, `time` и `week`.

`min` — Позволяет ограничить допустимый ввод числовых данных минимальным значением.

`step` — Используется для элементов, предполагающих ввод числовых значений, указывает величину увеличения или уменьшения значений в процессе регулировки диапазона (шаг).

`multiple` — Позволяет пользователю ввести несколько значений атрибутов, разделяя их запятой. Применяется в отношении файлов и адресов электронной почты. Указывается без значения атрибута.

`width` — Значение атрибута содержит количество пикселей. Позволяет задать ширину полей формы.

`height` — Значение атрибута содержит количество пикселей без указания единицы измерения. Устанавливает высоту поля формы типа `type="image"`, например, `<input type="image" src="img_submit.gif" height="50">`. Рекомендуется одновременно устанавливать как высоту, так и ширину поля.

`name` — Определяет имя, которое будет использоваться для доступа к элементу `<form>`

HTML-формы

Текстовые поля ввода

Элемент `<textarea>...</textarea>`используется вместо элемента `<input type="text">`, когда нужно создать большие текстовые поля. Текст, отображаемый как исходное значение, помещается внутрь.

Размеры поля устанавливаются при помощи атрибутов `cols` - размеры по горизонтали, `rows` - размеры по вертикали. Высоту поля можно задать свойством `height`. Все размеры считаются исходя из размера одного символа моноширинного шрифта.

Атрибуты те же что и для `<input>: autofocus, disabled, maxlength, name, placeholder, readonly, required.`

`cols` — Устанавливает ширину через количество символов. Если пользователь вводит больше текста, появляется полоса прокрутки.
`rows` — Указывает число, которое означает, сколько строк должно отображаться в текстовой области.
`wrap` — Определяет, должен ли текст сохранять переносы строк при отправке формы. Значение `hard` сохраняет перенос, а значение `soft` не сохраняет. Если используется значение `hard`, то должно указываться значение атрибута `cols`.

HTML-формы

Раскрывающийся список

Списки дают возможность расположить большое количество пунктов компактно. Раскрывающиеся списки создаются при помощи элемента `<select>...</select>`. Они позволяют выбрать одно или несколько значений из предложенного множества. По умолчанию в поле списка отображается его первый элемент.

Для добавления в список пунктов используются элементы `<option>...</option>`, которые располагаются внутри `<select>`.

Для систематизации списков применяется элемент `<optgroup>...</optgroup>`, который создает заголовки в списках.

`multiple` — Дает возможность выбора одного или нескольких пунктов, для этого при выборе нужно нажать и удерживать нажатой клавишу `Ctrl`.
`size` — Задает количество одновременно видимых на экране элементов списка. Если количество элементов списка превышает установленное количество, появляется полоса прокрутки. Значение атрибута задается целым положительным числом.
Также доступны: `autofocus, disabled, name, required`

HTML-формы

АТРИБУТЫ ЭЛЕМЕНТА <OPTION>

`disabled` — Делает недоступным для выбора элемент списка.
`label` — Задает укороченную версию для элемента, которая будет отражаться в выпадающем списке. Значение атрибута содержит текст, описывающий соответствующий пункт выпадающего списка.
`selected` — Отображает выбранный элемент списка по умолчанию при загрузке страницы браузером.
`value` — Указывает значение, которое будет отправлено на сервер при отправке формы.

АТРИБУТЫ ЭЛЕМЕНТА <OPTGROUP>

`disabled` — Отключает данную группу элементов списка для выбора.
`label` — Задает заголовок для группы элементов выпадающего списка. Значение атрибута содержит текст, недоступный для выбора, который будет располагаться над соответствующими пунктами списка. Текст выделяется в браузере жирным начертанием.

HTML-формы

Надписи к полям формы

Надписи к элементам формы создаются с помощью элемента `<label>...</label>`. Существует два способа группировки надписи и поля. Если поле находится внутри элемента `<label>`, то атрибут `for` указывать не нужно.

```
<!-- с указанием атрибута for -->
<label for="comments">Когда вы последний раз летали на самолете?/label>
<textarea id="comments"></textarea>

<!-- без атрибута for -->
<p><label>Кошка<input id="cat" type="checkbox"></label></p>
```

`for` — Определяет, к какому полю формы привязан данный элемент. Можно создавать поясняющие надписи к следующим элементам формы: `<input>`, `<textarea>`, `<select>`. Значение атрибута содержит идентификатор поля формы.

HTML-формы

Кнопки

Элемент `<button>...</button>` создает кликабельные кнопки. В отличие от кнопок, созданных `<input>` (`<input type="submit"></input>`, `<input type="image">`, `<input type="reset">`, `<input type="button">`), внутрь элемента `<button>` можно поместить контент — текст или изображение.

Для корректного отображения элемента `<button>` разными браузерами нужно указывать атрибут `type`, например, `<button type="submit"></button>`.

Кнопки позволяют пользователям передавать данные в форму, очищать содержимое формы или предпринимать какие-либо другие действия. Можно создавать границы, изменять фон и выравнивать текст на кнопке.

`type` — Определяет тип кнопки. Возможные значения:

`button` — кликабельная кнопка

`reset` — кнопка сброса, возвращает первоначальное значение

`submit` — кнопка для отправки данных формы.

HTML-формы

Флажки и переключатели в формах

Флажки в формах задаются с помощью конструкции `<input type="checkbox">`, а переключатель — при помощи `<input type="radio">`.

Флажков, в отличие от переключателей, в одной форме может быть установлено несколько. Если для флажков указан атрибут `checked`, то при загрузке станицы на соответствующих полях формы флажки уже будут установлены.

Элемент `<label>` применяется при реализации выбора с помощью переключателей и флажков. Можно выбрать нужный пункт, просто щелкая кнопкой мыши на тексте, связанном с ним. Для этого нужно поместить `<input>` внутрь элемента `<label>`.

Git/GitHub

Git - это консольная утилита, для отслеживания и ведения истории изменения файлов, в вашем проекте. Чаще всего его используют для кода, но можно и для других файлов. Например, для картинок - полезно для дизайнеров.

С помощью Git-а вы можете откатить свой проект до более старой версии, сравнивать, анализировать или сливать свои изменения в репозиторий.

Репозиторием называют хранилище вашего кода и историю его изменений. Git работает локально и все ваши репозитории хранятся в определенных папках на жестком диске.

Git - это одна из систем контроля версий. По существу это значит, что она хранит всю историю изменений проекта. Эти команды определят информацию, которая будет использоваться при каждом commit(фиксирование изменений). Их стоит выполнить всего один раз при первичной установке Git.

```
git config --global user.name "Ваше Имя"
git config --global user.email "ВашаПочта@mail.com"
```

```
git pull https://github.com/имя пользователя/имя пользователя.github.io.git
git remote add origin https://github.com/имя пользователя/имя пользователя.github.io.git
git branch -M main
git push -u origin main
```

https://habr.com/ru/post/541258/