# HTML

# Основы HTML

HTML-документ — это обычный текстовый документ, имеет расширение `.html`.

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

`<имя тега>...</имя тега>`

Между начальным и закрывающим тегами находится содержимое тега — контент.

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.

1.1. Элемент `<html>`

1.2. Элемент `<head>`

1.2.1. Элемент `<title>`

1.2.2. Элемент `<meta>`

1.2.3. Элемент `<style>`

1.2.4. Элемент `<link>`

1.2.5. Элемент `<script>`

1.3. Элемент `<body>`

HTML-документ состоит из двух разделов — заголовка — между тегами `<head>...</head>` и содержательной части — между тегами `<body>...</body>`.

# Структура HTML-документа

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

## Элемент <html>

Является корневым элементом документа. Все остальные элементы содержатся внутри тегов `<html>...</html>`. Все, что находится за пределами тегов, не воспринимается браузером как код HTML и никак им не обрабатывается.

## 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.

## Элемент <title>

Обязательным тегом раздела `<head>` является тег `<title>`. Текст, размещенный внутри этого тега, 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 <meta>

Необязательным тегом раздела `<head>` является одинарный тег `<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.

```
<meta name="description" content="Описание содержимого страницы">

<meta name="description" lang="en" content="Description">

<meta name="keywords" content="Ключевые слова через запятую">

<meta name="robots" content="index, follow">
```

## 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 <link>

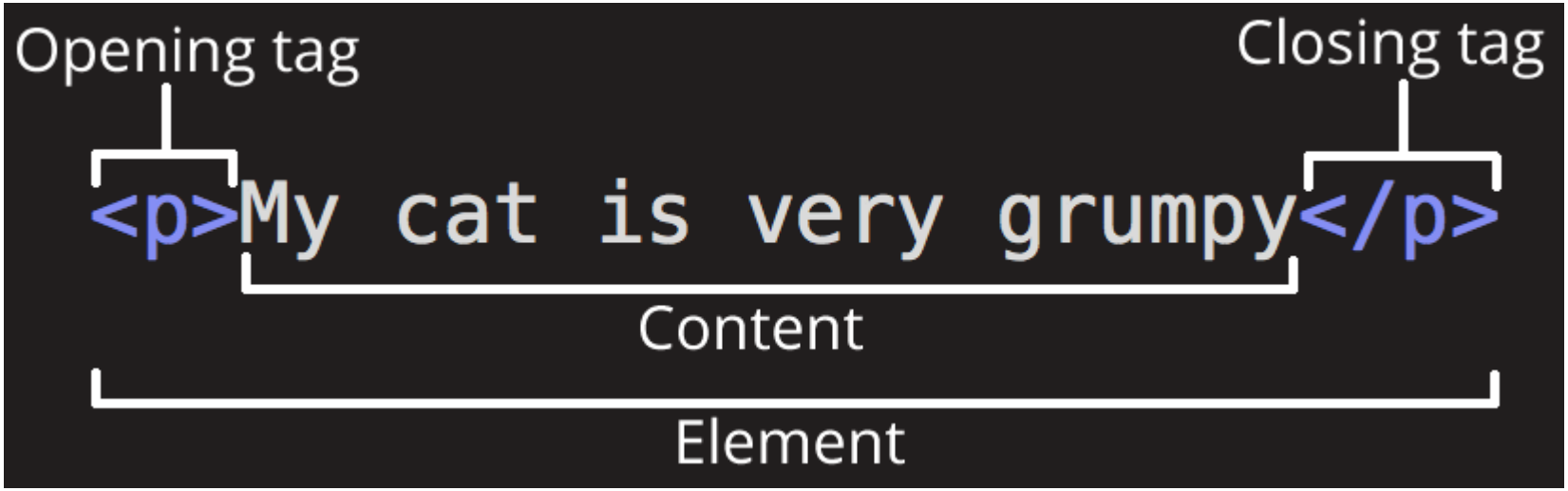
Задать стили для документа можно также при помощи другого способа — записать их в отдельный файл с расширением `.css`, например, `style.css`.

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

## Элемент <script>

Элемент `<script>` позволяет присоединять к документу различные сценарии. Закрывающий тег обязателен, при этом 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.

# Анатомия HTML элемента

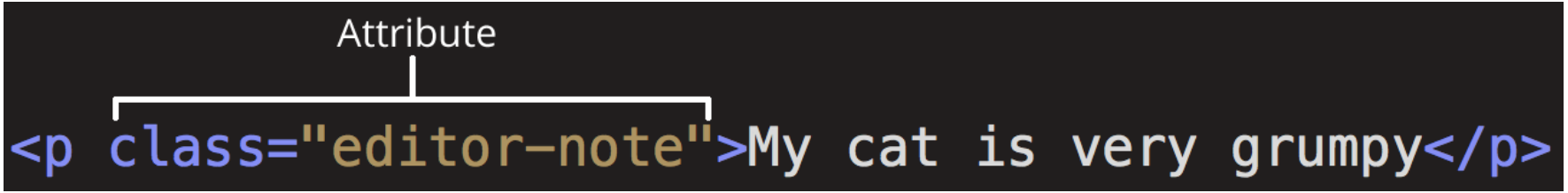


- **Открывающий тег (Opening tag)**: Состоит из имени элемента (в данном случае, "p"), заключенного в открывающие и закрывающие **угловые скобки**. Открывающий тег указывает, где элемент начинается или начинает действовать, в данном случае — где начинается абзац.

- **Закрывающий тег (Closing tag):** Это то же самое, что и открывающий тег, за исключением того, что он включает в себя косую черту перед именем элемента. Закрывающий элемент указывает, где элемент заканчивается, в данном случае — где заканчивается абзац.
- Отсутствие закрывающего тега является одной из наиболее распространенных ошибок начинающих и может приводить к странным результатам.

- **Контент (Content):** Это контент элемента, который в данном случае является просто текстом.
- **Элемент(Element):** Открывающий тег, закрывающий тег и контент вместе составляют элемент.

# Элементы также могут иметь атрибуты




- **Атрибуты** содержат дополнительную информацию об элементе, которую вы не хотите показывать в фактическом контенте.

- **Атрибут всегда должен иметь:**
- Пробел между ним и именем элемента (или предыдущим атрибутом, если элемент уже имеет один или несколько атрибутов).
- Имя атрибута, за которым следует знак равенства.
- Значение атрибута, заключенное с двух сторон в кавычки.

# Вложенные элементы

- Можно располагать элементы внутри других элементов — это называется вложением.
- Если мы хотим заявить, что наша кошка очень раздражена, мы можем заключить слово "очень" в элемент  , который указывает, что слово должно быть сильно акцентированно:

```
<p>Моя кошка <strong>очень</strong> раздражена.</p>
```

- Однако элементы должны быть правильно вложены: в примере выше мы открыли первым элемент <p>, затем элемент , потом мы должны закрыть сначала элемент , затем <p>. Приведенное ниже неверно:

```
<p>Моя кошка <strong>очень раздражена.</p></strong>
```

# Теги для HTML текста

- Теги заголовков: <h1...h6>
- Теги для форматирования текста: <b>, , <i>, <small>, , <sub>, <sup>, <ins>, <del>
- Теги для ввода «компьютерного» текста: <code>, <kbd>, <samp>, <var>, <pre>
- Теги для оформления цитат и определений: <abbr>, <bdo>, <blockquote>, <q>, <cite>, <dfn>
- Абзацы, средства переноса текста: <p>, <br>, <hr>

## Теги заголовков

Заголовки являются важными элементами веб-страницы, они упорядочивают текст, формируя его визуальную структуру.

Теги `<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 допускается вложение других тегов в теги `<h1>...<h6>`.

## Теги заголовков

- Тег `<h1>` - Заголовок самого верхнего уровня, на странице рекомендуется использовать только один раз, по возможности частично дублируя заглавие страницы.
- Тег `<h1>` должен быть уникальным для каждой страницы сайта. Рекомендуется прописывать тег в начале статьи, используя ключевое слово в тексте заголовка.
- Размер шрифта в браузере равен **2em**, верхний и нижний отступ по умолчанию **0.67em**.

# Теги заголовков

Теги `<h2>`, `<h3>`, `<h4>`, `<h5>`, `<h6>` - Обозначают подзаголовки второго, третьего, четвертого, пятого и шестого уровня. Размер шрифта в браузере равен **1.5em / 1.17em/ 1em** / **0.83em** / **0.67em**, верхний и нижний отступ по умолчанию **0.83em / 1em / 1.33em** / **1.67em**/ **2.33em** соответственно.

## Абзацы, средства переноса текста

- Тег `<p>` - Разбивает текст на отдельные абзацы, отделяя друг от друга пустой строкой. Браузер автоматически добавляет верхний и нижний отступ, равный `1em`, при этом отступы соседних абзацев «схлопываются».
- Тег `<br>` - Переносит текст на следующую строку, создавая разрыв строки.
- Тег `<hr>` - Используется для разделения контента на веб-странице. Отображается в виде горизонтальной линии.

# Преформатированный текст и код

- Тег `<pre>` (сокращение от «preformatted text»). Используется для отображения примеров кода, также применяется для отображения картинок ASCII Art. Браузер сохраняет и отображает все пробелы и переносы, которые есть внутри тега `<pre>`.
- Тег `<code>`. Используется для обозначения фрагментов кода.
- С его помощью размечается любой фрагмент текста, который распознается компьютером: код программы, разметки, название файла и так далее. Обычно браузеры отображают текст в теге `<code>` моноширинным шрифтом.

# Цитаты

- **Небольшие цитаты**
- Тег `<q>` (сокращение от «quote»). Предназначен для выделения цитат внутри предложения. Текст внутри тега браузер автоматически обрамляет кавычками, поэтому добавлять кавычки вручную не нужно.
- **Источник цитат**
- Тег `<cite>`. В нём можно указывать помимо адреса источника цитаты ещё и название произведения, откуда цитируется текст, а также имя автора или организации, чей текст цитируется. Содержимое `<cite>` в браузере выделяется курсивом.
- Может быть самостоятельным и не привязываться к цитате

- **Длинные цитаты**
- Тег `<blockquote>`. Предназначен для выделения длинных цитат, которые могут состоять из нескольких абзацев. Тег выделяет цитату не как фрагмент текста в предложении, а как отдельный блок текста с отступами.
- В браузере контенту тега `<blockquote>` обычно добавляется дополнительный отступ слева и справа.

# Разметка фрагментов текста

- **Символы-мнемоники**
- Это особые строки, которые начинаются с амперсанда (&) и заканчиваются точкой с запятой (;). Например, знак меньше на страницу можно вставить мнемоникой `<` (less than), а знак больше мнемоникой `>` (greater than):
- Некоторые символы в HTML зарезервированы, то есть браузер считает их HTML-кодом. Чтобы использовать специальные символы в тексте страницы как обычные символы их нужно заменить на символы-мнемоники.

- **Перенос строк**
- Тег <br> (сокращение от «line break»). Применяется, чтобы вставить в текст перенос строки, не создавая при этом абзац.
- **Верхний и нижний индексы**
- Теги <sup> и <sub>. Названия образованы от слов «superscript» и «subscript».
- Тег <sup> отображает текст в виде верхнего индекса, а <sub> отображает текст в виде нижнего индекса.
- Их используют для указания единиц измерения или для написания простых формул

- Тег `<time>`. С помощью него можно описывать даты одновременно и для человека, и для машины. Для указания даты в машиночитаемом формате ISO 8601 существует атрибут `datetime`

```
<time datetime="2016-11-18T09:54">09:54 утра</time>
```

- **Акцентирование внимания**
- Теги  и <i>. Названия образованы от слов «emphasis» и «italic». Предназначены для акцентирования внимания на слово или фразу. Визуально оба тега одинаковы, они выделяют текст курсивом.
- Тег  определяет текст, на который сделан особый акцент, меняющий смысл предложения.
- Тег <i> применяется для обозначения текста, который отличается от окружающего текста, но не является более важным. Например, в <i> можно заключать названия, термины, иностранные слова.

- **Выделение и придание важности**
- Теги `<strong>` и `<b>`. Название `<b>` образовано от слова «bold». Отображаются оба тега одинаково, они выделяют текст жирным.
- Тег `<strong>` указывает на важность отмеченного текста.
- Тег `<b>` предназначен для выделения текста с целью привлечения к нему внимания, но без придания ему особой важности.

- **Описание изменений**
- Теги `<del>` и `<ins>`. Названия тегов образованы от слов «delete» и «insert». Предназначены для описания изменений в документе.

- Тег `<del>` выделяет текст, который был удалён в новой версии документа. В браузере этот текст перечёркивается.

- Тег `<ins>` выделяет текст, который был добавлен в новой версии документа. В браузере этот текст подчёркивается.

# Разделение контента

- Теги `<div>` и `<span>`. Это «чистые» элементы, и обычно они отлично подходят в качестве обёртки для стилизации или группировки других элементов. Использовать эти теги рекомендуется в тех случаях, если более подходящих семантических тегов не нашлось.
- Тег `<div>` используется для группировки структурных элементов или в качестве вспомогательных контейнеров для создания нужной раскладки.
- Тег `<span>` используется для группировки текстовых элементов, выделения отдельных слов или фраз внутри абзацев, пунктов списка и так далее.

# 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 тега `<link>` и используются для расширения возможностей текущего документа при обработке браузером;

**гиперссылки** — ссылки на другие ресурсы, которые пользователь может посетить или загрузить.

## Структура ссылки

Гиперссылки создаются с помощью парного тега `<a></a>`.
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.
Обязательным параметром тега `<a>` является атрибут `href`, который задает URl-адрес веб-страницы.

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

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.
Адресная часть ссылки состоит из URL. **URL** (Uniform Resource Locator) — унифицированный адрес ресурса. При создании адресов для разделения слов между собой рекомендуется использовать дефис, а не символ подчёркивания.
В общем виде URl имеющий следующий формат:

```
метод доступа://имя сервера:порт/путь
```

**Метод доступа**, или протокол, осуществляет обмен данными между рабочими станциями в разных сетях. Наиболее распространенные протоколы передачи данных:

`file` обеспечивает чтение файла с локального диска:

```
file:/gallery/pictures/summer.html
```

`http` предоставляет доступ к веб-странице по протоколу HTTP:

```
http://site.ru/
```

`https` — специальная реализация протокола HTTP, использующая шифрование (как правило, SSL или TLS)

```
https://site.ru/
```

`ftp` осуществляет запрос к FTP-серверу на получение файла:

```
ftp://pgu/directory/library
```

`mailto` запускает сеанс почтовой связи с указанным адресатом и хостом:

```
mailto: nika@gmail.com
```

**Имя сервера** описывает полное имя машины в сети, например, site.ru. Если имя сервера не указано, то ссылка считается локальной, т.е. она относится к той же машине, на которой находится HTML-документ, содержащий ссылку.

**Номер порта TCP**, на котором функционирует веб-сервер. Представляет собой число, которое необходимо указывать, если метод требует номер порта (отдельные сервера могут иметь свой отличительный номер порта). Если порт не указан, по умолчанию используется порт 80. Стандартными портами являются:

21 — FTP  23 — Telnet  70 — Gopher  80 — HTTP

**Путь** содержит имя папки, в которой находится файл.

## Абсолютный и относительный путь

**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:

1) протокол, например, http (опционально);

2) домен (доменное имя или IP-адрес компьютера);

3) папка (имя папки, указывающей путь к файлу);

4) файл (имя файла).

## Относительный путь

**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.

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

# Ссылка с якорем

- Ссылки с якорем обычно используются для создания навигации внутри страницы. Например, оглавления в начале страницы с большой статьёй.
- Для создания такой навигации используются ссылки-якоря. Ссылка-якорь — это обычная ссылка, в адресе которой используется символ #, после которого следует идентификатор элемента.
- Идентификатор создаётся с помощью атрибута `id` у того тега, к которому надо перейти при щелчке по ссылке. Причём сам тег может быть любым: `<h1>`, `<section>`, `<p>` и так далее.

- Вот так выглядит адрес, состоящий из одного якоря:

```
<a href="#part1">Глава 1</a>
```

- При щелчке по такой ссылке браузер найдёт на странице элемент с соответствующим атрибутом `id` и прокрутит окно страницы к нему.

```
<article id="part1">Содержание первой
главы</article>
```

- При этом перезагрузки страницы не произойдёт.
- Интересно, что якорь также можно использовать и в абсолютных адресах, тогда после перехода на нужную страницу по аналогии произойдёт прокрутка к заданной части этой страницы.

# HTML-изображения

**HTML-изображения** добавляются на веб-страницы с помощью тега `<img>`.

С помощью HTML-тегов `<map>` и `<area>` можно 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>`.

Атрибут `src` задает путь к изображению. `src="flower.jpg"`.

Атрибут `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. `alt="описание изображения"`.

Атрибуты `height, width` задают высоту изображения. Может указываться в `px` или `%`. (`height: 300px; width: 100%`)

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.

# Форматы изображений

- Существует несколько основных форматов изображений: JPEG, PNG, SVG и GIF.
- Формат SVG переводится как масштабируемая векторная графика. Качество таких изображений не меняется при изменении размеров, да и вес у них небольшой. Отлично подходит для малоцветных схем, логотипов и иконок. Чаще всего используется в случаях, когда необходимо масштабировать изображение без потерь, изменять цвет элементов изображения, анимировать части изображения.

- Формат JPEG подходит для фотографий, рисунков с большим количеством разноцветных деталей, изображений с плавным переходом яркости и контраста. При сжатии изображения ухудшается его качество.
- Формат PNG позволяет сохранять изображения, в которых требуется особенная чёткость. Главная особенность этого формата — поддержка прозрачности. Подходит для изображений с прозрачностью и полупрозрачностью, когда необходима повышенная точность полноцветных изображений и для изображений с резкими переходами цветов.

- Формат GIF используется для простейших анимаций. В последнее время GIF-изображения становятся всё менее используемыми и заменяются на другие, более оптимальные форматы (векторная анимация – tgs).

# Изображение-ссылка

- Ссылки можно делать не только с помощью текста, но и с помощью изображений. Для этого нужно обернуть тег `<img>` в тег `<a>`. Например:

```
<a href="http://keksby.ru">
    <img src="cat.png" alt="Кекс">
</a>
```

# Элемент <figure>

- Для добавления иллюстраций, графиков, диаграмм, фотографий применяется элемент `<figure>`.
- Его содержимое не ограничивается изображениями, допустимо вставлять видео, примеры кода, даже текст.
- Внутри `<figure>` при желании добавляется заголовок с помощью элемента `<figcaption>` до или после содержимого.

```
<figure>
    <img src="image/fig4.png" width="600"
     height="416" alt="Рис. 4">
    <figcaption>Рис. 4. - Название</figcaption>
</figure>
```

- Для изменения положения текста — сверху или снизу изображения, достаточно просто поменять местами элементы `<img>` и `<figcaption>`.
- Для размещения сбоку применяем свойство `float` к селектору `figcaption`, но надо помнить, что это свойство воздействует на все нижележащие элементы тоже.

# Карты-изображения

```
<img src="image/navigate.png" width="640"
height="30" alt="Навигация по сайту"
usemap="#navigation">
```

Для указания того, что изображение является картой, применяется атрибут `usemap` элемента `<img>`. В качестве значения используется указатель на описание конфигурации карты, которая устанавливается с помощью элемента `<map>`. Значение атрибута `name` у `<map>` должно соответствовать имени в `usemap`. При этом значение `usemap` в `<img>` начинается с символа решётки.

Внутри контейнера <map> располагается один или несколько элементов <area>, они задают форму области, её координаты, устанавливают адрес документа, на который следует сделать ссылку, а также всплывающую подсказку.

```
<map name="navigation">
  <area shape="poly"
    coords="113,24,211,24,233,0,137,0"
    href="page/inform.html" alt="Информация"
    title="Информация">
```

…

Элемент <area> имеет следующие атрибуты.

- shape — определяет форму активной области. Форма может быть в виде окружности (circle), прямоугольника (rect), полигона (poly).
- alt — добавляет альтернативный текст для каждой области. Служит лишь комментарием для ссылки, поскольку на экран не выводится. Обязательный атрибут при наличии href.
- title — выводит всплывающую подсказку при наведении курсора на область.

- `href` — задаёт адрес документа, на который следует перейти, по своему действию аналогичен подобному атрибуту элемента `<a>`.
- `coords` — задаёт координаты активной области. Координаты отсчитываются в пикселях от левого верхнего угла изображения, которому соответствует значение 0, 0. Первое число является координатой по горизонтали, второе — по вертикали. Список координат зависит от формы области.

- Для окружности задаются три числа — координаты центра круга и радиус.

```
<area shape="circle" coords="230, 340, 100"
href="circle.html" alt="">
```

- Для прямоугольника — координаты левого верхнего и правого нижнего угла.

```
<area shape="rect" coords="24,18, 210, 56"
href="rect.html" alt="">
```

# 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

**Маркированный список** представляет собой неупорядоченный список *(от англ. Unordered List)*. Создаётся с помощью парного тега `<ul></ul>`.

В качестве маркера элемента списка выступает метка, например, закрашенный кружок.

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

```
<ul>
   <li>Microsoft</li>
   <li>Google</li>
   <li>Apple</li>
   <li>IBM</li>
</ul>
```

- Microsoft
- Google
- Apple
- IBM

## Нумерованный список

**Нумерованный список** создаётся с помощью парного тега `<ol></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 тега `<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 тега `<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">`.

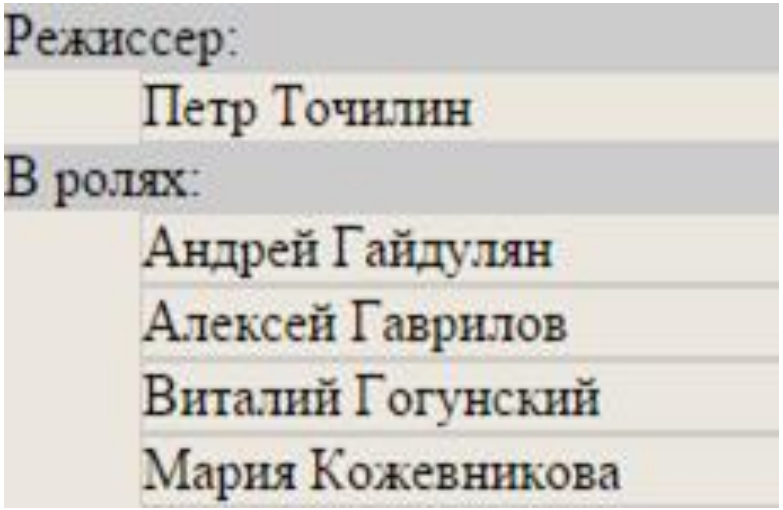
Атрибут `type` задает вид маркера для использования в списке (в виде букв или цифр). Принимаемые значения:

`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).

## Список определений

**Списки определений** создаются с помощью тега `<dl></dl>`. Для добавления термина применяется тег `<dt></dt>`, а для вставки определения — тег `<dd></dd>`.

```
<dl>
  <dt>Режиссер:</dt>
    <dd>Петр Точилин</dd>
  <dt>В ролях:</dt>
    <dd>Андрей Гайдулян</dd>
    <dd>Алексей Гаврилов</dd>
    <dd>Виталий Гогунский</dd>
    <dd>Мария Кожевникова</dd>
</dl>
```



# Вложенные списки

- Теги <ol> и <ul> можно вкладывать друг в друга и создавать многоуровневые списки. Количество уровней в списках не ограничено.
- Сначала нужно создать список первого уровня, а затем между тегами <li> и </li> этого списка добавить ещё один список. При этом необходимо аккуратно закрывать все теги

```
<ol>
  <li>1
    <ul>
      <li>1.1</li>
      <li>1.2</li>
    </ul>
  </li>
  <li>2</li>
</ol>
```

# 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.

- Таблица создаётся при помощи парного тега `<table></table>`. Данный тег является контейнером для элементов таблицы и все элементы должны находиться внутри него.
- По умолчанию таблица и ячейки не имеют видимых границ. Границы задаются с помощью свойства `border`.
- Промежутки между ячейками таблицы убираются с помощью свойства `table {border-collapse: collapse;}`.
- Ширина таблицы по умолчанию равна ширине её внутреннего содержимого. Чтобы установить ширину, нужно задать значение для свойства `width`.

```
<table>
      <tr>    <!--ряд с ячейками заголовков-->
            <th>текст заголовка1</th>
            <th>текст заголовка2</th>
      </tr>
      <tr>    <!--ряд с ячейками тела таблицы-->
            <td>данные1</td>
            <td>данные2</td>
      </tr>
</table>
```

- Строки или ряды таблицы создаются с помощью тега `<tr>`. Кол-во горизонтальных строк таблицы равно кол-ву парных тегов `<tr></tr>`.
- Элемент `<th>` создаёт заголовок столбца — специальную ячейку, текст в которой выделяется полужирным. Кол-во ячеек заголовка равно кол-ву пар тегов `<th></th>`.
- Элемент `<td>` создаёт ячейки таблицы, внутрь которых помещаются данные таблицы. Парные теги `<td></td>`, расположенные в одном ряду, определяют кол-во ячеек в строке таблицы. Кол-во пар ячеек `<td>` должно быть равно количеству пар ячеек `<th>`.

- Элемент `<caption>` создает подпись таблицы. Добавляется непосредственно после тега `<table>`, вне строки или ячейки.
- 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 `<caption>`.
- Элемент `<col>` формирует группы столбцов, которые делят таблицу на разделы, не относящиеся к общей структуре

```
<table>
  <colgroup>
    <col span="2" style="background:red>
    <col style="background-color:LightCyan>
  </colgroup>
  <tr>
    <th>№ п/п</th>     <th>Наим.</th>     <th>Цена, руб.</th>
  </tr>
  <tr>
    <td>1</td>     <td>Карандаш </td>     <td>20,00</td>
  </tr>
  <tr>
    <td>2</td>     <td>Линейка </td>     <td>30,00</td>
  </tr>
</table>
```

# Группировка разделов таблицы

- Элемент `<thead>` создает группу заголовков для строк таблицы с целью задания единого оформления.
- Элемент `<tbody>` группирует основное содержимое таблицы.
- Элемент `<tfoot>` создает группу строк для представления информации о суммах или итогах, расположенную в нижней части таблицы.

# Как объединить ячейки таблицы

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

```
<tr>
<td colspan="5" style="…"> ИТОГО: </td>
<td>1168,80</td>
<!-- Задаем количество ячеек по горизонтали для
объединения-->
</tr>
```

| № п/п | Наименование товара | Ед. изм. | Количество | Цена за ед. изм., руб. | Стоимость, руб. |
|---|---|---|---|---|---|
| 1. | Томаты свежие | кг | 15,20 | 69,00 | 1048,80 |
| 2. | Огурцы свежие | кг | 2,50 | 48,00 | 120,00 |
| | | | | ИТОГО: | 1168,80 |

# Семантические элементы HTML5

**Семантические элементы HTML5** доступно описывают свой смысл или назначение как для браузеров, так и для веб-разработчиков.

До появления стандарта HTML5 вся разметка страниц осуществлялась преимущественно с помощью элементов `<div>`, которым присваивали классы `class` или идентификаторы `id` для наглядности разметки (например, `<div id="header">`). С их помощью в HTML-документе размещали верхние и нижние колонтитулы, боковые панели, навигацию и многое другое.

Стандарт HTML5 предоставил новые элементы для структурирования, группировки контента и разметки текстового содержимого.

Новые семантические элементы позволили улучшить структуру веб-страницы, добавив смысловое значение заключенному в них содержимому (было `<div id="header">`, стало `<header>`).

Для отображения внешнего вида элементов не задано никаких правил, поэтому элементы можно стилизовать по своему усмотрению.

## Элемент <header>

Группирует вводные и навигационные элементы, не является обязательным. Может содержать заголовки, оборачивать содержание раздела страницы, форму поиска или логотип. В HTML-документе может содержаться одновременно несколько элементов `<header>` и они могут располагаться в любой части страницы.

Элемент `<header>` нельзя помещать внутрь элементов `<footer>`, `<address>` или другого элемента `<header>`.

## Элемент <nav>

Предназначен для создания блока навигации веб-страницы или всего веб-сайта, при этом не обязательно должен находиться внутри <header>.

На странице может быть несколько элементов <nav>.

Не заменяет теги <ul> или <ol>, он просто их обрамляет.

Не все группы ссылок на странице должны быть обёрнуты <nav>, этот элемент предназначен в первую очередь для разделов, которые состоят из главных навигационных блоков.

## Элемент <article>

Используется для группировки записей — публикаций, статей, записей блога, комментариев.

Представляет собой независимый обособленный блок, предназначенный для многократного использования, как правило, начинается с заголовка.

Может дублироваться на других страницах сайта и содержать внутри другие элементы `<article>`, которые по содержанию имеют близкое отношение к содержанию внешней статьи.

Если на странице присутствует только одна статья с заголовком и текстовым содержимым, она не нуждается в обёртке элементом <article>.

Элемент рекомендуется использовать только в том случае, если содержимое элемента будет явно указано в схеме документа.

## Элемент <section>

Элемент представляет собой универсальный раздел документа.

Группирует тематическое содержимое и обычно содержит заголовок.

Не является блоком-оберткой, для этих целей уместнее использовать элемент `<div>`.

В качестве содержимого может выступать оглавление, разделы научных публикаций, программа мероприятия.

Домашняя страница сайта также может быть поделена на секции — блок вводной информации, новости и контакты.

Элемент рекомендуется использовать только в том случае, если содержимое элемента будет явно указано в схеме документа.

**<article> внутри <section>**

Можно создавать родительские элементы <section> с вложенными элементами <article>, в которых есть один или несколько элементов <article>.

Не все страницы должны быть устроены именно так, но это допустимый способ вложения элементов.

Например, основная область контента страницы содержит два блока со статьями разной тематики.

Можно сделать на этом акцент, поместив каждую статью одной тематики внутрь элемента <section>.

## Элемент <aside>

Группирует содержимое, связанное с окружающим его контентом напрямую, но которое можно счесть отдельным *(т.е., удаление этого блока не повлияет на понимание основного содержимого)*.

Чаще всего элемент позиционируется как боковая колонка (как в книгах) и включает в себя группу элементов: `<nav>`, цифровые данные, цитаты, рекламные блоки, архивные записи.

Не подходит для блоков, просто позиционированных в стороне.

## Элемент <footer>

Представляет собой нижний колонтитул содержащей его секции или корневого элемента.
Обычно содержит информацию об авторе статьи, данные о копирайте и т.д.

Если используется как колонтитул всей страницы, содержимое дополняется сведениями об авторских правах, ссылками на условия использования, контактную информацию, ссылками на связанное содержимое и т.п.

В одном веб-документе может быть несколько элементов <footer>.

Как каждая страница, так и каждая статья может иметь свой элемент <footer>, более того, <footer> можно поместить в элемент <blockquote>, чтобы указать источник цитирования.

# Элемент <address>

Используется для определения контактной информации автора/владельца документа или статьи.

Для обозначения автора документа тег размещают внутри элемента <body>, для отображения автора статьи — внутри тега <article>.

В браузере обычно отображается курсивом.

## Элемент <main>

Элемент `<main>` представляет основное содержимое документа (содержимое элемента `<body>`).
Контент, находящийся внутри элемента, должен быть уникальным и не повторяться во всех документах сайта, таких как навигационные ссылки, информация о копирайте, логотипы, формы поиска (в случае, если форма поиска является основной функцией документа).

Элемент `<main>` не может быть потомком таких элементов как `<article>`, `<aside>`, `<footer>`, `<header>` или `<nav>`.

## Элемент <figure> <figcaption>

Элемент `<figure>` представляет автономный контент (необязательно с заголовком), являющийся самостоятельным элементом основного потока.

Элемент может быть перемещен из основного содержимого документа в боковую колонку или приложение, не затрагивая поток документа. С

помощью элемента `<figure>` можно добавлять краткие характеристики к иллюстрациям, фотографиям, диаграммам, фрагментам кода и т.д..

Элемент `<figcaption>` — потомок элемента `<figure>`, не принадлежит ни к одной категории контента.

Элемент является блочным, по ширине равен ширине элемента `<figure>`, высота по умолчанию равна `18px`.

## Элемент `<time>`

Определяет время (24 часа) или дату по григорианскому календарю с возможным указанием времени и смещения часового пояса.

Текст, заключенный в данный тег, не имеет стилевого оформления браузером.

Для тега доступен атрибут `datetime`, в качестве содержимого которого указывается то, что будет видеть пользователь на экране своего компьютера

- Конец 1 лекции 2021 осень

# HTML формы

# Элементы HTML форм

- Форма в HTML-документе реализуется тегом-контейнером FORM, в котором задаются все управляющие элементы - поля ввода, кнопки и.т.д.
- Если управляющие элементы указаны вне содержимого тега FORM, то они не создают форму, а используются для построения пользовательского интерфейса на веб-странице, то есть для привнесения в нее различных кнопок, флажков, полей ввода.

- Обработка элементов формы производится с помощью скриптов, но они могут и вообще никак не обрабатываться.
- Имена элементам формы присваиваются через их атрибут NAME.

# Тег FORM - контейнер форм

- Как уже было сказано, форма в HTML-документе реализуется тегом-контейнером FORM.
- Этот тег своими атрибутами указывает адрес сценария (скрипта), которому будет послана форма, способ пересылки и характеристику данных, содержащихся в форме.
- Начальный и конечный теги FORM задают границы формы, поэтому их указание является обязательным.

# Атрибуты тега FORM

- **action-** единственный обязательный атрибут.
- В качестве его значения указывается URL-адрес запрашиваемого скрипта, которая будет обрабатывать данные, содержащиеся в форме.
- Допустимо использовать запись mailto:URL, благодаря которой форма будет послана по электронной почте.
- Если атрибут ACTION все-таки не указан, то содержимое формы будет отправлено на URL-адрес, с которого загружалась данная веб-страница;

- **method -** определяет метод HTTP, используемый для пересылки данных формы от браузера к серверу. Атрибут METHOD может принимать два значения: GET и POST;
- **enctype -** необязательный атрибут. Указывает тип содержимого формы, используемый для определения формата кодирования при ее пересылке. В HTML определены два возможных значения для атрибутов ENCTYPE:

- APPLICATION/X-WWW-FORM-URLENCODED (используется по умолчанию);
- MULTIPART/FORM-DATA.

# Кнопки

- Кнопки - задаются с помощью элементов BUTTON и INPUT. Различают:
  - кнопки отправки - при нажатии на них, они осуществляют отправку формы серверу;
  - кнопки сброса - при нажатии на них, управляющие элементы принимают первоначальные значения;
  - прочие кнопки - кнопки, для которых не указано действие, выполняемое по умолчанию при нажатии на них.

# Зависимые переключатели

- Зависимые переключатели (переключатели с зависимой фиксацией) - задаются элементом INPUT и представляют собой переключатели "вкл/выкл".
- Если несколько зависимых переключателей имеют одинаковые имена, то они являются взаимоисключающими.
- Это значит, что если одна из них ставится в положение "вкл", то все остальные автоматически - в положение "выкл".
- Именно это и является преимуществом их исп-я.

# Независимые переключатели

- Независимые переключатели (переключатели с независимой фиксацией) - задаются элементом INPUT
- Представляют собой переключатели "вкл/выкл", но в отличие от зависимых, независимые переключатели могут принимать и изменять свое значение независимо от остальных переключателей.
- Даже если последние имеют такое же имя.

# Меню. Ввод текста.

- Меню - реализуется с помощью элементов SELECT, OPTGROUP и OPTION.
- Меню предоставляют пользователю список возможных вариантов выбора.
- Ввод текста - реализуется элементами INPUT, если вводится одна строка, и элементами TEXTAREA - если несколько строк.
- В обоих случаях введенный текст становится текущим значением управляющего элемента.

# Выбор файлов.
# Скрытые управляющие элементы

- Выбор файлов - позволяет вместе с формой отправлять выбранные файлы, реализуется HTML-элементом INPUT.
- Скрытые управляющие элементы - создаются управляющим элементом INPUT.

# Группировка элементов формы

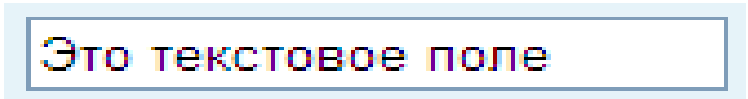
- Элемент `<fieldset>...</fieldset>` предназначен для группировки элементов, связанных друг с другом, разделяя таким образом форму на логические фрагменты.
- Каждой группе элементов можно присвоить название с помощью элемента `<legend>`, который идет сразу за тегом `<fieldset>`. Название группы проявляется слева в верхней границе `<fieldset>`.

# Тег INPUT и его методы

- Элемент INPUT является наиболее употребительным тегом HTML-форм.
- С помощью этого тега реализуются основные функции формы. Он позволяет создавать внутри формы поля ввода строки текста, имени файла, пароля и.т.д.
- Обратите внимание на особенность INPUT - у него нет конечного (завершающего) тега.
- Атрибуты и особенности использования INPUT зависят от способа его использования.

# Однострочные поля ввода

- Формат тега INPUT для создания поля ввода текстовой строки:
- `<input type=“text” name=“имя_параметра” [value=“значение”] [size=“размер_поля”] [maxlen=“длина_поля”]>`
- Данный тег создает поле ввода с максимально допустимой длиной текста `maxlen` и размером в `size` знакомест.




- Если указан атрибут `value`, то в поле будет изначально отображаться значение данного атрибута.

# Поля ввода пароля

- Конечно, имя пользователя можно ввести с помощью обыкновенного текстового поля. А вот пароль не должен отображаться на экране при его вводе. В этом нам поможет поле ввода пароля:
- `<input type=password name=имя_параметра [value=значение] [size=размер] [maxlen=длина]>`
- Принцип работы данного тега точно такой же, как и текстового. Разница заключается в том, что вводимая информация в поле не отображается, а заменяется "звездочками".
- Не рекомендуется устанавливать значение по умолчанию из соображений безопасности (`value`).

# Скрытое текстовое поле

- Для передачи служебной информации (о которой пользователь даже не должен подозревать) используются скрытые поля.
- С помощью таких полей, например, могут передаваться параметры настройки.

```
<input type=hidden name=имя_параметра
value=значение>
```

- Такие поля передаются серверу, но на веб-странице не отображаются.

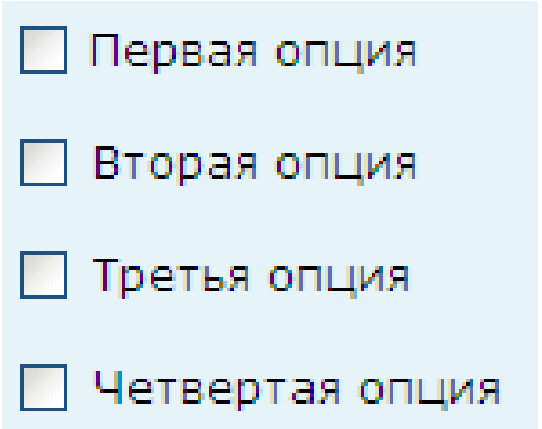
# Независимые переключатели

- Описывают значения, а рядом с ними помещается небольшое квадратное поле, в котором можно установить, или убрать галочку.
- При этом значение, соответственно, будет либо выбрано, либо нет.
- Для этого только необходимо в качестве значения атрибута type указать chechbox.

```
<input type=checkbox name=имя_пар-ра
value=значение [checked]>
```

- Если переключатель был включен на момент нажатия кнопки отправки данных, то скрипту будет передан параметр имя=значение.
- Если же флажок выключен, то сценарию вообще ничего не будет передано - как будто переключателя вообще нет.
- Переключатель по умолчанию либо включен, либо выключен.
- Чтобы переключатель был по умолчанию включен, необходимо для него указать атрибут `checked`.

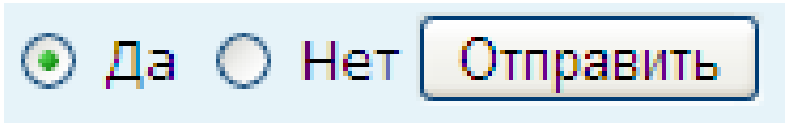
- Переключатель `checkbox` называется независимым, так как его состояние не зависит от состояния других переключателей `checkbox`.
- Таким образом, в одной форме может быть одновременно выбрано несколько переключателей.



# Зависимые переключатели

- Зависимые переключатель, так же как и независимый переключатель, может быть либо включен, либо выключен.
- При этом переключатель radio является зависимым переключателем, поскольку на форме может быть только один включенный переключатель типа radio.
- Точнее, если в форме присутствуют несколько одноименных зависимых переключателей, то включен из них может быть только один.

- При выборе одного переключателя все одноименные зависимые переключатели автоматически выключаются.
- В качестве имени переключателей воспринимается значение атрибута name. Может быть только один активный переключатель.
- `<input type=radio name=answer value=yes checked>Да`
- `<input type=radio name=answer value=no>Нет`



- `number` — предназначено для ввода целочисленных значений.
- `range` — позволит создать такой элемент интерфейса, как ползунок, min / max — позволят установить диапазон выбора
- 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.

- `password` — создает текстовые поля в форме, при этом вводимые пользователем символы заменяются на звездочки, маркеры, или другое.
- `search` — обозначает поле поиска, по умолчанию поле ввода имеет прямоугольную форму.
- `url` — поле предназначено для указания URL-адресов.
- `email` — браузеры, поддерживающие данный атрибут, будут ожидать, что пользователь введет данные, соответствующие синтаксису адресов электронной почты.

- `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.
- `month` — позволяет пользователю вводить год и номер месяца по шаблону гггг-мм.

- `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 (53).
- `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.

# Кнопка отправки формы

- Еще одним элементом управления типа INPUT являются кнопки. Кнопка отправки служит для отправки скрипту введенных в форму параметров. Синтаксис тега INPUT при этом такой:
- `<input type=submit [name=go] value=Отправить>`

- Атрибут `value` определяет текст, который будет написан на кнопке отправки. Атрибут name определяет имя кнопки и является необязательным.
- Если значение этого атрибута не указывать, то скрипту будут переданы введенные в форму значения и все.
- Если атрибут `name` для кнопки будет указан, то дополнительно к основным данным формы будет отправлена пара имя=значение от самой кнопки.

# Кнопка сброса параметров

- Кроме кнопки submit есть еще кнопка reset, которая сбрасывает параметры формы, а точнее, устанавливает для всех элементов формы значения по умолчанию.
- Желательно, чтобы на форме была такая кнопка, особенно, если это большая форма.
- Наличие данной кнопки обеспечивает очистку формы, например, в случае, когда были введены неправильные параметры. Синтаксис кнопки сброса:
- `<input type=reset value=Сброс>`

# Кнопка отправки с рисунком

- Вместо кнопки submit можно использовать рисунок для отправки данных. Клик на этом рисунке дает то же самое, что и нажатие на кнопку submit.
- Однако, кроме этого, сценарию будут переданы координаты места клика на рисунке.
- Координаты будут переданы в формате имя.x=коор_X, y=коор_Y. Синтаксис кнопки отправки с рисунком:

```
<input type=image name=имя src=рис.>
```

# Многострочные текстовые поля. Тег TEXTAREA

- Поле, создаваемое этим тегом, позволяет вводить и отправлять не одну строку, а сразу несколько строк. Синтаксис тега TEXTAREA:

```
<textarea name=имя
[cols=ширина_в_символах]
[rows=высота_в_символах]
wrap=тип_переноса>
текст по умолчанию
</textarea>
```

- Несколько значений относительно использования атрибутов: необязательные параметры `cols` и `rows` желательно все-таки указывать.
- Первый из них задает количество символов в строке, а второй - количество строк в области.
- Атрибут wrap определяет тип переноса текста, как будет выглядеть текст в поле ввода:

- `virtual` - справа от текстового поля выводится полоса прокрутки. Вводимый текст выглядит разбитым на строки, а символ новой строки вставляется при нажатии клавиши ENTER;
- `physical` - этот тип зависит от типа браузера и выглядит по-разному;
- `none` - текст выглядит в поле в том виде, в котором пользователь его вводит. Если текст не уменьшается в одну строку, появляется горизонтальная полоса прокрутки.
- Следует заметить, что наиболее удобным является тип `virtual`.

Это текст, введенный в многострочное текстовое поле по умолчанию

# Списки выбора. Тег SELECT

- **Списки с единственным выбором**
- Довольно часто существует необходимость представить какие-нибудь данные в виде списка и предусмотреть возможность выбора в этом списке.
- В HTML списки реализуются с помощью тега SELECT.
- Список выбора позволяет выбрать один вариант из множества.

```
<select name=day size=1>
 <option value=1>Понедельник</option>
 <option value=2>Вторник</option>
 <option value=3
selected>Среда</option>
 <option value=4>Четверг</option>
 <option value=5>Пятница</option>
 <option value=6>Суббота</option>
 <option value=7>Воскресенье</option>
 </select>
```

- Атрибут `name` определяет имя параметра, который будут передан скрипту.
- Если атрибут `size` равен 1, то список будет "оснащен" полосой прокрутки. Значение, выбранное в списке по умолчанию, можно указать с помощью атрибута `selected` для соответствующего тега option.
- В приведенном примере день недели по умолчанию - Среда.
- Атрибут `value` является необязательным.
- Если его не указать, то будет передана строка, заключенная в тег `option`.

# Списки множественного выбора

- С помощью тега SELECT можно также создавать списки множественного выбора.
- В таких списках можно выбрать не одно, а сразу несколько вариантов значений.
- Для того, чтобы создать список с множественным выбором, необходимо для тега SELECT указать атрибут multiple.
- Вот практический пример такого списка:

```
<select name=day size=7 multiple>
 <option value=1>Понедельник</option>
 <option value=1>Вторник</option>
 <option value=1>Среда</option>
 <option value=1>Четверг</option>
 <option value=1>Пятница</option>
 <option value=1>Суббота</option>
 <option value=1>Воскресенье</option>
 </select>
```

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

# Загрузка файлов на сервер

- Тег INPUT позволяет реализовать еще одну возможность форм, а именно создавать поле выбора файла для его отправки на сервер. Синтаксис следующий:
- `<input type=file name=имя [value=имя_файла]>`

# 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>`

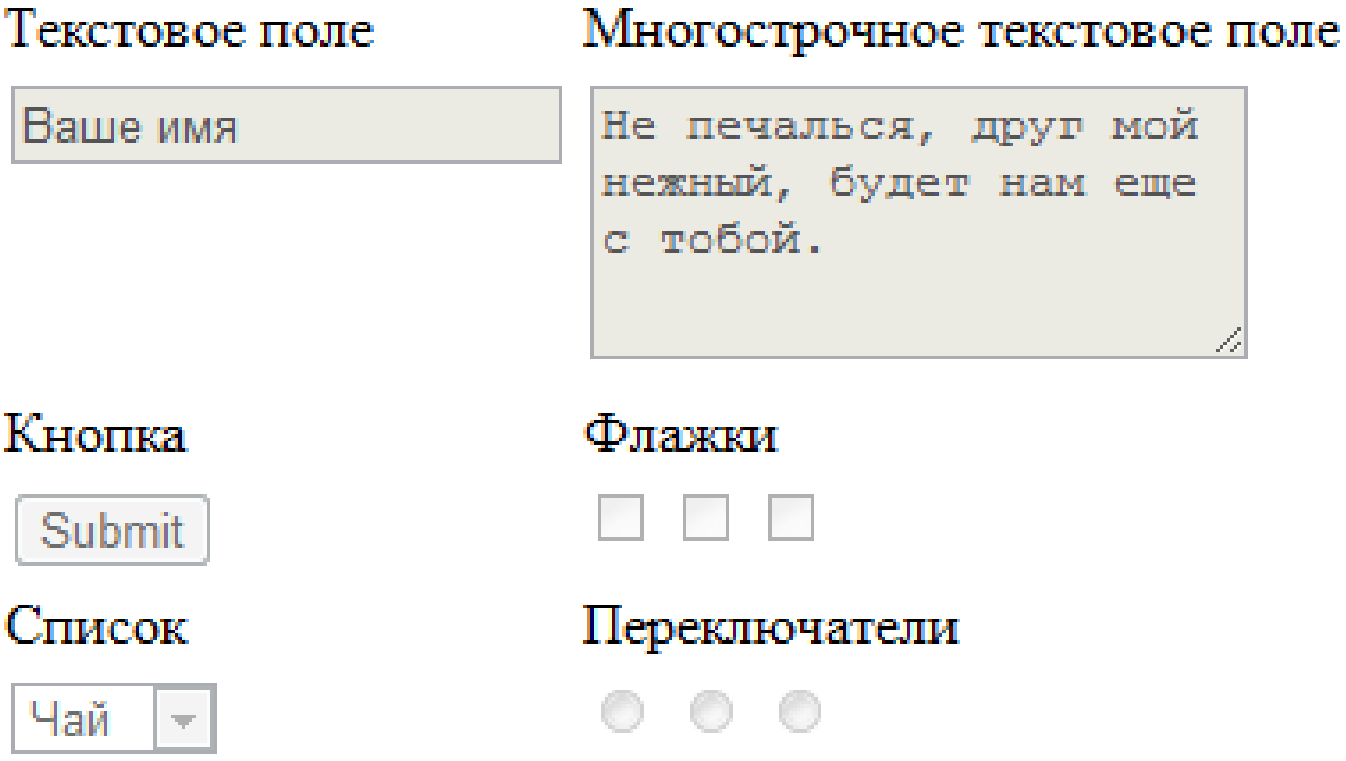
```
<!-- без атрибута for -->
<p><label>Кошка<input id="cat"
type="checkbox"></label></p>
```

`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.

# Блокирование элементов форм

- У любого элемента формы есть два состояния, которые ограничивают доступ к элементу или ввод данных, — блокирование (disabled) и только для чтения (readonly).
- Блокирование элемента не позволяет вообще производить с ним каких-либо действий, в том числе выделять содержимое текстового поля, изменять его или активировать. К тому же такие поля не пересылаются на сервер.
- Некоторые браузеры позволяют выделять и копировать содержимое заблокированного текстового поля, но все остальные действия недоступны.

- Для блокирования элемента формы используется атрибут `disabled`. Добавление этого атрибута разрешает отображать элемент формы, но не позволяет изменять его.



# Поле только для чтения

- Поля формы можно не только блокировать, но и переводить их в режим только для чтения. В этом случае доступ к ним сохраняется, но изменять значения заданные по умолчанию нельзя. Разумеется, речь идёт только о полях, где требуется вводить текст. Выделять и копировать текст можно, но изменить не получится.
- Для установки режима «только для чтения» используется атрибут `readonly`, он добавляется к элементу `<input>` или `<textarea>`. На вид элемента формы это никак не влияет, но как было уже замечено, модифицировать значение поля не удастся.

# Автофокус

- Фокус это активность элемента формы, позволяющая производить с ним какие-то действия. Для текстового поля можно вводить текст, для списка выбирать пункт с помощью клавиатуры и др.
- Автофокус — это автоматически установленный фокус поля формы.
- Автофокус создаётся с помощью атрибута `autofocus`, который можно добавлять к следующим элементам: `<button>`, `<input>`, `<keygen>`, `<select>`, `<textarea>`.

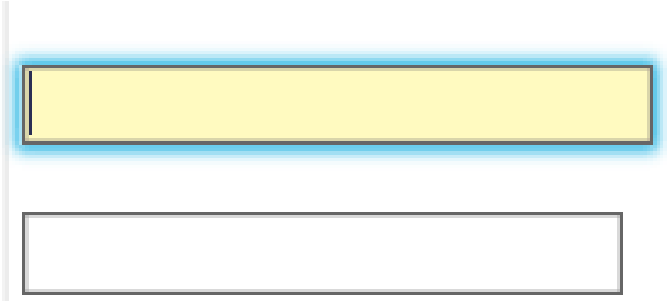
- Для текстового поля синтаксис такой.

  ```
  <input autofocus>
  ```

- На странице должен быть только один элемент с автофокусом.

- Поле с фокусом можно изменить через стили воспользовавшись псевдоклассом `:focus`, добавляя его к селектору `input`.

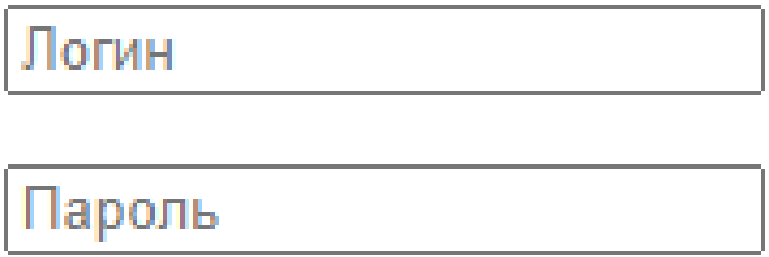
```
<style>
   input {
    border: 1px solid #666; /*Параметры рамки*/
   }
   input:focus {
    box-shadow: 0 0 5px 1px #00a8de; /*
                              Свечение вокруг поля */
    background: #fffac0; /* Цвет фона */
   }
  </style>
```



# Подсказывающий текст

- В дизайне часто требуется вставить пояснение к текстовому полю, но не всегда для этого имеется место. Решением в таком случае является добавление подсказывающего текста непосредственно внутрь поля, а при получении фокуса или вводе текста подсказка пропадает.
- Это делается с помощью атрибута `placeholder`, значением которого служит любой текст.
- Подсказка делается для полей `text`, `password`, `search`, `email`, `tel`, `url` и `<textarea>`, иными словами, везде, где вводится текст.

- Подсказывающий текст отображается серым цветом, и в зависимости от браузера исчезает при получении фокуса или вводе текста. Подсказка не появляется при наличии атрибута `value` с непустым значением.

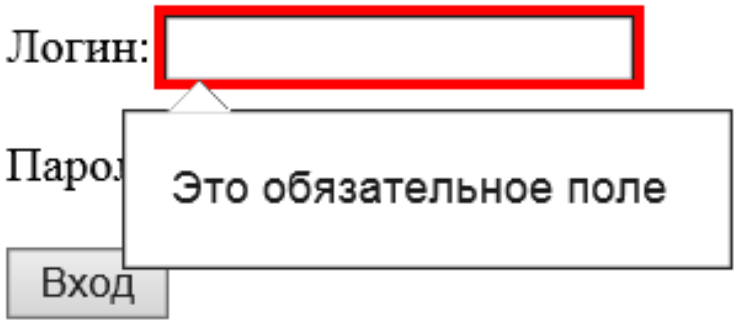


# Валидация средствами HTML

- Валидацией называется комплекс мер по пресечению ввода неправильной информации в форме.
- Например, если в поле требуется ввести положительное число от 0 до 10, то следует проверить, чтобы пользователь не ввёл текст или число, которое не лежит в указанном диапазоне, т. е. число не должно быть меньше нуля и больше десяти.

- Валидация на стороне клиента позволяет быстро проверить данные, вводимые пользователем, на корректность без отправки формы на сервер.
- Таким образом экономится время пользователя и снижается нагрузка на сервер.
- С позиции юзабилити тоже имеются плюсы — пользователь сразу получает сообщение о том, какую информацию он указал неверно и может исправить свою ошибку.

- **Обязательное поле**
- Некоторые поля формы должны быть обязательно заполнены перед их отправкой на сервер. Это, к примеру, относится к форме регистрации, где требуется ввести логин и пароль.
- Для указания обязательных полей используется атрибут `required`
- `<input name="login" required>`

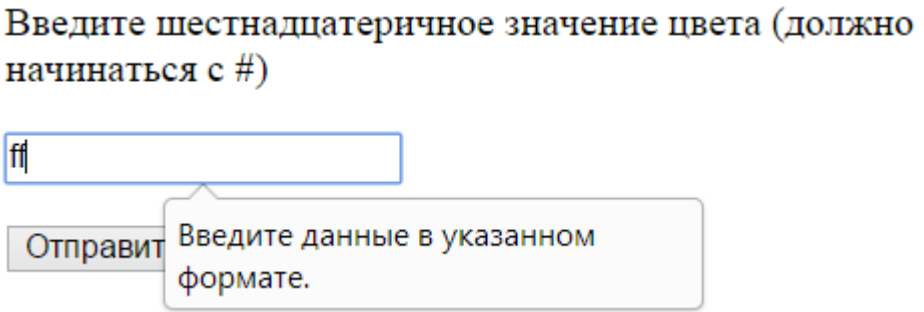


- **Корректность данных**
- Исходно имеется два поля в котором вводимые пользователем данные проверяются автоматически.
- Это веб-адрес и адрес электронной почты. Для этих элементов характерны следующие правила.
- Веб-адрес (`<input type="url">`) должен содержать протокол (http://, https://, ftp://).
- Адрес электронной почты (`<input type="email">`) должен содержать буквы или цифры до символа @, после него, затем точку и домен первого уровня.
- У браузеров несколько различается политика по проверке данных пользователя. Chrome и Opera требуют, чтобы в почтовом адресе была точка, для Firefox она не обязательна.

- **Шаблон ввода**
- Некоторые данные нельзя отнести к одному из видов элементов формы, поэтому для них приходится использовать текстовое поле.
- При этом их ввод происходит по определённому стандарту. Так, IP-адрес содержит четыре числа разделённых точкой (192.168.0.1), почтовый индекс России ограничен шестью цифрами (124007), телефон содержит код города и конкретное количество цифр часто разделяемых дефисом (391 555-341-42) и др.

- Браузеру необходимо указать шаблон ввода, чтобы он согласно нему проверял вводимые пользователем данные.
- Для этого используется атрибут `pattern`, а его значением выступает регулярное выражение.

```
<input name="digit" required
    pattern="#[0-9A-Fa-f]{6}">
```



| Выражение | Описание |
| --- | --- |
| \d [0-9] | Одна цифра от 0 до 9. |
| \D [^0-9] | Любой символ кроме цифры. |
| \s | Пробел. |
| [A-Z] | Только заглавная латинская буква. |
| [A-Za-z] | Только латинская буква в любом регистре. |
| [А-Яа-яЁё] | Только русская буква в любом регистре. |
| [A-Za-zA-Яа-яЁё] | Любая буква русского и латинского алфавита. |
| [0-9]{3} | Три цифры. |
| [A-Za-z]{6,} | Не менее шести латинских букв. |
| [0-9]{,3} | Не более трёх цифр. |
| [0-9]{5,10} | От пяти до десяти цифр. |
| ^[a-zA-Z]+$ | Любое слово на латинице. |
| ^[А-Яа-яЁё\s]+$ | Любое слово на русском включая пробелы. |
| ^[ 0-9]+$ | Любое число. |

# Отмена валидации

- Валидация не всегда требуется для формы, разработчик может пожелать использовать универсальное решение на JavaScript и дублирующая проверка браузером ему уже ни к чему. В подобных случаях необходимо отключить встроенную валидацию. Для этого применяется атрибут `novalidate` элемента `<form>`.

```
<form novalidate>
    …
</form>
```

- Для аналогичной цели применяется и атрибут `formnovalidate`, который добавляется к кнопке для отправки формы, в данном случае к `<input type="submit">`.

```
<form>
  <p><input name="user" required
    placeholder="Ваше имя"></p>
  <p><input type="submit" value="Отправить"
    formnovalidate></p>
</form>
```

# Основы CSS

**CSS (Cascading Style Sheets), или каскадные таблицы стилей**, используются для описания внешнего вида документа, написанного языком разметки.

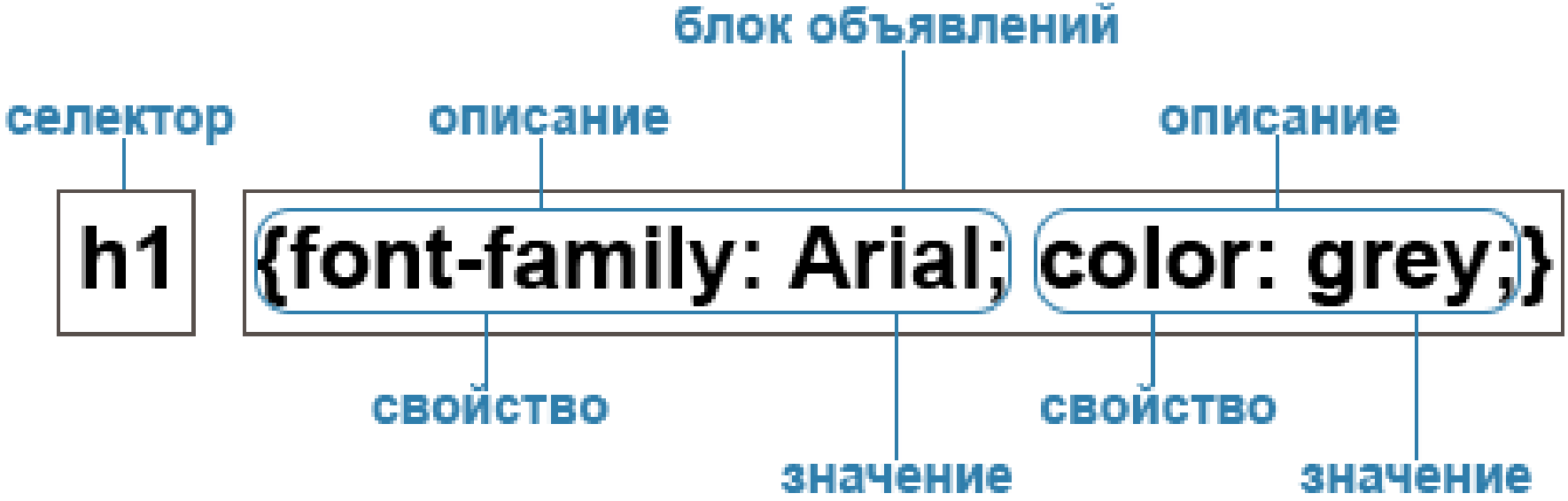
Обычно CSS-стили используются для создания и изменения стиля элементов веб-страниц и пользовательских интерфейсов, написанных на языках HTML и XHTML, но также могут быть применены к любому виду XML-документа, в том числе XML, SVG и XUL.

Каскадные таблицы стилей описывают правила форматирования элементов с помощью свойств и допустимых значений этих свойств.

Для каждого элемента можно использовать ограниченный набор свойств, остальные свойства не будут оказывать на него никакого влияния.

Объявление стиля состоит из двух частей: элемента веб-страницы — **селектора**, и команды форматирования — **блока объявления**.

Селектор сообщает браузеру, какой именно элемент форматировать, а в блоке объявления (код в фигурных скобках) перечисляются форматирующие команды — свойства и их значения.


блок объявлений
селектор
описание
описание
h1
{font-family: Arial;
color: grey;}
свойство
значение
свойство
значение

# Виды таблиц стилей

**Внешняя таблица стилей** представляет собой текстовый файл с расширением .css, в котором находится набор CSS-стилей элементов.

Внутри файла могут содержатся только стили, без HTML-разметки.

Внешняя таблица стилей подключается к веб-странице с помощью тега `<link>`, расположенного внутри раздела `<head></head>`. Такие стили работают для всех страниц сайта.

К каждой веб-странице можно присоединить несколько таблиц стилей, добавляя последовательно несколько тегов `<link>`, указав в атрибуте те`rel="stylesheet"`га `media` назначение данной таблицы стилей. указывает тип ссылки (ссылка на таблицу стилей).

```
<head>

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

<link rel="stylesheet" href="css/assets.css"
media="all">

</head>
```

**Внутренние стили** встраиваются в раздел `<head></head>` HTML-документа и определяются внутри тега `<style></style>`.

Внутренние стили имеют приоритет над внешними, но уступают встроенным стилям (заданным через атрибут `style`).

**Встроенные стили**

Когда мы пишем **встроенные стили**, мы пишем CSS-код в HTML-файл, непосредственно внутри тега элемента с помощью атрибута `style`:

```
<p style="font-weight: bold; color: red;">Обратите внимание на этот текст.</p>
```

Такие стили действуют только на тот элемент, для которого они заданы.

**Правило @import**

**Правило `@import`** позволяет загружать внешние таблицы стилей. Чтобы директива `@import` работала, она должна располагаться в таблице стилей (внешней или внутренней) перед всеми остальными правилами:

```
<style>

@import url(mobile.css);

p {...}

</style>
```

# Виды селекторов

**Селекторы** представляют структуру веб-страницы.

С их помощью создаются правила для форматирования элементов веб-страницы.

Селекторами могут быть элементы, их классы и идентификаторы, а также псевдоклассы и псевдоэлементы.

**Универсальный селектор**

Соответствует любому HTML-элементу.

Например, `* {margin: 0;}` обнулит внешние отступы для всех элементов сайта.

Также селектор может использоваться в комбинации с псевдоклассом или псевдоэлементом:

`*:after {CSS-стили}`,

`*:checked {CSS-стили}`.

**Селектор элемента**

Селекторы элементов позволяют форматировать все элементы данного типа на всех страницах сайта.

Например, `h1 {font-family: Lobster, cursive;}` задаст общий стиль форматирования всех заголовков `h1`.

**Селектор класса**

Селекторы класса позволяют задавать стили для одного и более элементов с одинаковым именем класса, размещенных в разных местах страницы или на разных страницах сайта.

Например, для создания заголовка с классом `headline` необходимо добавить атрибут `class` со значением `headline` в открывающий тег `<h1>` и задать стиль для указанного класса. Стили, созданные с помощью класса, можно применять к другим элементам, не обязательно данного типа.

```
<h1 class="headline">Инструкция пользования персональным компьютером</h1>
<p class="headline"> Тест параграфа </p>

.headline {
text-transform: uppercase;
color: lightblue;
}
```

## Селектор идентификатора

Селектор идентификатора позволяет форматировать **один** конкретный элемент.

Идентификатор `id` должен быть уникальным и на одной странице может встречаться только один раз.

```
<div id="sidebar"> ... </div>

#sidebar {
      width: 300px;
      float: left;
}
```

**Селектор потомка**

Селекторы потомков применяют стили к элементам, расположенным внутри элемента-контейнера.

Например, `ul a {text-transform: uppercase;}` — выберет все элементы `a`, являющиеся потомками всех элементов `ul`.

```
<ul>
      <li> <a> link </a> </li>
       <li> <a> link </a> </li>
</ul>
```

**Дочерний селектор**

Дочерний элемент является прямым потомком содержащего его элемента. У одного элемента может быть несколько дочерних элементов, а родительский элемент у каждого элемента может быть только один.

Дочерний селектор позволяет применить стили только если дочерний элемент идёт сразу за родительским элементом и между ними нет других элементов, то есть дочерний элемент больше ни во что не вложен.

Например, `p > strong` — выберет все элементы `strong`, являющиеся дочерними по отношению к элементу `p`.

**Сестринский селектор**

Сестринские отношения возникают между элементами, имеющими общего родителя. Селекторы сестринских элементов позволяют выбрать элементы из группы элементов одного уровня.

`h1 + p` — выберет все первые абзацы, идущие непосредственно за любым тегом `<h1>`, не затрагивая остальные абзацы;

`h1 ~ p` — выберет все абзацы, являющиеся сестринскими по отношению к любому заголовку `h1` и идущие сразу после него.

```
<h1> Title </h1>
<p> qwewqewq </p>
<p> dsadasdasd </p>
<p> dsadasdasd </p>
<p> dsadasdasd </p>

h1+p {color:red;}
```

```
<h1> Title </h1>
<p> qwewqewq </p>
<p> dsadasdasd </p>
<p> dsadasdasd </p>
<p> dsadasdasd </p>

h1~p {color:red;}
```

**Селектор атрибута**

Селекторы атрибутов выбирают элементы на основе имени атрибута или значения атрибута:

`[атрибут]` — все элементы, содержащие указанный атрибут, `[alt]` — все элементы, для которых задан атрибут `alt`; `селектор[атрибут]` — элементы данного типа, содержащие указанный атрибут, `img[alt]` — только картинки, для которых задан атрибут `alt`; `селектор[атрибут="значение"]` — элементы данного типа, содержащие указанный атрибут с конкретным значением, `img[title="flower"]` — все картинки, название которых содержит слово `flower`;

`селектор[атрибут~="значение"]` — элементы частично содержащие данное значение, например, если для элемента задано несколько классов через пробел, `p[class~="feature"]` — абзацы, имя класса которых содержит `feature`;

`селектор[атрибут|="значение"]` — элементы, список значений атрибута которых начинается с указанного слова, `p[class|="feature"]` — абзацы, имя класса которых `feature` или начинается на `feature`;

`селектор[атрибут^="значение"]` — элементы, значение атрибута которых начинается с указанного значения, `a[href^="http://"]` — все ссылки, начинающиеся на `http://`;

`селектор[атрибут$="значение"]` — элементы, значение атрибута которых заканчивается указанным значением, `img[src$=".png"]` — все картинки в формате `png`;

`селектор[атрибут*="значение"]` — элементы, значение атрибута которых содержит в любом месте указанное слово, `a[href*="book"]` — все ссылки, название которых содержит `book`.

**Селектор псевдокласса**

Псевдоклассы — это классы, фактически не прикрепленные к HTML-тегам. Они позволяют применить CSS-правила к элементам при совершении события или подчиняющимся определенному правилу.

Псевдоклассы характеризуют элементы со следующими свойствами:

`:link` — не посещенная ссылка;

`:visited` — посещенная ссылка;

`:hover` — любой элемент, по которому проводят курсором мыши;

`:focus` — интерактивный элемент, к которому перешли с помощью клавиатуры или активировали посредством мыши;

`:active` — элемент, который был активизирован пользователем;

`:valid` — поля формы, содержимое которых прошло проверку в браузере на соответствие указанному типу данных;

`:invalid` — поля формы, содержимое которых не соответствует указанному типу данных;

`:enabled` — все активные поля форм;

`:disabled` — заблокированные поля форм, т.е., находящиеся в неактивном состоянии;

`:in-range` — поля формы, значения которых находятся в заданном диапазоне;

`:out-of-range` — поля формы, значения которых не входят в установленный диапазон;

`:lang()` — элементы с текстом на указанном языке;

`:not(селектор)` — элементы, которые не содержат указанный селектор — класс, идентификатор, название или тип поля формы — `:not([type="submit"])`;

`:target` — элемент с символом `#`, на который ссылаются в документе;

`:checked` — выделенные (выбранные пользователем) элементы форм

**Селектор структурных псевдоклассов**

Структурные псевдоклассы отбирают дочерние элементы в соответствии с параметром, указанным в круглых скобках:

`:nth-child(odd)` — нечётные дочерние элементы;

`:nth-child(even)` — чётные дочерние элементы;

`:nth-child(3n)` — каждый третий элемент среди дочерних;

`:nth-child(3n+2)` — выбирает каждый третий элемент, начиная со второго дочернего элемента `(+2)`;

`:nth-child(n+2)` — выбирает все элементы, начиная со второго;

`:nth-child(3)` — выбирает третий дочерний элемент;

`:nth-last-child()` — в списке дочерних элементов выбирает элемент с указанным местоположением, аналогично с `:nth-child()`, но начиная с последнего, в обратную сторону;
`:first-child` — позволяет оформить только самый первый дочерний элемент тега;
`:last-child` — позволяет форматировать последний дочерний элемент тега;
`:only-child` — выбирает элемент, являющийся единственным дочерним элементом;
`:empty` — выбирает элементы, у кот. нет дочерних элементов;
`:root` — выбирает элемент, являющийся корневым в документе — элемент `html`.

**Селектор структурных псевдоклассов типа**

Указывают на конкретный тип дочернего тега:

`:nth-of-type()` — выбирает элементы по аналогии с `:nth-child()`, при этом берёт во внимание только тип элемента;

`:first-of-type` — выбирает первый дочерний элемент данного типа;

`:last-of-type` — выбирает последний элемент данного типа;

`:nth-last-of-type()` — выбирает элемент заданного типа в списке элементов в соответствии с указанным местоположением, начиная с конца;

`:only-of-type` — выбирает единственный элемент указанного типа среди дочерних элементов родительского элемента.

**Селектор псевдоэлемента**

Псевдоэлементы используются для добавления содержимого, которое генерируется с помощью свойства `content`:

`:first-letter` — выбирает первую букву каждого абзаца, применяется только к блочным элементам;

`:first-line` — выбирает первую строку текста элемента, применяется только к блочным элементам;

`:before` — вставляет генерируемое содержимое перед элементом;

`:after` — добавляет генерируемое содержимое после элемента.

**Комбинация селекторов**

Для более точного отбора элементов для форматирования можно использовать комбинации селекторов:

`a[href][title]` — выберет все ссылки, для которых заданы атрибуты `href` и `title`;

`img[alt*="css"]:nth-of-type(even)` — выберет все четные картинки, альтернативный текст которых содержит слово `css`.

**Группировка селекторов**

Один и тот же стиль можно одновременно применить к нескольким элементам. Для этого необходимо в левой части объявления перечислить через запятую нужные селекторы:

```
h1, h2, p, span {

color: tomato;

background: white;

 }
```

# Наследование и каскад

Наследование и каскад — два фундаментальных понятия в CSS, которые тесно связаны между собой.

Наследование заключается в том, что элементы наследуют свойства от своего родителя (элемента, их содержащего).

Каскад проявляется в том, как разные виды таблиц стилей применяются к документу, и как конфликтующие правила переопределяют друг друга.

**Наследование** является механизмом, с помощью которого определенные свойства передаются от предка к его потомкам.

Спецификацией CSS предусмотрено наследование свойств, относящихся к текстовому содержимому страницы, таких как `color`, `font`, `letter-spacing`, `line-height`, `list-style`, `text-align`, `text-indent`, `text-transform`, `visibility`, `white-space` и `word-spacing`. Во многих случаях это удобно, так как не нужно задавать размер шрифта и семейство шрифтов для каждого элемента веб-страницы.

Свойства, относящиеся к форматированию блоков, не наследуются.

Это `background`, `border`, `display`, `float` и `clear`, `height` и `width`, `margin`, `min-max-height` и `-width`, `outline`, `overflow`, `padding`, `position`, `text-decoration`, `vertical-align` и `z-index`.

**Принудительное наследование**

С помощью ключевого слова `inherit` можно принудить элемент наследовать любое значение свойства родительского элемента.
Это работает даже для тех свойств, которые не наследуются по умолчанию.

# Как задаются и работают CSS-стили

1) Стили могут наследоваться от родительского элемента (наследуемые свойства или с помощью значения `inherit`);
2) Стили, расположенные в таблице стилей ниже, отменяют стили, расположенные в таблице выше;
3) К одному элементу могут применяться стили из разных источников.
4) При определении стиля можно использовать любую комбинацию селекторов — селектор элемента, псевдокласса элемента, класса или идентификатора элемента.

**Каскадирование** — это механизм, который управляет конечным результатом в ситуации, когда к одному элементу применяются разные CSS-правила.

Существует три критерия, которые определяют порядок применения свойств — правило `!important`, специфичность и порядок, в котором подключены таблицы стилей.

## Правило `!important`

Вес правила можно задать с помощью ключевого слова `!important`, которое добавляется сразу после значения свойства, например, `span {font-weight: bold!important;}`.

Правило необходимо размещать в конец объявления перед закрывающей скобкой, без пробела. Такое объявление будет иметь приоритет над всеми остальными правилами.

Это правило позволяет отменить значение свойства и установить новое для элемента из группы элементов в случае, когда нет прямого доступа к файлу со стилями.

**Специфичность**

Для каждого правила браузер вычисляет **специфичность селектора**, и если у элемента имеются конфликтующие объявления свойств, во внимание принимается правило, имеющее наибольшую специфичность.

Значение специфичности состоит из четырех частей: 0, 0, 0, 0.

Специфичность селектора определяется следующим образом:

для `id` добавляется `0, 1, 0, 0`;

для `class` добавляется `0, 0, 1, 0`;

для каждого элемента и псевдоэлемента добавляется `0, 0, 0, 1`;

для встроенного стиля, добавленного непосредственно к элементу — `1, 0, 0, 0`;

универсальный селектор не имеет специфичности.

```
<p class="cl1">  </p>
p {color:red;}          //=1
.cl1 {color:blue;}      //=10
p.cl1 {color:blue;}     //=11
```

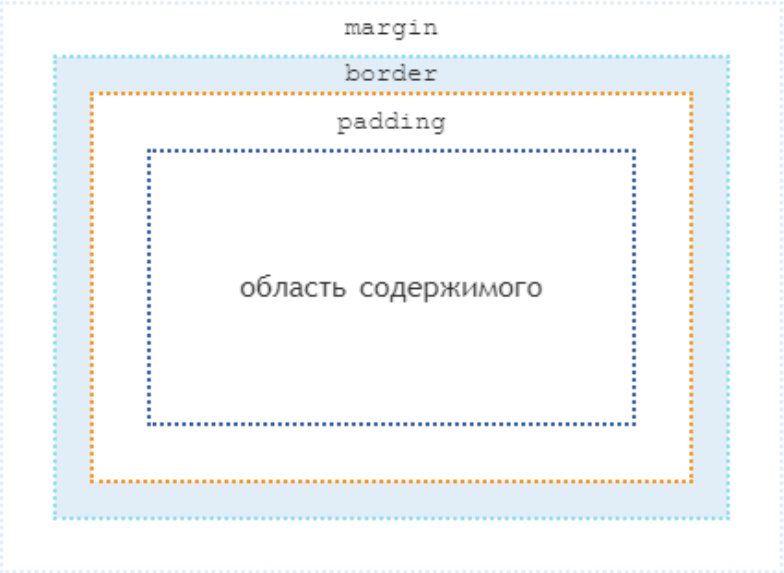
**Порядок подключённых таблиц**

Вы можете создать несколько внешних таблиц стилей и подключить их к одной веб-странице.

Если в разных таблицах будут встречаться разные значения свойств одного элемента, то в результате к элементу применится правило, находящееся в таблице стилей, идущей в списке ниже.

# CSS блочная модель

- Модуль CSS Box Model описывает свойства `padding` и `margin`, которые создают поля внутри и отступы снаружи CSS блока. Размеры блока также могут быть увеличены за счет рамки (`border`).
- Каждый блок имеет прямоугольную область содержимого в центре, поля вокруг содержимого, рамку вокруг полей и отступ за пределами рамки. Размеры этих областей определяют свойства `padding` и его подсвойства — `padding-left`, `padding-top` и т.д., `border` и его подсвойства, `margin` и его подсвойства.




- Каждый блок имеет область содержимого, в которой находится текст, дочерние элементы, изображение и т.п., и необязательные окружающие ее `padding`, `border` и `margin`. Размер каждой области определяется соответствующими свойствами и может быть нулевым, или, в случае `margin`, отрицательным.

# Отступы элемента

- Отступы окружают край рамки элемента, обеспечивая расстояние между соседними блоками.
- Свойства отступов определяют их толщину.
- Применяются ко всем элементам, кроме внутренних элементов таблицы.
- Сокращенное свойство margin задает отступы для всех четырех сторон, а его подсвойства задают отступ только для соответствующей стороны.
- Смежные вертикальные отступы элементов в блочной модели схлопываются.

# Схлопывание вертикальных отступов

- Смежные вертикальные отступы двух или более элементов уровня блока margin объединяются (перекрываются).
- При этом ширина общего отступа равна ширине большего из исходных.
- Исключение составляют отступы корневого элемента, которые не схлопываются.
- Объединение отступов выполняется только для блочных элементов в нормальном потоке документа.

- Если среди схлопывающихся отступов есть отрицательные значения, то браузер добавит отрицательное значение к положительному, а полученный результат и будет расстоянием между элементами.
- Если положительных отступов нет, то максимум абсолютных значений соседних отступов вычитается из нуля.

**Отступы не схлопываются**:

- Между плавающим блоком и любым другим блоком;
- У плавающих элементов и элементов со значением `overflow`, отличным от `visible`, со своими дочерними элементами в потоке;
- У абсолютно позиционированных элементов, даже с их дочерними элементами;
- У строчно-блочных элементов.

Область содержимого

margin-bottom: 30px;

Область содержимого

margin-bottom: 30px; width: 100%;
display: inline-block;

Область содержимого

margin-top: 15px;

Область содержимого

margin-top: 15px; width: 100%;
display: inline-block;

- **Выпадание вертикальных отступов**
- Если внутри одного блока расположить другой блок и задать ему margin-top, то внутренний блок прижмется к верхнему краю родительского, а у родительского элемента появится отступ сверху, т.е. внутренний блок «выпадет» из родительского блока.
- Если у родительского элемента также был задан верхний отступ, то выберется наибольшее из значений.

# Поля элемента

- Область полей представляет собой пространство между краем области содержимого и рамкой элемента.
- Свойства полей определяют толщину их области.
- Применяются ко всем элементам, кроме внутренних элементов таблицы (за исключением ячеек таблицы).
- Сокращенное свойство padding задает поля для всех четырех сторон, а подсвойства устанавливают только их соответствующие стороны.

## Рамки элемента

- Рамки элемента заполняют область рамок, визуально очерчивая края блока.
- Свойства рамок определяют толщину области границы блока, а также ее стиль и цвет.
- Стиль рамки `border-style`
- Цвет рамки `border-color`
- Ширина рамки `border-width`
- Свойство `border` позволяет объединить в себе следующие свойства: `border-width, border-style, border-color`

# Блочные и строчные элементы

- Выделяют две основные категории HTML-элементов, которые соответствуют типам их содержимого и поведению в структуре веб-страницы — блочные и строчные элементы.
- С помощью блочных элементов можно создавать структуру веб-страницы, строчные элементы используются для форматирования текстовых фрагментов (за исключением элементов `<area>` и `<img>`).

- **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.

- Блочные элементы могут размещаться непосредственно внутри элемента <body>. Они создают разрыв строки перед элементом и после него, образуя прямоугольную область, по ширине занимающую всю ширину веб-страницы или блока-родителя.
- Блочные элементы могут содержать как строчные, так и блочные элементы, но не оба типа элементов сразу.
- Блочные элементы могут содержаться только в пределах блочных элементов.

- 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>`.

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

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

- 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.

# Изменение блочной модели: свойство box-sizing

- Свойство `box-sizing` переключает блочную модель с фиксированных размеров длины и ширины на `content-box` и `border-box`.
- Это влияет на интерпретацию всех свойств, определяющих размеры, включая `flex-basis`.

**content-box**

- Это поведение ширины и высоты, как указано в CSS2.1.
- Заданные ширина и высота (и соответствующие min/max-свойства) применяются к ширине и высоте области содержимого элемента.
- Поля `padding` и рамка `border` элемента располагаются за пределами указанной ширины и высоты.
- Значение по умолчанию.

## border-box

- Любые `padding` или `border`, заданные для элемента, размечаются и отрисовываются внутри указанных значений ширины и высоты.
- Ширина и высота содержимого вычисляются путем вычитания ширины границ и полей соответствующих сторон из указанных свойств ширины и высоты.

- Значение `auto` свойств `width` и `height` не зависит от свойства `box-sizing` и всегда устанавливает размер блока с содержимым.
- Сумма `padding` и `border` не должна превышать заданные значения `width` и `height`, в противном случае размер области содержимого будет равен нулю.

# Единицы измерения

# Относительно шрифта: em

1em – текущий размер шрифта. Можно брать любые пропорции от текущего шрифта: 2em, 0.5em и т.п.

**Размеры в em – *относительные*, они определяются по текущему контексту.**

```
<div style="font-size:1.5em">

  Студенты

  <div style="font-size:1.5em">Студенты ИС</div>

</div>
```

Так как значение в `em` высчитывается относительно *текущего шрифта*, то вложенная строка в `1.5` раза больше, чем первая.

Выходит, размеры, заданные в `em`, будут уменьшаться или увеличиваться вместе со шрифтом. С учётом того, что размер шрифта обычно определяется в родителе, и может быть изменён ровно в одном месте, это бывает очень удобно.

## em

Когда единицы измерения em используются в дочерних элементах, которые не имеют определенного размера шрифта, они наследуют его от родителей, вплоть до корневого элемента документа.

```
.example {
    font-size: 20px;
}
```

```
.example {
    font-size: 20px;
    border-radius: .5em;
}
```

Значение border-radius равное .5em будет равно 10px (то есть 20 * 0,5). Аналогично:

```
.example {
    font-size: 20px;
    border-radius: .5em;
    padding: 2em;    }
```

В данном случае 1em этого элемента или его дочерних элементов (при отсутствии других определений `font-size`) будет равен 20px.

Если в CSS размер шрифта не определен, то em будет равна размеру шрифта, используемого по умолчанию в браузере. Чаще всего это значение составляет 16px.

# Проценты %

Проценты %, как и em – относительные единицы.

Когда мы говорим «процент», то возникает вопрос – «Процент от чего?»

Как правило, процент будет от значения свойства родителя с тем же названием, но не всегда.

Это очень важная особенность процентов, про которую, увы, часто забывают.

Отличный источник информации по этой теме – стандарт, Visual formatting model details.

# Примеры-исключения, в которых % берётся не так:

**margin-left**

При установке свойства margin-left в %, процент берётся от ширины родительского блока, а вовсе не от его margin-left.

**line-height**

При установке свойства line-height в %, процент берётся от текущего размера шрифта, а вовсе не от line-height родителя. Детали по line-height и размеру шрифта вы также можете найти в литературе.

**width/height**

Для width/height обычно процент от ширины/высоты родителя, но при position:fixed, процент берётся от ширины/высоты окна (а не родителя и не документа). Кроме того, иногда % требует соблюдения дополнительных условий, их можно дополнительно уточнить в литературе.

**Единица `rem` задаёт размер относительно размера шрифта элемента `<html>`.**

# Относительно экрана: vw, vh, vmin, vmax

Во всех современных браузерах, исключая IE8-, поддерживаются новые единицы из черновика стандарта CSS Values and Units 3:

- vw – 1% ширины окна
- vh – 1% высоты окна
- vmin – наименьшее из (vw, vh), в IE9 обозначается vm
- vmax – наибольшее из (vw, vh)

Эти значения были созданы, в первую очередь, для поддержки мобильных устройств.

Их основное преимущество – в том, что любые размеры, которые в них заданы, автоматически масштабируются при изменении размеров окна.