Методическое пособие по курсу "html"

Автор: старший преподаватель Алексей Тарасов

Основные понятия

Расширения: `*.html`, *.htm
Web-страница (текст, ссылки, картинки, списки, таблицы, формы, видео, аудио, анимацию, JavaScript)
web-сайт
web-сервер
web-адрес

Веб-страница

Веб-страница - это документ или информационный ресурс Всемирной паутины, доступ к которому осуществляется с помощью веб-браузера. Типичная веб-страница представляет собой текстовый файл в формате HTML, который может содержать ссылки на файлы в других форматах, а также гиперссылки для быстрого перехода на другие веб-страницы

Несколько веб-страниц, объединённых общей темой и дизайном, а также связанных между собой ссылками, образуют веб-сайт

Веб-сервер

Веб-сервер - компьютер, предоставляющий компьютерное и программное обеспечение, необходимое для функционирования веб-сайта.

Веб-адрес

Веб-адрес - адрес, по которому сайт или страница доступны на сервере в Интернет

Как это работает

1. Посетитель запрашивает страницу
2. Браузер запрашивает сервер
3. Сервер находит и отдаёт страницу
4. Браузер показывает страницу
5. Браузер запрашивает стили, скрипты, изображения...

Описание работы

Давайте опишем взаимодействие посетителя сайта, который запрашивает HTML страницу, с сервером.

Посетитель набирает адрес страницы в браузере. Браузер отправляет запрос на сервер, а сервер находит и отдаёт HTML страницу. Браузер получает и показывает HTML страницу и запрашивает новую информацию на сервере - изображения, скрипты, стилевые файлы.

История

1. 1991 – HTML
2. 1995 – CSS
3. 1996 – Internet Explorer
4. 1999, декабрь – HTML 4.0.1
5. 2012 – HTML 5

Редакторы HTML-кода

Notepad++
Brackets
Sublime Text
Coda (Mac)
Webstorm
Visual Studio Code

Редакторы HTML-кода

На первое время для работы нам не понадобится веб-сервер. Нужны будут только текстовый редактор и браузер. Существует множество текстовых редакторов, например, Notepad++, Sublime Text, Brackets. Также существуют интегрированные среды разработки IDE - Netbeans, WebStorm. Некоторые из них платные.

При работе с редактором помогает знание сочетаний быстрых клавиш. Есть общие сочетания для редакторов, типа копирования `Ctrl+c` или вставки `Ctrl+v` данных, но есть отличающиеся, например дублирование строки `Ctrl+d`, в Visual Studio Code выполняется при помощи `Ctrl+Shift+↓`

HTML-разметка

HTML-элемент заголовок

```
<h1>Текст заголовка</h1>
```

`<h1>` - открывающий тег

`</h1>` - закрывающий тег

HTML-элемент линия `<hr>`

`<hr>` - открывающий тег

`<hr />` - возможный вариант

HTML-разметка

HTML не является языком программирования, это язык разметки. Он определяет структуру веб-страниц, которые видят посетители сайта. Страницы могут иметь простую или сложную структуру, всё зависит от разработчика. Язык разметки состоит из ряда элементов, которые вы будете использовать чтобы разметить или обернуть различные фрагменты содержимого, чтобы добиться определённого результата. Элементы состоят из тегов, например элемент отвечающий за главный заголовок страницы `<h1>Заголовок</h1>` состоит из открывающего тега <h1> и закрывающего тега </h1>.

Открывающий тег состоит из названия (обозначения) элемента, помещённого внутри угловых скобок. Данный тег служит признаком начала элемента, с этого момента тег начинает влиять на следующее после него содержимое.

Закрывающий тег выглядит как и открывающий, но содержит слэш перед названием тега. Он служит признаком конца элемента.

Нужно не забывать записывать закрывающие теги у элементов, если этого требует HTML

HTML-комментарии

```
<!-- Комментарий -->
```

```
<!-- <h2>текст</h2> -->
```

Для снабжения документа комментариями, а также для отключения работы части содержимого страницы, используются наборы символов из угловых скобок, восклицательного знака и знаков тире.

Структура HTML-документа

```
<!DOCTYPE html> <!-- работаем по стандартам w3c -->
<html lang="ru"> <!-- корневой элемент -->
  <head> <!-- служебная информация -->
    <meta charset="UTF-8" /> <!-- указание на кодировку док-та -->
    <title>Видно во вкладках</title> <!-- название страницы -->
  </head>
  <body>
   <!-- часть, видимая в браузере -->
  </body>
</html>
```

Страница начинается с описания типа документа - doctype html. Это означает, что мы используем текущую версию языка гипертекстовый разметки. Неуказание doctype означает использование старой четвёртой версии HTML.

После, записывается корневой элемент HTML. Корнеевым он называется потому, что включает все другие элементы. То есть служебную информацию и часть, видимую пользователям в браузере.

Помимо элементов и тегов, в HTML существует понятие атрибутов. Атрибуты - это настройки html элементов. Атрибуты записываются в открывающем теге html элемента. Например, атрибут lang указывает на язык, который используется в документе или отдельном элементе. Lang равно ru в элементе html, означает что используется русский язык.

Внутри корневого элемента, находятся. Головная часть документа - head. Тело документа - body. Head содержит служебную информацию, например кодировку документа в элементе meta, описание документа, подключение стилей, элемент title с названием заглавия документа. Body содержит видимую для пользователя часть страницы браузера.

Обычно, для страниц указывается уникальное заглавие. То есть текст элемента title для разных страниц должен различаться. Это важно для тех, кто будет заниматься поисковым продвижением сайта.

HTML-атрибуты

```
<!DOCTYPE html>
<html lang="ru">  <!-- указание на язык документа -->
  <head>
    <meta charset="UTF-8" /> <!-- указание на кодировку док-та -->
    <title>Видно во вкладках</title>
  </head>
  <body>
    <h1 align=center>Заголовок</h1><!-- устаревший атрибут -->
  </body>
</html>
```

На наших мы будем всегда использовать универсальную кодировку utf-8. Потому у своих страниц в элементе `meta`, то есть элемент содержит *метаинформация*, нужно указать атрибут `charset` со значением utf-8. Это позволит избежать проблем с отображением страницы в разных браузерах.

Помимо полезных атрибутов, встречаются устаревшие. Типа атрибута `align`, который отвечает за выравнивание. Он был создан в предыдущих версиях HTML (Гипертекстовый язык разметки) и признан нехорошим. Причина проста, он влиял на внешнее оформление. Вместо него используется механизм каскадных стилей - CSS

Вложенность элементов

правильно

```
<h1>…</h1> <p>…</p>
```

неправильно

```
<h1>… <p> </h1>…</p>
```

Размещение одних элементов в других - называется вложенностью. Нужно всегда следить, чтобы элементы были вложены корректно. То есть, внутренний элемент не должен закрываться позже внешнего HTML-элемента.

Базовые блочные элементы

1. заголовки
2. блочная цитата
3. предварительное форматирование
4. параграф/абзац

Давайте поговорим о базовых блочных элементах html. Блочными они называются, потому что могут содержать другие элементы, но самое главное - создают блочную область для дальнейшей работы. У таких элементов ширина всегда составляет 100% от доступной. А текст после блочных элементов всегда выводится с новой строки

Если говорить упрощенно, блочные элементы помогают создать прямоугольные области на странице. Но это серьёзное упрощение, в дальнейшем мы вернёмся к этому вопросу.

Заголовки

```
<!DOCTYPE html>
<html lang="ru">
  <head>
  <meta charset="UTF-8" />
  <title>Заголовки</title>
</head>
<body>
  <!-- заголовок первого уровня  -->
  <h1>Самый главный и большой заголовок </h1>
  <!-- заголовок шестого уровня  -->
  <h6>Самый маленький заголовок для подразделов</h6>
</body>
</html>
```

Первые блочные элемент, которые мы посмотрим, будут заголовки. Название каждого тега заголовка начинается с буквы h, от слова header и последующей цифры от единицы до шести. Чем меньше цифра, тем более важный заголовок мы описываем. HTML не запрещает вставлять любое количество заголовков на страницу. Но те, кто продвигает сайты, рекомендуют указывать не более одного заголовка `h1` на странице. `h1` - самый важный заголовок.

Вспомните устройство заголовков в дипломной работе или реферате. Там тоже есть один важный заголовок - название работы. Потом идут менее важные заголовки - названия глав. Потом пункты, подпункты. Думаю теперь идея с заголовками стала более понятной

Блочная цитата

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta charset="UTF-8" />
    <title>Пословица</title>
  </head>
  <body>
    <h1>Пословица</h1>
    <blockquote>Ученье свет, а неученье — тьма.</blockquote>
  </body>
</html>
```

При необходимости привести цитату, можно воспользоваться элементом блочного тестирования `blockquote`.

Он создаёт отступы вокруг текста.

Параграф/Абзац

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Document</title>
  </head>
  <body>
    <h1>Lorem ipsum dolor.</h1>
    <!-- параграф/абзац -->  <p>Lorem ipsum..
    <!-- параграф/абзац -->  <p>Labore itaque ..</p>
  </body>
</html>
```

После заголовка обычно следует разбитый на фрагменты текст. Обычно, фрагменты заключается в блочный элемент параграф - `p`. Между параграфом и соседними элементами всегда присутствует некоторый отступ. Если разбить текст на параграфы, будет проще его читать.

У параграфов есть особенности. Одна из особенностей, у параграфа может отсутствовать закрывающий тег. Вторая особенность, параграф не может содержать блочных элементов. Любой блочный элемент автоматически закрывает параграф.

Предварительное форматирование

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta charset="UTF-8" />
    <title>Document</title>
  </head>
  <body>
    <h1>Форматирование</h1>
    <pre>Текст с несколькими пробелами      и переносом
    строки. Только шрифт
    отличается</pre>
  </body>
</html>
```

В html, несколько подряд идущих пробелов или табуляций воспринимаются как один пробел. И хотя можно вывести дополнительные пробелы через специальную последовательность символов, амперсанд nbsp точка с запятой, бывает нужно воспользоваться тегом предварительного форматирования `pre`

Например, им сопровождается вывод стихотворения, фрагмента программного кода или ASCII (American standard code for information interchange - название таблицы (кодировки, набора), в которой некоторым распространённым печатным и непечатным символам сопоставлены числовые коды) *графики*.

Текст внутри pre выводится моноширинным шрифтом. В дальнейшем это можно поменять при помощи CSS

Базовые строчные элементы

```
<cite>Цитата/Название работы</cite>
<strong>Важный текст</strong>
<em>Акцент на тексте</em>
<ins>Добавленный текст</ins>
<del>Удалённый текст</del>
<mark>Подсветка/выделение текста</mark>
<code>Компьютерный код</code>
<kbd>Пользовательский ввод CTRL+C</kbd>
<b>Ключевые слова</b>
<i>Альтернативный голос</i>
<u>Ключевые слова</u>
```

Базовые строчные элементы

<cite>*Цитата/Название работы*</cite>
Важный текст
Акцент на тексте
<ins>Добавленный текст</ins>
<del>~~Удалённый текст~~</del>
<mark>Подсветка/выделение текста</mark>
<code>Компьютерный код: let course = true</code>
<kbd>Пользовательский ввод: CTRL+C</kbd>
Ключевые слова
Альтернативный голос
<u>Ключевые слова</u>

Помимо блочных элементов могут понадобиться строчные элементы. Они помогают обрамлять отдельные слова или фразы, и не создают блочной области. У строчных элементов не указывается ширина или высота.

Строчный элемент `cite`, позволяет разметить цитату. Попробуйте сравнить работу этого элемента с блочным цитированием `blockquote`! У `cite` не создаётся таких отступов.

`strong` позволяет выделить важный текст. Укажите важность некоторых фраз или ключевых слов в тексте своей html-страницы. Текст который оформлен `strong`отображается полужирным начертанием. Одна из ошибок начинающих верстальщиков - использование элемента `strong` для получения полужирного текста. Запомните, главным является предназначение элемента, а внешний вид это следствие.

`em` указывает акцент на тексте. Внешне, текст отображается курсивом.

`mark` выделяет фрагменты текста, словно офисный маркер - ярко жёлтым цветом

Разрыв строки

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Document</title>
  </head>
  <body>
    <h1>Lorem ipsum dolor.</h1>
    <p>Lorem ipsum dolor sit amet, consectetur adipisicing<br>
elit. Quidem iste doloribus <br>
quisquam molestiae voluptate eaque iusto quia<br>
dolorem repellendus unde.
  </body>
</html>
```

Если вам понадобиться разбить параграф. Так, чтобы часть слов выводилась с новой строки. Нужно использовать элемент break, который не имеет закрывающего тега.

Элементами разрыва строки лучше не злоупотреблять. Некоторые, создают несколько переносов для увеличения отступа. Это очень плохая идея. Размер отступа лучше изменять через каскадные таблицы стилей. О них, речь пойдёт далее.

У элементов из одного тега, можно ставить слеш перед правой угловой скобкой.

Специальные символы

```
Специальные символы (ссылки мнемоники)
  &copy; &#169;  © Знак охраны авторского права
           Неразрывный пробел
  &laquo; &#171;  « Левая угловая - ёлочка
  &raquo; &#187;  » Правая угловая - ёлочка
  &mdash; &#8212; —  Длинное тире
  &lt; &#060; < Знак 'меньше'
  &gt; &#062; > Знак 'больше'
```

Другие символы

[другие последовательности]
[эмодзи 😀]
[unicode-символы]

Часть символов в html-коде можно выводить при помощи специальных последовательностей. Если нам нужен знак копирайта, воспользуемся знаком амперсанда `&`, и запишем copy, а затем завершим последовательность точкой с запятой `&copy;` Это приведёт к вставке знака копирайта ©.

Но этот знак можно вставить и без последовательности! Как насчёт угловых скобок, которые используются в html элементах. Вдруг понадобиться вывести на экран набор символов, а браузер подумает что вы указываете новый тег?

При этом валидатор будет недоволен, а браузер не поймёт как работать с этим как-бы тегом. Чтобы исправить ситуацию, можно вместо левой угловой скобки записать `<` или `&#171;`. В любом случае браузер отобразит левую угловую скобку.

Тест

Пройти

После каждой практической работы должна чувствоваться радость от проделанной работы. В этот момент надо взять паузу и освежить в памяти наиболее важные, и интересные моменты занятия. Мы делаем это при помощи микротестов.

Запустите тестирование и ответьте на максимум вопросов. Там, где варианты ответов представлены квадратными областями, можно выбирать несколько вариантов ответов.

Внимание, этот тест не является средством контроля. Он просто освежает материал в памяти и позволяет вспомнить наиболее важные моменты. После теста можно посмотреть все свои правильные и неправильные ответы.

Лабораторная работа

1. Откройте в текстовом редакторе **index.html**
2. Вставьте нужные HTML-элементы: html, head, title, body, h1-h4, p

Структура страницы

1. HTML-элементы
2. атрибуты
3. корректность структуры

Валидация страницы

1. https://validator.w3.org/

Валидация html-страницы - это проверка написанного кода на соответствие правилам языка html. Всегда можно проверить свою или чужую страницу, нужно только указать адрес страницы или отправить код страницы в форму по адресу https://validator.w3.org/

Если получили подтверждение правильности, поздравляю! Ваш код валиден! Иначе, будет показан перечень ошибок с указанием строк, на которых они произошли.

Семантика

Семантика — раздел лингвистики, изучающий смысловое значение единиц языка

Html-элементы и их атрибуты всегда имеют какое-то смысловое предназначение. Например, создание акцента на тексте или выделение важного фрагмента текста как заголовка. Смысловое предназначение элементов именуется семантикой.

Нарушением семантики может быть пример, когда для создания крупного текста разработчик выделяет его элементами `h1` или `h3`. Ещё одним примером является многократное использование разрыва строки `br` для создание большого отступа. Это неправильно.

Нарушение семантики приводит к плохому восприятию страницы поисковыми системами, логическим ошибкам, непониманию между разработчиками, непрофессионализму.

Глобальные атрибуты

```
<p contenteditable="true">c атрибутом contenteditable</p>
<p class="some" >c атрибутом class</p>
<p id="message">c атрибутом id</p>
<p style="color:green">c атрибутом</p>
<p title="...">c атрибутом</p>
```

Часть html-атрибутов называются глобальными. Они могут размещаться практически во всех элементах.

Поговорим о самых распространённых глобальных атрибутах. Атрибут `class` позволяет указать произвольное обозначение для элемента. По этому обозначению потом можно менять оформление всех элементов, принадлежащих классу, который вы ввели.

Ещё раз закрепим. Атрибут `class` - это зарезервированный глобальный атрибут. Его можно вставлять в любой элемент. Значение атрибута - это произвольные обозначения, обычно слова на английском языке. В дальнейшем мы сможем изменить фон всех элементов определенного класса или назначить любые другие CSS (Каскадные таблицы стилей) оформления.

Один из атрибутов, о которых мы поговорим всего один раз на этом курсе - атрибут `contenteditable`. Если вы укажите его в html-элементе, то можно будет менять содержимое этого элемента сразу в браузере!

```
<p class="some">c атрибутом class</p>
<p contenteditable="true">c атрибутом contenteditable</p>
```

Следующий атрибут используется для создания идентификатора элемента на странице. То есть его значение не должно повторяться у других элементов. Обозначается как `id` и содержит произвольный набор алфавитно- цифровых символов. Обычно там размешаются слова на английском.

Как и атрибут `class` , идентификатор помогает работать с CSS. Но ещё он может использоваться при работе с гиперссылками. Если вы захотите ссылаться на определенную часть документа, создайте идентификатор у нужных html-элементов

```
<p id="message">c атрибутом id</p>
```

Один из способов задать оформление элементам через CSS - размещение атрибута `style`. Значение атрибута записывается как пары свойств и значений. Например, CSS свойство `color` отвечает за цвет текста элемента, а его значение `green` указывает на зелёный цвет.

Вначале, некоторые разработчики путают html-атрибуты и CSS-свойства. Вы должны их строго различать.

```
<p style="color:green">c атрибутом</p>
```

Атрибут `title` содержит текст будущей всплывающей подсказки. При наведении на элемент с таким атрибутом, появляется однострочная подсказка. Это помогает посетителям сайта разобраться что к чему.

```
<p title="...">c атрибутом</p>
```

К сожалению, нельзя просто поменять оформление этой подсказки.

Группирующий элемент div

```
<div>текст в контейнере</div>
<div class="some">текст в контейнере</div>
<div title="..." id="d">текст в контейнере</div>
```

Для группировки наборов HTML-элементов может использоваться элемент `div`. Его иногда называют контейнером. Добавляя контейнеру классы или индентификаторы, мы можем обращаться к этим элементам и оформлять их при помощи CSS

Например, чтобы создать блок с четырьмя колонками, можно создать структуру типа

```
 <div class="cols">
   <div class="col">...</div>
   <div class="col">...</div>
   <div class="col">...</div>
   <div class="col">...</div>
 </div>
```

Классы `cols` и `col` выбраны для соответствия словам **колонки** и **колонка**. Можно использовать другие названия.

После добавления стилей, можно изменить ширину и расположение колонок.

Структура до HTML5

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html lang="en">
  <head>
    <meta http-equiv="Content-Type"
       content="text/html;charset=UTF-8" >
    <title>Document</title>
  </head>
  <body>
    <div id="container">
      <div id="header">"Шапка" страницы</div>
      <div id="nav"> Меню </div>
      <div id="section">
        <h1>Заголовок</h1>
        <p>текст текст текст текст текст</p>
      </div>
      <div id="footer">"Подвал" страницы</div>
    </div>
  </body>
</html>
```

До появления HTML5, классы или идентификаторы элементов `div` записывались обозначения типа `header` , `nav` , `footer` , `section` , `aside` .

`header` обозначал верхнюю часть страницы или блока. `nav` - навигационное меню, `section` - секцию, `aside` - выноску. Некоторые и сейчас используют такие классы и идентификаторы

Секционные/структурные элементы

`article`

`section`

`address`

`nav`

`aside`

`header`

`footer`

о секционных...

В HTML5 создали блочные элементы с такими же названиями. `article` - статья или заметка, `section` - секция, `nav` - навигация и так далее. Работать стало удобней. Мы открываем код какой-нибудь страницы и понимаем, что `header` - это верхний фрагмент блока. Возможно вы спросите что именно улучшилось, кроме краткого написания? При чтении страницы, поисковые роботы могут догадаться о том, что за элементы перед ними. А с `div` и классом `nav` сделать это было бы трудней. Кроме того, разработчику удобней находить нужные элементы, когда понятно обозначение элемента

Структура в HTML5

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Document</title>
  </head>
  <body>
    <div class="container">
      <header>"Шапка" страницы</header>
      <nav>Меню</nav>
      <section>
        <h1>Заголовок</h1>
        <p>текст текст текст текст текст</p>
      </section>
      <footer>"Подвал" страницы</footer>
    </div>
  </body>
</html>
```

Теперь типичная HTML5-страница может выглядеть так: общий контейнер с классом или идентификатором, который имеет название `container` или `wrapper`. Внутри элементы `header`, `nav`, `section`, `footer`, `div`

Лабораторная работа

1. Откройте в текстовом редакторе `index.html`
2. Добавьте секционные элементы `header`, `nav`, `div`, `footer`
3. Сохраните изменения `index.html`
4. Посмотрите результат в браузере (не должно быть явных изменений)
5. Проверьте HTML-код страницы на валидаторе
6. Исправьте ошибки

Гиперссылки

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Document</title>
  </head>
  <body>
    <a href="#metka">Ссылка на метку с id (внутренняя)</a>
    <a href="drugaya.html#label">Ссылка на другую страницу</a>
    <a href="http://www.specialist.ru">Внешняя ссылка</a>
    <p>Lorem ipsum dolor sit amet, consectetur adipisicing.</p>
    <p id="metka">Eligendi optio nihil vel nobis.</p>
  </body>
</html>
```

Гиперссылки связывают наши документы с другими документами или ресурсами. Они могут вести с одной страницы нашего сайта, на другую, а могут отправлять посетителей на другой сайт. Могут отправлять пользователей на архивы или запускать почтовый сервис при нажатии на ссылку с емейлом.

Для создании гиперссылки нужно записать html элемент `a`. Его название сформировано от слова `anchor`, то есть якорь. Морская тематика иногда просматривается в HTML, смайл. Далее нужно указать куда будет вести гиперссылка - заполняется атрибут `href`.

Если атрибут `href` будет начинаться с решетки, то при нажатии на гипертекст браузер будет переносить нас к фрагменту документа, где в HTML элементе указан идентификатор с меткой. Если метка совпадает со значением после решетки. Давайте посмотрим на практический пример.

Если в атрибуте `href` указано название документа, например *drugaya.html*, то при нажатии на гипертекст браузер отправит нас на этот документ. Правда если название документа записано неправильно, то появится неприятное сообщение, что ресурс не найден. Проблема решается переименованием. Или созданием документа, если его не существовало.

В случае когда нужно перейти на внешний сайт, `href` заполняется адресом этого внешнего сайта. Например, https://vk.com

Адресация

```
<a href="http://htmllab.ru/test-html">Абсолютная ссылка</a>
<a href="ftp://site.com/test.zip">Абсолютная ссылка2</a>
<a href="mailto:some@some.some">Ссылка с псевдопротоколом</a>
<a href="javascript:alert(1)">Ссылка с псевдопротоколом 2</a>
<a href="page.html">Относительная ссылка 1</a>
<a href="dir/page.html">Относительная ссылка 2</a>
<a href="../page.html">Относительная ссылка 3</a>
<a href="../../page.html">Относительная ссылка 4</a>
<a href="../dir/page.html">Относительная ссылка 5</a>
<a href="/page.html">Ссылка от корневой папки сервера</a>
```

Если на вашем сайте рядом с документом есть папка в которую нужно сослаться, в атрибуте `href` название папки предваряет название документа. Например, нужно из `index.html` перейти в папку `folder` к странице `catalog.html`. В этом случае адрес атрибута `href` будет строится так: `folder` слеш `catalog.html`

Если нужно создать ссылку, ведущую из `catalog.html` на `index.html`, то в `href` перед названием файла должны стоять две точки и слеш. Будьте аккуратны, нужно ставить не двоеточие, а две точки. Это означает переход на один уровень выше по файловой структуре.

Отлично, осталось узнать как организовать работу с гиперссылками содержащими емейлы. Всё просто, напишите в атрибуте `href` псевдопротокол `mailto` двоеточие и адрес электронной почты. Готово! Если повезёт, у пользователя откроется почтовый клиент или почтовый сервис. Но может и не повезти, если на клиентской машине ничего не настроено. Рекомендуется и сам гипертекст указать электронным адресом, тогда если почтовый клиент не настроен, пользователь сможет скопировать адрес вручную.

Примечание. Псевдопротокол - это специфическая запись, которая не является полноценным протоколом. Но она иногда помогает что -то сделать. Например, кроме псевдопротокола `mailto` есть псевдопротокол `javascript`. Если написать в адресной строке браузера или в атрибуте `href` что-то типа javascript двоеточие alert(124), то появится кое-что - 124.

Лабораторная работа

1. Откройте в текстовом редакторе `index.html`
2. Создайте ссылки внутри `nav`
3. Создайте ссылку с псевдопротоколом `mailto` для email
4. Создайте внешнюю ссылку на сайт https://www.specialist.ru
5. Запустить страницу `index.html` в браузере и убедитесь, что она правильно работает
6. Убедитесь в правильной работе ссылок на странице

Практическая посвящена созданию гиперссылок и связыванию страниц сайта друг с другом и с внешним сайтом.

Откройте текстовом редакторе файл `index.html`. Создайте навигационное меню так, чтобы с главной страницы можно было перейти на любые страницы вашего сайта. Предполагается, что их сейчас не более 5, иначе возникнут сложности. Вообще всё можно сделать проще при помощи скриптовых языков программирования. Но это другая история.

Создайте гиперссылку с псевдопротоколом `mailto`. Гипертекст этой ссылки тоже должен быть электронным адресом.

Убедитесь, что после создания гиперссылок, они работают как задумано. То есть они ведут на соответствующие файлы или запускается почтовый клиент в случае с почтой.

Мы работали с файлом `index.html`, следующим шагом нужно создать подобное меню в остальных страницах.

Тест

Пройти

Сделайте паузу. Как бы ни захватывало изучение HTML - нужно делать перерыва. Можно сделать гимнастику для глаз, потянуться, выпить воды, чая или кофе. И пройти его - наш тест на закрепление знаний по текущему модулю.

Основные понятия CSS

1. селектор `div`
2. свойство `color`
3. значение `navy`
4. декларация `color: navy;`
5. правило `div{color: navy; background: lightyellow}`

Каскадные таблицы стилей (Cascading Style Sheets) - это язык, который отвечает за визуальное представление документов.

Базовыми понятиями CSS являются: селектор, свойство, декларация и правило.

Селектор - это правило нахождения и оформления html элементов на странице. Можно найти элемент по названию тега, атрибута и других условий. Например, `p`, параграф, найдёт все параграфы в документе

CSS свойством называется характеристика, меняющая внешнеее представление найденных элементов.

Декларация - комбинация CSS свойства и значения, записанных через двоеточие и завершаюзаяся точкой с запятой. Например, color двоеточие navy. Цвет текста найденного элемент будет синим.

CSS правило, это набор деклараций

Способы определения стилей

1. Атрибут `style`: inline встраивание
2. Элемент `<style>`: embedding вложение
3. Элемент `<link>`: linking подключение

Кроме встраивание стилей через атрибут `style`, можно описывать правила в элементе `style`. Этот элемент обычно находится в разделе `head` . Как раз тут мы и можем указать селектор, потом декларации окружённые фигурными скобками. Скобки указывают какие именно декларации будут применяться для текущего селектора.

Также можно разместить стили в отдельном файле. Такому файлу дают название типа `style.css` и заполняют селекторами, правилами. Никаких html элементов в подобном файле не присутствует. Файл подключается через элемент `link` в разделе `head`. Этот элемент `link` состоит из одного тега и содержит обязательные атрибуты `rel` и `href`. `rel` будет у нас указывать значение `stylesheet`, так мы говорим что работаем с стилями. `href` указывает на стилевой файл

Стили через style (inline)

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Document</title>
  </head>
  <body>
   <p style="color: navy; background:lightyellow">Lorem ipsum</p>
  </body>
</html>
```

Ещё раз вспомним, как указываются стили через атрибут `style`. В параграфе, в кавычках атрибута `style` записываем `color:navy` и текст параграфа. Если запустить страницу в браузере, можно увидеть синий текст.

Стили через <style> (embedding)

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Document</title>
    <style>
      p{color: navy; background:lightyellow}
    </style>
  </head>
  <body>
   <p>Lorem ipsum dolor sit amet</p>
  </body>
</html>
```

В `head` создаём элемент `style` и указываем то же свойство, но после селектора `p`. И не забываем это свойство указать внутри фигурных скобок.

Стили через <link> (linking)

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Document</title>
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <p>Lorem ipsum dolor sit amet</p>
  </body>
</html>
```

Через элемент `link` можно подключить один или более стилевых файлов на страницу. Когда CSS-кода становится много или увеличивается посещаемость сайта, приходится объединять несколько CSS-файлов в один. Но пока, для нас это не проблема.

Каскадирование и наследование

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Каскадность</title>
    <!-- style.css: h1{color: green; background: yellow} -->
    <link rel="stylesheet" href="style.css" />
    <style> h1{ color: white; background: orange} </style>
  </head>
  <body>
    <h1 style="background:lightgreen">Белый на зелёном</h1>
  </body>
</html>
```

Аббревиатура CSS (Cascading Style Sheets) расшифровывается как каскадные таблицы стилей. Почему каскадные? Когда мы прописываем стили, они применяются последовательно от самых первых, до тех, что прописаны в атрибутах. Это и есть каскад.

Правда стили более важного нижележащего селектора могут перекрыть стили вышележащего. Но это не нарушает каскад - селекторы с одинаковыми приоритетами работают как и говорилось

Иногда разработчики указывают параметр, восклицательный знак important - `!important` для добавления приоритета некоторому свойству. Не стоит его использовать до того, как разберетесь с приоритетами.

В каждом браузере есть отладчик, который показывает какие стили были применены к элементу, а какие перекрыты другими стилями

Единицы измерения размера в CSS

Относительные

em - высота используемого элементом шрифта

ex - ширина буквы "x" используемого элементом шрифта

% - относительные значения(например +20%)

Абсолютные

in - inches, дюймы

px - pixels, пикселы

cm - centimeters, сантиметры

mm - millimeters, миллиметры

pt - points, пункты(1pt = 1/72in = 0,35mm)

pc - picas, пики(1pc = 12pt)

Различают относительные единицы измерения CSS, типа `em` или `%`, и абсолютные единицы типа `cm` - это сантиметры, `in` - дюймы. Часто используются `px` - пикселы, `rem` похожи на `em`, но отсчитывается от базового размера текста в браузере. По умолчанию, 16 пикселей.

Иногда используются градусы `deg`, секунды `s` и другие величины

Единицы измерения цвета в CSS

Название цвета: `red`, `orange`, `green`, `blue`
Шестнадцатеричный вид: `#00CC00`, `#0c0`
Формат RGB (Red Green Blue - красный зелёный синий): `rgb(0,240, 125)`, `rgb(40%, 20%, 80%)`
Формат RGBA (Red Green Blue Alpha- красный зелёный синий прозрачность): `rgba(255, 0,0, 0.5)`
`transparent`

Существует несколько вариантов указания цвета для нужных свойств. Во-первых, можно указать название цвет на английском языке - доступно более двухсот названий.

Можно указать один из почти семнадцати миллионов цветов в формате RGB (Red Green Blue - красный зелёный синий). Такой цвет начинается с символа решётки, а потом содержит три или шесть шестнадцатиричных чисел, с единицы по F (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F).

Если ничего не понятно по RGB (красный зелёный синий) и шестнадцатиричные, то либо повторите школьную информатику, либо просто воспользуйтесь выбором цвета в популярных текстовых редакторах. Удобно находить комбинации красного зеленого и синего запросом в поисковике. Яндекс и Гугл подсказывают выбор цвета

Цвет можно указывать в формате функционала. Когда записывается слово `rgb`, следом сразу в скобках через запятую идут три десятичных числа от 0 до 255, либо три числа в виде процентов для каждого компонента цвета. От нуля до ста процентов для каждого цвета. Не должно быть пробела между названием `rgb` и открывающейся круглой скобки.

Есть возможность указать цвета с полупрозрачностью. Для этого указывается функционал `rgba`. Последняя буква обозначает альфа-канал, это и есть полупрозрачность. Она задаётся числом от ноля до единицы. Например, 0.5 указывает полупрозрачность.

Цветовым свойствам можно задать значение `transparent` - прозрачный цвет

Селекторы: тега, класса, id

```
h1{ color: #369 } /* селектор тега */
.some { color: orange } /* селектор класса */
p.some { color: purple } /* комбинация селекторов */
#second { color: silver; } /* селектор идентификатора */
```

```
<h1>Заголовок</h1>
<p class="some">текст текст</p>
<p id="second">текст текст </p>
<div>
  <p>текст текст</p>
</div>
```

Пора узнать как использовать селекторы для нахождения и оформления элементов.

Селектор по тегу позволяет найти все элементы с таким же названием как у селектора. Например, чтобы найти все гиперссылки, нужно использовать селектор a. Для нахождения всех контейнеров - HTML-элементов `div`, нужно указать селектор `div`. При этом фигурные скобки не указываются.

Для указания селектора по классу и идентификатору, нужно указать название класса или идентификатора, предваряя само название символом решетки или точки. С решеткой мы уже сталкивались, потому легко запомнить её связь с идентификатором `#second`. Ещё раз, для нахождения и оформления всех элементов с определенным классом some, нужно указать селектор `.some`

Контекстный и дочерний селекторы

```
div p {color: grey}  /* контекстный селектор */
body > p {background: yellow;} /* дочерний */
```

```
<p class="some">текст текст текст текст текст </p>
<p id="second">текст текст текст текст текст </p>
<div>
  <p>текст текст текст текст текст </p>
</div>
```

Контекстный селектор - мощный инструмент нахождения элементов, вложенных в другие элементы. Если нужно найти все параграфы что входят в блоки `div`, то следует записать селектор как `div p`. Найдутся даже те параграфы, которые вложены глубоко в `div`. Например, если `div` содержит секцию с вложенным параграфом, то такой параграф тоже найдется.

Дочерними элементами называются элементы, которые находятся на первом уровне вложенности. Селектор дочернего элемента находит только элементы, непосредственно вложенные в другие. Когда мы записываем селектор `body`, угловая скобка, `p` , то находим все дочерние параграфы. В этом случае, если существуют ещё более вложенные параграфы, то они не находятся

Псевдоклассы для гиперссылок

```
a:link {color: orange}  /* непосещенная  */
a:visited {color: grey} /* посещенная  */
a:active {color: lime}  /* в момент нажатия */

a:hover, p:hover{ background: #ff9; } /* при наведении*/
```

Пример HTML-кода

```
<nav>
  <a href="#">Lorem.</a>
  <a href="#">Excepturi!</a>
  <a href="#">Tempora!</a>
</nav>

<div>
  <p>Lorem ipsum dolor sit amet.</p>
  <p>Esse a eius amet, quisquam?</p>
  <p>Quia id autem maxime impedit!</p>
  <p>Commodi est possimus ea quam!</p>
</div>
```

Чтобы изменить оформление гиперссылок на странице, достаточно указать селектор ссылки и указать стили. Возможно вы обращали внимание, что не нажатая ссылка всегда имеет синий цвет, а после нажатия становится фиолетовой. А ещё в момент нажатия она на мгновение краснеет, попробуйте нажать на ссылку и не отпускать кнопку мыши.

Чтобы поменять цвета, показываемые по умолчанию, нужно использовать псевдоклассы для гиперссылок. Псевдоклассами называются классы, которые не указываются в HTML-коде, но их указание среди селекторов позволяет менять визуальное представление элементов. Названия псевдоклассов предваряются двоеточием. В случае с гиперссылками, псевдокласс `:link` меняет цвет не нажатой гиперссылки, `:active` - нажатой ссылки, `:visited` - посещённой гипер ссылки.

Обратите внимание, при помощи псевдокласса `:visited` можно менять только цвет посещённой ссылки. Изменение других CSS свойств не приведёт к изменению оформления ссылки.

Ещё один псевдокласс, `:hover`, используется для оформления элемента при наведении на него мышью. Этот псевдокласс можно использовать не только с гиперссылками. Его можно применять с любыми HTML-элементами, например с блоками `div` или параграфами `p`

Псевдоклассы: first-, last-, nth-child

```
* {font-family: Arial} /*универсальный селектор*/

p:first-child { color: orange } /* первый дочерний  */
p:last-child { color: green }   /* последний дочерний  */
p:nth-child(2n+1) {background: #ff9} /* нечетные параграфы  */

/* первый при обратном отсчете */
p:nth-last-child(1) {
  background: yellow
}
```

Пример HTML-кода

```
<nav>
    <a href="#">Lorem.</a>
    <a href="#">Excepturi!</a>
    <a href="#">Tempora!</a>
  </nav>

<div>
  <p>Параграф первый дочерний в div</p>
  <p>Esse a eius amet, quisquam?</p>
  <p>Третий параграф в div</p>
  <p>Параграф последний дочерний в div</p>
</div>
```

Псевдоклассы завершающиеся на `child` оформляют дочерние элемента. То есть элементы, находящиеся в других элементах. Псевдокласс `:first-child` находит все вхождения дочерних элементов, где они являются самыми первыми в родительском элементе, а `:last-child` находит элементы при условии что они последние элементы.

Начинающие разработчики иногда с трудом привыкают, что если мы указываем селектор p двоеточие `:first-child`, то параграфы найдутся только при условии, что они являются первыми элементами других элементов. То есть, если в блоке `div` первым элементом идёт элемент `header`, а следом параграф, то такое параграф не будет первым дочерним элементом блока `div`. Конечно он останется первым параграфом в диве, он не первым элементом.

Селектор `:nth-child` указывает какой по счёту дочерний элемент нужно окрасить. Второй или пятый, просто укажите число. При указании формул типа `2n` или `2n+1` будут найдены чётные или нечётные элементы. Такую раскраску можно получить и в случае использования в скобках ключевых слов `odd` и `even`

Псевдоэлементы

```
* {font-family: Arial}

p::first-letter{ color: #f00;} /* первая буква */
p::first-line {background: #ff9;} /* первая строка */
p::before {content: 'Внимание! '} /* строка в начале p */
p::after {content: '...'} /* строка в конце p */
```

```
<p>Lorem ipsum dolor sit amet...</p>
```

Селекторы псевдоэлементов помогают оформить первую букву текст HTML-элемента, первую строку блочного элемента, создать области по умолчанию области в начале или в конце HTML элемента. Псевдоэлементы могут записываться и как псевдоклассы с двоеточием, но для избежания путаницы следует использовать два двоеточия - так правильней

Для оформления первой буквы элемента используйте псевдоэлемент `:first-letter`.

`:first-letter` поможет изменить оформление первой строки. Независимо от ширины экрана и объема текста, первая строка может иметь свой размер, цвет и любые другие CSS-свойства

Псевдоклассы `:after` и `:before` дают возможность указывать в CSS-правиле свойство `content`. В него можно поместить текст, функционал `attr` с названием атрибута или оставить свойство пустым. Дополняя правило другими CSS-свойствами, при помощи `:after` и `:before` можно создавать типовые картинки рядом с ссылками, отрисовывать разделители

Лабораторная работа

1. Откройте в текстовом редакторе `index.html`
2. Создайте класс `.lead` для параграфа после заголовка
3. Создайте в папке css файл style.css
4. Создайте в index.html элемент <link rel="stylesheet" href="" /> и в атрибуте `href` укажите путь к файлу `style.css`
5. Напишите селекторы и правила для оформления страницы
6. Проверьте CSS-код на ошибки

В этой практической работе нужно закрепить полученные навыки работы с CSS. Откройте в текстовом редакторе файл index.html и создайте класс для параграфа. Он может называться, например `lead`. Создайте в папке `css` файл `style.css` и подключите его через элемент `link`. Проверьте правильность указания пути к файлу. Напишите селекторы, правила и свойства так, чтобы страница приблизительно получила симпатичный вид

Может получиться так, что стили не работают, проверьте правильность именования стилевого файла и подключения в `link`. Проверьте правильность написать CSS-свойств и наличие символа ";" в конце каждой декларации. Проверьте корректность открытия и закрытия фигурных скобок CSS-правил. Проверьте CSS-код на ошибки на валидеторе CSS. Исправьте ошибки. Затем подключите созданный style.css ко всем страницам сайта и проверьте всю работу в целом

Свойства шрифтов

`font-family` - название
`font-size` - размер
`font-weight` - жирность
`font-style` - курсив
`font-variant` - начертание
`font` - сокращённое

font-family

Arial
Tahoma
`Monospace`
Times New Roman
Fantasy
В CSS есть целое семейство свойств, которые влияют на оформление. `font-family` отвечает за название шрифта или семейства. Если название шрифта состоит из нескольких слов, то название стоит помещать в кавычки. Часть шрифтов общеизвестны и достаточного указать название, шрифт будет использован. Другие шрифты нуждаются в подключении при помощи директивы `@font-face`

font-size

16px
24px
2em
1.5rem
50%
Свойство `font-size` отвечает за размер шрифта. Размер в `1em` в элементе `body` по умолчанию соответствует размеру `16px`. Но если размер текста в `body` станет больше, например `20px`, то для всех дочерних элементов `1em` будет соответствовать `20px`. Иногда применяется `rem` для постоянного отсчёта относительно базового размера шрифта браузера. У `font-size` есть и другие обозначения для определения размера шрифта

font-weight

normal
bold
`font-weight` отвечает за полужирное начертание шрифта. По эффекту это напоминает оформление элемента `strong`. Главное отличие в том, что это свойство никак не влияет на семантику документа. Самые часто используемые значения - `bold`, то есть полужирное начертание и `normal`, то есть обычное

font-style

normal
italic
За курсивное начертание отвечает свойство `font-style` . Значение `italic` создаёт эффект, `normal` - отменяет курсив.

font-variant

normal
SMALL-CAPS
Свойство `font-variant` помогает отобразить буквы в верхнем регистре, но маленьким начертанием. Для этого используется значение `small-caps`. Или `normal`, если хотите чтобы все было как прежде

font

italic 20px Arial
style variant weight size family
Свойство `font` объединяет шрифтовые оформления в краткую запись. Последовательно свойств лучше не перепутать. Учтите, минимально возможная запись должна содержать размер и название шрифта

Эксперимент

Эксперимент

Попробовать...

Свойства текста

`text-indent` - отступ первой строки
`word-spacing` - расстояние между слов
`letter-spacing` - расстояние между буквами
`line-height` - высота строки
`text-align` - выравнивание текста
`text-decoration` - декорирование текста
`text-shadow` - тень
`text-transform` - формат записи букв
`white-space` - управление переносами

Свойства оформления текста помогают работать с отступами и другими оформлениями текста

text-indent

0
10px
50px
100px

Свойства оформления текста помогают работать с отступами и другими оформлениями текста. `text-indent` задаёт отступ для первой строки блочного элемента.

`word-spacing` и `letter-spacing` указывают расстояние между словами и буквами соответственно. Их удобно задавать, когда дизайнер указал в макете нестандартные отступ и нам нужно их реализовать.

word-spacing

`normal`: text text text
`10px`: text text text
`30px`: text text text
`50px`: text text

letter-spacing

`normal`: text text
`10px`: t e x t t e x t
`30px`: t e x t t e x t
`50px`: texttext

line-height

`normal`
`2rem`
`50px`
`200%`

Высота строки задается свойством `line-height`. Важно помнить, что свойством определяется расстояние между базовыми линиями строк. Поэкспериментируйте с данным свойством.

text-align

`left`

`center`

`right`

`justify`

Выравнивание текста в блочном элементе можно выполнить с `text-align`. Свойство умеет выравнивать по левому и правому краям, по центру, и по ширине - значение `justify`

text-decoration

underline

none

overline

~~line-through~~

~~underline overline line-through~~

Декорирование текста подчеркиванием, зачеркиванием или надчеркиванием выполняется свойством `text-decoration`. Если нужно отменить подчеркивание, например у ссылок, используйте значение `none`.

text-shadow

`1px 1px 10px #369;`

`#FC0 1px 0 10px`

`5px 5px #558ABB`

`5px 10px`

`1px 1px 2px red, 0 0 1em blue, 0 0 0.2em blue;`

Свойство `text-shadow` создаёт тень у текста. Первые два числа определяют смещение тени по оси x и оси y. Точка отсчета - левый верхний угол содержимого. Третье число определяет размытие тени. Чем больше это число, тем менее резкой получается тень. Четвёртое значение - это цвет тени. В одном свойстве можно указать сразу несколько наборов теней, записывая их через запятую

text-transform

`none` - text text text
`capitalize` - Text Text Text
`uppercase` - TEXT TEXT TEXT
`lowercase` - text text text
`full-width` - text text text
Для создания заголовков или заполнении фамилии и имени в форме оплаты на сайте, используется отображение букв в верхнем регистре. `text-transform` помогает реализовать этот эффект, если будете использовать значение `uppercase`. Значение `capitalize` делает заглавными только первые буквы каждого слова. Значение `lowercase` наоборот отражает символы в строчном варианте, то есть в нижнем регистре

white-space

`normal` - пробелы объединяются в один пробел
`nowrap` - как `normal`, но не переносит строки
`pre` - последовательности пробелов сохраняются так, как они указаны в источнике. Строки переносятся только там, где в источнике указаны символы новой строки и там, где в источнике указаны элементы `<br/>`
`pre-wrap` - как `pre`, но строки переносятся при необходимости для заполнения строчных боксов
`pre-line` - последовательности пробелов объединяются в один пробел. Строки разбиваются по символам новой строки, по элементам `<br/>`, и при необходимости для заполнения строчных боксов..
Чтобы добиться эффекта, который создаёт элемент pre, можно воспользоваться свойством `white-space` со значением `pre`. У него есть ещё несколько специфических значений, по форматированию строк с разным учётом переносов

Свойства фона

`background-color` - цвет фона элемента

`background-image` - фоновое изображение

`background-position` - координаты фонового изображения

`background-attachment` - способ закрепления изображения

`background-repeat` - повторяемость изображения

`background-size` - размер фонового изображения

`background` - сокращённое свойство

`background-color background-image background-repeat background-attachment background-position`

background-color

`#336699` или `#369`

`rgba(133, 212, 80, 0.589)`

`transparent`

`lightyellow`

Для оформления фона страницы или произвольного HTML-элемента, используется `background-color`. Можно использовать и сокращённый вариант `background`, но тогда в краткой записи доступным станет и указание других свойств.

background-image

`url(../img/image.jpg)`

`none`

`inherit`

`linear-gradient(to right, #cfc, #369)`

`linear-gradient(180deg, #ffc, #f63)`

Для размещения фонового изображения используется background-image с функционалом `url`. В круглых скобках функционала указывается путь к графическому файлу в формате png, jpg или gif.

background-position

`inherit`

`40px 100px`

`top center`

`center`

`20% 100%`

Размещённые изображения всегда позиционируются от верхней левой точки отсчета. Ось y всегда направлена вниз. Сместить изображение на некоторое расстояние по каждой из осей позволяет `background-position`. Смещение может быть указано в пикселях, процентах или словами `top`, `bottom`, `center`, `left`, `right`

background-attachment

`scroll`

`fixed`

`local`

Фиксация изображения в фоне любого элемента относительно окна браузера выполняется свойством `background-attachment` со значением `fixed`. Значение по умолчанию у этого свойства - `scroll`, то есть изображение прокручивается вместе с элементом

background-repeat

`repeat-x`

`repeat-y`

`repeat`

`no-repeat`

`space`

`round`

По умолчанию, изображение выводится по принципу обоев - многократно повторяясь, если позволяют размеры элемента. Если вы не хотите многократно повторять изображение или выполнить повторение только по одной оси, используйте свойство `background-repeat` с одним из значений: `repeat`, `repeat-x`, `repeat-y`, `no-repeat`

space

Изображение повторяется в заданном направлении столько раз, сколько необходимо, чтобы покрыть большую часть области рисования фонового изображения, не обрезая изображение. Оставшееся незакрытое пространство равномерно распределено между изображениями. Первое и последнее изображения касаются края элемента. Значение CSS-свойства `background-position` игнорируется для рассматриваемого направления, за исключением случаев, когда отдельное изображение больше области рисования фоновго изображения, что является единственным случаем, когда изображение может быть обрезано, когда используется значение `space`.

round

Изображение повторяется в заданном направлении столько раз, сколько необходимо, чтобы покрыть большую часть области рисования фонового изображения, не обрезая изображение. Если оно не покрывает точно область, плитки изменяются в этом направлении, чтобы соответствовать ей.

background-size

Размер фонового изображения меняется при помощи `background-size` . Кроме процентов и пикселей тут можно применить ключевые слова `cover`, `contain`. Посмотрите чем отличается их поведение!

Пример CSS-кода

```
color: #369     любой допустимый формат
background-color: #ff9  любой допустимый формат
background-image: url(img.png)  none, url()
background-position: top
top, bottom, center, left, right, величина, %
background-attachment: fixed
fixed, scroll
background-repeat: no-repeat
repeat, repeat-x, repeat-y, no-repeat
background-size: 100%
величина, %, cover, contain
background: #ff9  url(img.png) no-repeat fixed top
background-color background-image background-repeabackground-attachment background-position
```

Лабораторная работа

1. Откройте файл style.css в папке `css`
2. Добавьте шрифтовое оформление заголовкам, параграфам и ссылкам
3. Проверьте работу в браузере

В этой практической работе нужно применить к содержимому своей страницы свойства шрифтовых и текстовых оформлений. Постарайтесь больше экспериментировать со свойствами. Комбинируйте их и наблюдайте за изменением внешнего вида страницы.

Тест

Пройти!

Наступил долгожданный момент закрепления полученных знаний! Давайте отдохнём, нажав на ссылку с тестированием в конспекте. После прохождения теста, проведите работу над ошибками, изучив неправильные ответы . Вполне возможно, что все ответы будут верными!

Форматы изображений

Растровые: png, jpg, gif
Векторные: svg

Редко какая HTML-страница обходится без изображений. Фотографии товаров, в каталог интернет-магазина, логотипы курсов в учебном центре, аватарки пользователей - всё это примеры работы HTML-элемента `img`.

Можно подключать как растровые изображения в форматах png, jpg, gif, так и векторную масштабируемую графику svg.

PNG

PNG (portable network graphics) — растровый формат хранения графической информации, использующий сжатие без потерь по алгоритму Deflate. PNG был создан как свободный формат для замены GIF (рекурсивный акроним «PNG is Not GIF», PNG — не GIF).

.png
полутоновые изображения - глубина цвета 16 бит
цветное индексированное изображение - палитра 8 бит и глубина цвета 24 бит
полноцветное изображение - глубина цвета 48 бит

JPEG

JPEG (Joint Photographic Experts Group, по названию организации-разработчика) — один из популярных растровых графических форматов, применяемый для хранения фотоизображений и подобных им изображений.

.jpg, .jpeg
24 бит/пиксель

GIF

GIF (Graphics Interchange Format) — «формат для обмена изображениями») — популярный растровый формат графических изображений

.gif
8 бит/пиксель = 256 цветов
анимация

SVG

SVG (Scalable Vector Graphics — масштабируемая векторная графика) — язык разметки масштабируемой векторной графики, созданный Консорциумом Всемирной паутины (W3C) и входящий в подмножество расширяемого языка разметки XML, предназначен для описания двумерной векторной и смешанной векторно/растровой графики в формате XML.

.svg
масштабируемость
анимация

Встраивание изображений

```
<img
   src="img.png"
   alt="Текст для img.png"
   width="220"
   height="170"/>

 <a href="index.html"><!-- Ссылка-картинка -->
   <img src="logo.gif" alt="Лого" />
 </a>
```

У элемента `img` есть два обязательных атрибута - `src` - который содержит путь к изображению и `alt` - описывающий альтернативный текст. То есть текст, который покажется если изображение не будет доступно.

src

Атрибут `src` должен присутствовать и содержать действительный непустой URL-адрес, ссылающийся на ресурс изображение

alt

Если не указано иное, атрибут `alt` должен быть указан, а его значение не должно быть пустым; значение должно быть соответствующей функциональной заменой изображения. Конкретные требования к содержимому атрибута `alt` зависят от функции изображения на странице, как описано в следующих разделах.

Чтобы определить подходящую текстовую альтернативу, важно подумать о том, почему изображение включается в страницу. Какова его цель? Мышление, подобное этому, поможет вам понять, что важно для имиджа для целевой аудитории. Каждое изображение имеет причину для того, чтобы быть на странице, потому что оно предоставляет полезную информацию, выполняет функцию, маркирует интерактивный элемент, улучшает эстетику или является чисто декоративным. Поэтому, зная, для чего изображение, облегчает написание подходящей текстовой альтернативы.

source

Элемент `picture` и элемент `source` вместе с атрибутом `media` могут использоваться для предоставления нескольких изображений, которые изменяют содержимое изображения (например, меньшее изображение может быть обрезанной версией большего изображения).

```
<picture>
  <source media="(min-width: 45em)" srcset="large.jpg">
  <source media="(min-width: 32em)" srcset="med.jpg">
  <img src="small.jpg" alt="The wolf runs throuthe snow."gt;
</picture>
```

Пользовательский агент выберет первый элемент источника, для которого совпадает медиа-запрос в атрибуте `media`, а затем выберет соответствующий URL-адрес из своего атрибута `srcset`.

Размер изображения зависит от того, какой ресурс выбран. Чтобы указать размеры, которые пользовательский агент может использовать перед загрузкой изображения, можно использовать CSS.

```
img { width: 300px; height: 300px }
@media (min-width: 32em) { img { width: 500px; height:300px } }
@media (min-width: 45em) { img { width: 700px; height:400px } }
```

Элемент figure

```
<figure>
  <figcaption>Заголовок для img</figcaption>
  <img
    src="img.png"
    alt="Текст для img.png" />
</figure>
```

Чтобы с изображением ассоциировался видимый пользователю текст, используется HTML-элемент `figure`. Он может содержать не только фотографии, но и графики, диаграммы, схемы. Внутри `figure` рядом с изображением помещается подпись к изображению в элементе `figcaption`.

Элемент figure

Элемент `figure` представляет некоторый контент потока, необязательно с заголовком, который является автономным и на него обычно ссылаются как на единое целое из основного потока документа.

Таким образом, этот элемент может использоваться для аннотирования иллюстраций, диаграмм, фотографий, списков кодов и т. Д., На которые ссылаются из основного содержимого документа, но которые можно, не затрагивая поток документа, удалить из этого основного содержимого. например, на боковой части страницы, на выделенных страницах или в приложении.

Дочерний элемент `figcaption` элемента, если таковой имеется, представляет заголовок содержимого элемента `figure`. Если дочерний элемент `figcaption` отсутствует, то заголовок отсутствует.

Пример figure с кодом

```
<figure>
  <figcaption>Листинг 1. Объявление переменных.</figcaption>
  <pre><code>
    let S = 2e6;
    let p = 10;
    let n = 10;
  </code></pre>
</figure>
```

Листинг 1. Объявление переменных.

```
let S = 2e6;
let p = 10;
let n = 10;
```

Лабораторная работа

1. В редакторе откройте файл index.html
2. Добавьте изображения из папки `img`
3. Проверьте результат на валидаторе

Практическая работа посвящена встраиванию изображений в страницу. Будьте внимательны, речь идёт не о CSS-свойстве `background-image`, а об элементе `img`. Для выполнения работы, нужно открыть в текстовом редакторе файл `index.html` и встроить изображения так, чтобы результат совпадал с вариантом в конспекте. После выполнения работы убедитесь что код валиден - нужно проверить код на валидаторе

Тест

Пройти!

Долгожданная минута отдыха с нашими закрепляющими тестами! Попробуйте освежить в памяти изученный материал и потыкать в интерфейс теста Google Forms. Если будут обнаружены пробелы в знаниях, посмотрите верные ответы или спросить у преподавателя почему ответ не засчитан.

Карта изображений

```
<img src="shapes.png"
      usemap="#shapes" alt="четыре изображения.">

<map name="shapes">
  <area shape=rect coords="50,50,100,100">

  <area shape=rect coords="25,25,125,125"
  href="red.html" alt="Red box.">

  <area shape=circle coords="200,75,50"
   href="green.html" alt="Green circle.">

  <area shape=poly coords="325,25,262,125,388,125"
href="blue.html"  alt="Blue triangle.">
  <area shape=poly coords="450,25,435,60,400,75,435,90,450,
125,465,90,500,75,465,60" href="yellow.html" alt="Yellow star."> </map>
```

Элемент map

Элемент `map` (карта) содержит элементы `area` (область) с указанием координат областей и ссылок на нужые файлы - при нажатии на область, происходит переход по адресу в атрибуте `href`

Для связи изображения с картой, нужно использовать атрибут `usemap` в элементе `img`

Элемент area

Атрибут `shape` (форма) указывает на форму области. Существует четыре вида областей `circle` (круг), `default`, `poly` (ломаная) и `rect` (прямоугольник).

`circle` (круг)
В состоянии круга у элементов области должен присутствовать атрибут координаты с тремя целыми числами, последнее из которых должно быть неотрицательным. Первое целое число должно быть расстоянием в пикселях CSS от левого края изображения до центра круга, второе целое число должно быть расстоянием в пикселях CSS от верхнего края изображения до центра круга, и третье целое число должно быть радиусом круга, опять же в пикселях CSS.

`default`
В состоянии состояния по умолчанию у элементов области не должно быть атрибута cords. (Площадь всего изображения.)

`poly` (ломаная)
В состоянии многоугольника элементы области должны иметь атрибут согласования с по крайней мере шестью целыми числами, а число целых должно быть четным. Каждая пара целых чисел должна представлять координату, заданную как расстояния слева и сверху изображения в пикселях CSS соответственно, и все координаты вместе должны представлять точки многоугольника по порядку.

`rect` (прямоугольник)
В состоянии прямоугольника элементы области должны иметь атрибут согласования с ровно четырьмя целыми числами, первое из которых должно быть меньше третьего, а второе - меньше четвертого. Четыре точки должны представлять соответственно расстояние от левого края изображения до левой стороны прямоугольника, расстояние от верхнего края до верхней стороны, расстояние от левого края до правой стороны и расстояние от верхнего края до нижней стороны, все в пикселях CSS.

Типы списков

Неупорядоченный `ul`

Упорядоченный `ol`

Список определений `dl`

```
<ul>
  <li>Неупорядоченный</li>
  <li>Упорядоченный</li>
  <li>Список определений</li>
</ul>
```

Неупорядоченный список

Неупорядоченный список `ul` создаёт маркеры в виде дисков, окружностей или квадратов. Можно задать и произвольное изображение через CSS-свойство list-style-image и функционал url. Внутри, как и OL, идут элементы списка li. Список UL часто используется для создания меню на сайте, в этом случае внутри Li прописывается гиперссылка, а в ссылке текст.

Элемент `ul` представляет собой список элементов, где порядок элементов не важен, то есть когда изменение порядка не приведет к существенному изменению значения документа.

Элементы списка `li` являются дочерними узлами элемента `ul`.

Элемент `li` представляет элемент списка. Если его родительским элементом является `ol` или `ul`, то этот элемент является элементом списка родительского элемента, как определено для этих элементов. В противном случае элемент списка не имеет определенного отношения, связанного со списком, с любым другим элементом `li`.

```
<ul>
    <li>пункт 1</li>
    <li>пункт 2

      <ul>
        <li>пункт 2.1</li>
        <li>пункт 2.2</li>
        <li>пункт 2.3</li>
      </ul>

    </li>
    <li>пункт 3</li>
  </ul>
```

Упорядоченный список

Элемент `ol` представляет список элементов, в которых элементы были упорядочены преднамеренно, так что изменение порядка изменило бы значение документа.

```
<ol>
  <li>пункт 1</li>
  <li>пункт 2</li>
  <li>пункт 3</li>
</ol>
```

Атрибуты `reversed` и `start` управляют отображением нумерации. `start` содержит целое число, с которго начнётся нумерация. Наличие атрибута `reversed` меняет порядок отображения символов упорядочивания

Следующий код создаст список из трёх элементов, нумерация которых начнётся с пяти:

```
<ol reversed start="5" >
  <li>пункт 1</li>
  <li>пункт 2</li>
  <li>пункт 3</li>
</ol>
```

5. пункт 1
4. пункт 2
3. пункт 3

Элемент `li` может содержать атрибут `value` с целочисленным значением, которое соответствует номеру или букве перед `li`

Атрибут `type` определяет внешний вид маркеров для упорядочивания. Возможные значения - единица, буквы `a` малое и `A` большое, большая и малая буквы `I` и `i`. Если интересно, попробуйте подставить значения и посмотреть что получится. А мы на курсе для изменения внешнего вида будем пользоваться CSS-свойством `list-style-type` со значениями `decimal`, `lower-alpha`, `upper-alpha`, `lower-roman`, `upper-roman`.

Список определений

```
<dl>
  <dt>термин 1</dt>
  <dd>определение термина 1</dd>
  <dt>термин 2</dt>
  <dd>определение термина 2</dd>
  <dt>термин 3</dt>
  <dd>определение термина 3</dd>
</dl>
```

Definition list или список определений, позволяет указать набор терминов и их расшифровку. Внутри элемента `dl` для каждого термина указывается элемент `dt`, а для каждого определения даётся расшифровка `dd`. Можно использовать для указания специфической терминологии на сайте, словаря, описания сотрудников компании или отдела и во всех случаях, где оправдано выделение терминов и описаний

CSS-свойства для списков

`list-style-type` - тип маркера
`list-style-image` - изображени
`list-style-position` - место маркера

Для оформления маркеров списка `list-style-type` может принимать значения `disc`, `circle` или `square`. Значение none отменит маркеры у списка - они пропадут. `list-style-image` содержать указание на небольшие изображения, `list-style-position` на расположение маркера. Значение `inset` - внутреннее, `outset` - внешнее

list-style-type

none
- disc
- circle
- square

5. decimal
06. decimal-leading-zero

vii. lower-roman
VIII. upper-roman
ι. lower-greek
j. lower-latin
ժԱ. armenian
ი. georgian
m. lower-alpha
n. upper-alpha
そ. hiragana
タ. katakana

list-style-image

none
url(img.png)

list-style-position

inset
outset

Лабораторная работа

1. В редакторе откройте файл index.html
2. Используя теги неупорядоченного списка <ul> оформите меню на странице (ссылки должны быть вложены в `li`)
3. Удалите маркеры у списка `list-style-type:none`

Практическая работа закрепляет навыки использования списков на странице Откройте в текстовом редакторе файл `index.html` и разместите списки так, что бы внешний вид страницы совпадал с изображением в конспекте. Проверьте код на валидаторе и убедитесь, что отсутствуют ошибки. Удалите у списка маркеры при помощи `list-style-type:none`

Тест

Пройти!

Теория хорошо, но после изучения она забывается. Потому давайте поскорей повторим, изученный по спискам материал, при помощи теста.

Таблицы в HTML

```
<table> <!-- таблица -->
  <tr> <!-- строка таблицы-->
    <th>Ячейка 1</th>  <!-- заголовочная ячейка-->
    <th>Ячейка 2</th>
  </tr>
  <tr>
    <td>Ячейка 3</td> <!-- ячейка -->
    <td>Ячейка 4</td>
  </tr>
</table>
```

Основные теги таблицы

`table` - элемент таблицы

`tr` - строка таблицы

`td` - ячейка строки

`th` - заголовочная ячейка

Для работы с таблицами используются HTML-элементы: `table` - именно он содержит все другие табличные элементы, `tr` - table row , строка таблицы, `td` - table data, данные таблицы в виде ячейки. Строки таблицы обязаны размещаться в табличном элементе, а ячейки обязательно в строках. При этом разработчик должен сам контролировать вложенность и количество элементов.

Если нужен заголовок для таблицы - используйте элемент `caption`, обычно он размещается сразу после открывающего тега `table`

Для создания ячеек с заголовками столбцов, используется элемент `th`. Он оформляет текст ячейки полужирным начертаниям и центрирует его относительно ячейки.

Элементы `tr`, `td`, `th` могут не иметь закрывающих тегов - это разрешается правилами языка. Каждый последующий открывающийся элемент на автомате закрывает предыдущий.

Рекомендация. Располагайте теги строк и ячеек с новых строк, это упростит размещение данных и чтение исходного кода таблицы.

Исторически сложилось так, что некоторые веб-разработчики используют таблицы в HTML как способ управления макетом своей страницы, что затрудняет извлечение табличных данных из таких документов.

Таблицы и CSS

`border` - рамка, граница
`border-spacing` - расстояние между ячеек
`width` - ширина нестрочного элемента
`height` - высота нестрочного элемента
`border-collapse` - режим отображения рамок
`padding` - внутренний отступ

Стандартная таблица не отображает границы ячеек и строк. Потому лучше для селекторов `td`, `th` указать свойство оформляющего границу, его называют рамкой или бордюром. `border` указывает на толщину рамки, ее стиль и цвет. Можно также указывать отдельные значения у свойств `border-width`, `border-style`, `border-color`. С толщиной и цветом все понятно, что насчёт стиля? В CSS есть одинарная `solid`, двойная `double`, точечная `dotted`, пунктирная `dashed` и другие стили рамок. Выберите свои рамки!

Свойство `border-spacing` помогает указать расстояние между рамками по горизонтали и вертикали, а `border-collapse` со значением `collapse` - схлопывает соседние рамки в одну линию. Неплохой вариант для получения аккуратных таблиц.

Пример применения свойств

```
table{
  border: 1px solid #000;
  border-spacing: 2px 6px;
  width: 30%;
  height: 200px;
  border-collapse: separate;
  }
td, th {
  border: 1px solid #888;
  padding: 10px;
}
```

border

`border-width` - 3px, 1px, 5px
`border-style`
`solid` - сплошная
`dotted` - точечная
`dashed` - пунктирная
`double` - двойная
`border-color` - orange, #369, rgba(47, 200, 0, 0.6)

border-spacing

Свойство задаёт расстояние между ячейками, при несхлопнутых рамках

width

Свойство используется для задания ширины таблицы, в пикселях или процентах от ширины родительского элемента

height

Используется для задания высоты таблицы

border-collapse

По умолчанию имеет значениес `separate` - несхлопнутые рамки ячеек, второе значение - `collapse`, схлопывает рамки

padding

Отвечает за внутренние отступы ячеек или заголовка `caption`. В последующих темах это свойство разберём подробней.

Объединение ячеек

```
<table>
  <tr>
    <td colspan="2">1</td> <!-- по горизонтали -->
    <!-- <td>2</td> -->
    <td rowspan="2">3</td> <!-- по вертикали-->
  </tr>
  <tr>
    <td>4</td>
    <td>5</td>
    <!-- <td>6</td> -->
  </tr>
</table>
```

Атрибуты объединения

`colspan` - сколько ячеек объединяется по горизонтали

`rowspan` - сколько ячеек объединяется по вертикали

Для объединения нескольких ячеек таблицы, нужно выбрать самую левую верхнюю из прямоугольной области объединения. Внимание, можно объединить ячейки по вертикали, горизонтали, или двумя способами одновременно, главное нельзя объединять наборы ячеек которые не образуют прямоугольную область. Буквой Г нельзя. Z тоже нельзя.

В найденной левой верхней ячейке указывается атрибут `colspan` с количеством объединяемых по горизонтали ячеек и `rowspan` с количеством объединяемых ячеек по вертикали. Остальные ячейки просто удаляются или комментируются. Резонный вопрос: разве это объединение? Можно соглашаться или нет, но именно такое название носит этот алгоритм

Дополнительные теги таблицы

`caption` - заголовок таблицы

`tbody` - тело таблицы

`thead` - шапка таблицы

`tfoot` - подвал таблицы

Описание дополнительных тегов

Кроме указанных HTML-элементов существуют: элемент `thead`, который объединяет верхние строки таблицы, `tfoot` - объединяет нижние строки, то есть своеобразный подвал таблицы, `tbody` - объединяет основные строки таблицы. Последний, `tbody`, создаётся на странице в браузере даже если вы его не указали.

```
<table>
  <caption>Подпись</caption>
  <thead> <tr><td>Ячейка 1  </thead>
  <tbody> <tr><td>Ячейка 2  </tbody>
  <tfoot> <tr><td>Ячейка 3  </tfoot>
</table>
```

Ячейка 1
Ячейка 2
Ячейка 3

Подпись

Как и с элементами списков, в любую часть таблицы можно встроить теги вставки JavaScript-кода. За это отвечает элемент `script`. JavaScript является очень популярным языком программирования и используется на страницах, серверах, смартфонах и так далее.

Лабораторная работа

1. Откройте в текстовом редакторе файл pages/courses.html
2. Создайте таблицу, результат должен выглядеть как в конспекте

В этой практической работе мы закрепим работу с базовыми элементами таблицы - table, tr, th, td, caption и укажем стилевые свойства для задания ширины таблицы, фона отдельных ячеек, рамок границы. Откройте в текстовом редакторе файл pages/courses.html и создайте таблицу так, чтобы результат был похож на изображение верстки. После работы, исправьте ошибки, проверив разметку на валидаторе

Тест

Пройти!

Ура ура ура, ещё раз тест. Мы можем спокойно повторить материал и узнать какие вопросы по таблицам и спискам были усвоены хорошо, а какие нужно повторить. Не забывайте смотреть правильные ответы в конце тестирования! Они будут недоступны в зачетном тестировании, но сейчас эту возможность нужно использовать

История фреймов

Грустная

Основная идея фреймов - встраивание одного HTML-файла в другие. На практике это используются при размещении рекламы, работы с JavaScript. Вы же помните? Мы говорили об этом языке программирования. Такие социальные сети как вконтакте, фейсбук, сервисы типа codepen тоже используют фреймы.

Популярные видеохостинга типа YouTube предлагают код встраивания своих видео на вашу страницу при помощи элемента iframe

Встраиваемый фрейм

```
<iframe src="page1.html" name="metka"></iframe>
```

```
<a href="page1.html" target="metka">
  страница 1
</a>
<a href="page2.html" target="metka">
  страница 2
</a>>
```

Чтобы разместить один документ в рамках другого, нужно в элементе `iframe` указать атрибут `src` с указанием пути к первому документу. Попробуйте, вам может понравится.

Если у элемента `iframe` будет указано значение в атрибуте `name`, что гиперссылки смогут открывать содержимое прямо во фрейме, достаточно указать атрибут `target` гиперссылки совпадающим с атрибутом `name` элемента `iframe`

Применение iframe

вставка Youtube-роликов
Яндекс-карты
онлайн-редакторы
iframe-приложения

Youtube-ролики

```
<iframe  width="560" height="315"
  src="https://www.youtube.com/embed/2o8BcE0XhMY"
  frameborder="0" allowfullscreen ></iframe>
```

Зайдите на сервис YouTube, откройте понравившееся видео и покопавшись в настройках скопируйте HTML-код `iframe`, там будет присутствовать ссылка на видео.

Атрибуты `frameborder` и `allowscreen` скопированного кода не являются валидность в html5, но позволяют скрыть рамку и разрешить разворот содержимого на весь экран

Яндекс-карты

```
<iframe  width="320" height="250"
  src="https://api-maps.yandex.ru/frame/v1/-/CVh7YBYg?"
  frameborder="0" ></iframe>
```

Онлайн-редакторы

https://jsbin.com/
https://codepen.io
https://codesandbox.io
https://thimble.mozilla.org

iframe-приложения

вконтакте

Лабораторная работа

Откройте index.html

Разместите iframe с адресом:

```
... src="https://www.youtube.com/embed/FhTy0i_RhVk" ...
```

Сохраните файл и проверьте выполненную работу в браузере

В этой практической работе мы разместим iframe с видеороликом на нашей странице. Откройте в текстовом редакторе файл index.html. Рядом с ссылкой на изображение banner.jpg разместите iframe с кодом, а затем Сохраните файл и проверьте выполненную работу в браузере, результат должен выглядеть приблизительно как на изображении

Что такое HTML-формы

123

•••

вариант 1

вариант 2

вариант 3

или

или

Отправить Сбросить

HTML-формы- это элементы интерфейса, при помощи которых посетители сайта могут вводить и отправлять информацию. Решение куда и как отправлять принимает программист серверного языка программирования, типа PHP или JavaScript на Node.js

При помощи HTML-форм, верстаются формы регистрации пользователей, оставления комментариев на сайте, обратной связи, создания новых товаров в магазине

Элемент <form>

```
<form
   action="/"
   method="get"
   enctype="application/x-www-form-urlencoded"
>
  <!-- HTML-элементы формы -->
  </form>
```

Обычно все элементы формы помещаются в элемент `form`. Так мы подсказываем какой именно набор данных будет отправляться. У элемента `form` программисты проверяют или указывают атрибуты. Атрибут `action` - адрес документа, на который будет выполнен запрос. Если ничего не написать, форма отправится на ту же страницу, где расположена.

Атрибут `method` определяет метод отправки формы. Доверьте работу по выбору метода программистам. Но если хочется написать самостоятельно, то укажите методу значение `post`, тогда отправляемые данные не будут видны в адресной строке, а в дальнейшем из формы можно будет отправлять прикреплённые файлы. На текущем курсе можно не указывать эти атрибуты.

Элемент <input>

```
<input type="text" name="parameter_name" value="значение"/>
```

`type` - тип элемента ввода
`name` - название параметра для программиста
`value` - значение параметра для программиста

Давайте рассмотрим один из наиболее часто используемых элементов формы - HTML-элемент `input`. Его основная задача принимать данные от пользователя формы. Внешний вид и поведение HTML-элемента `input` меняются в зависимости от значений атрибута `type`

Элемент `input` состоит из единственного открывающего тега. По умолчанию, атрибут `type` имеет значение `text`, это означает создание простого однострочного текстового поля вода. Рядом указывается атрибут `name` с названием параметра, который отправится при отправке формы.

Чтобы лучше вникнуть в работу форм, сделайте паузу, потом создайте элемент `form` с методом `get` и внутрь поместите однострочное текстовое поле `input`. Убедитесь, что форма стала видна на странице. Введите в поле произвольный текст, или цифру и нажмите `Enter`. Когда форма отправится, в адресной строке браузера вы обнаружите название отправленного параметра и его значение

Для указания начального значения параметра в `input` указывается атрибут `value` с начальным значением. Именно это значение отправится из формы, если пользователь страницы не введёт свои данные в поле.

Типы элемента ввода

`text` - однострочное поле ввода
`password` - поле ввода пароля
`submit` - кнопка отправки формы
`reset` - кнопка сброса формы
`image` - кнопка отправки формы с изображением
`button` - кнопка для JavaScript-программиста

Поговорим от других типах элемента `input`. При написании атрибута `type` со значением `password`, поле ввода будет готово принимать данные, которые будут маскироваться точками в элементе `input`. Это удобно использовать для указания пароля

Значения `submit`, `reset`, `image` и `button` создают кнопку. `submit` - при нажатии на кнопку форма будет отправляться. `reset` - поля формы будут очищены, а если в `value` были заданы начальные значения, то они снова будут установлены. `image` - создаётся кнопка-изображение, но этом случае нужно указать путь к изображению в атрибуте `src`. Значение `button` создаёт кнопку, которая без JavaScript будет только нажиматься. Ее основное предназначение - связь с клиентским языком программирования.

Флажки и радиокнопки

```
<input type="checkbox" name="c1" value="a" checked />  вариант А

<input type="checkbox" name="c2" value="b" id="c2" />
  <label for="c2"> вариант Б</label>
```

```
<input type="radio" name="r" value="1" checked /> вариант 1
<input type="radio" name="r" value="2" /> вариант 2
<input type="radio" name="r" value="3" /> вариант 3
```

Часто в формах требуется предложить пользователю варианты выбора, и дать возможность указания единственного или нескольких значений. Это используется для голосований, выбора способа доставки, выбора дополнительных товаров.

Для предоставления пользователям множественного выбора, при помощи значения `checkbox` создаются флажки или чекбоксы. У набора чекбоксов должны быть разные значения атрибутов name и `value`. При этом текст, видимый пользователю вы размещаете вне чекбоксов - слева или справа.

Для предоставления выбора единственного значения из набора, используется значение `radio`. У радиокнопок значения name могут быть одинаковыми, ведь из группы радиокнопок отправится только значение одного элемента.

Атрибут checked

Атрибут `checked` указывает на включенность флажка или радиокнопки.

Элемент label

```
<label>
  <input type="checkbox" name="c2" value="b" /> вариант Б
</label>
```

```
<input type="checkbox" name="c2" value="b" id="c2" />
<label for="c2"> вариант Б</label>
```

Атрибут `checked` можно указать и у чекбоксов, и у радиокнопок

Для удобства разметки текст рядом с элементами ввода помещают в HTML-элемент `label`. Если в чекбоксе указать идентификатор, а в элементе `label` атрибут `for` с таким же значением, то при нажатии на текст, чекбокс будет реагировать на нажатия. Это же касается радиокнопок

Элемент select

```
<select name="course">
  <option value="1">HTML и CSS</option>
  <option value="2">Основы JavaScript</option>
  <option value="3">JavaScript в браузере</option>
</select>
```

Элемент `select` позволяет выбрать одно или несколько значений в вертикальном наборе вариантов. Иногда элемент называют выпадающим списком - по умолчанию пользователь должен выбрать нужную опцию в открывшемся наборе списка

Атрибуты select

`name` - название параметра для программиста
`size` - количество одновременно видимых `option`
`multiple` - разрешение множественного выбора с `Ctrl+Click`
Элемент `select` содержит атрибут `name` для отправки формы

Если в `select` нужно предоставить множественный выбор, то добавляется атрибут `multiple` и указывается количество отображаемых строк, например `size=4`.

Атрибуты option

`value` - значение для программиста
`selected` - признак выбора опции
В атрибутах `value` элементов `option` помещаются варианты значений для отправки формы. Элементы `option` можно не закрывать. Если у элемента `option` указывается атрибут `selected`, именно этот вариант будет опцией по умолчанию.

Элемент optgroup

```
<select name="city" size="4" multiple>
  <optgroup label="Верстальщик">
    <option value="1">HTML и CSS</option>
    <option value="2">HTML Расширенный</option>
  </optgroup>
  <optgroup label="JavaScript-программист">
    <option value="3">Основы JavaScript</option>
    <option value="4">Работа с DOM и события</option>
  </optgroup>
</select>
```

Наборы опцией можно объединять в группы. Это выполняется элементом `optgroup`. Атрибут `label` элемента `optgroup` описывает название группы в списке и не может быть выбран, как элементы `option`

Другие HTML-элементы

`textarea` - многострочная текстовая область
`input c type`:
`file` - поле для отправки файла
`hidden` - скрытое поле
`color` - выбор цвета
`range` - выбор диапазона
`number` - указание числа
`url` - указание url-адреса
`email` - указание email-адреса

Кроме рассмотренных элементов форм есть элементы для ввода текста, отправки файла, пересылка скрытых данных, выбора цвета и другие

Элемент `textarea` используется от отправки текстовых фрагментов. Между открывающим и закрывающим тегами указывается текст по умолчанию или удаляются любые символы. Атрибуты `rows` и `cols` указывают ширину и высоту элемента в символах. Будьте аккуратны, высота символов обычно больше ширины.

Элемент `input` с `type` равным `file`, предоставляет посетителям сайта выбрать файл для отправки через форму. Без программиста обойтись сложно, в форме разместить большого труда не составит

`input` со значением `type` равным `hidden` обозначает скрытое поле. Оно пересылается вместе с данными формы. Это поле используется когда нужно отправить данные, минуя отображение поля в браузере

В HTML существует большое количество значений `type`, порой не одинаково отрабатывающих в разных браузерах. `color` позволяет выбрать и отправить rgb-цвет, `email` - указать электронной почты, `url` - адрес страницы и так далее.

Пример HTML-кода

```
<textarea name="msg" cols="30" rows="5">Текст</textarea>
<!-- выбор файла -->
<input type="file" name="myfile"/>
<!-- скрытое поле -->
<input type="hidden" name="cs" value="fg&32kj3hk0"/>

<!-- другие элементы -->
<input type="color" name="color" />
<input type="range" name="range" />
<input type="number" name="number" />
<input type="url" name="url" />
<input type="email" name="email" />
```

Другие атрибуты полей формы

`placeholder` - подсказка
`required` - атрибут для обязательного заполнения поля
`pattern` - задания шаблона ввода
`disabled` - отключение поля

Атрибут `placeholder` указывает в поле ввода подсказку для пользователю, которая исчезает как только начинается ввод и появляется при очистке поля. Значение подсказки никуда не передаётся.

Атрибут `required` заставляет пользования заполнить поле в обязательном порядке. Атрибут `disabled` блокирует элемент, в котором находится.

`pattern` - один из очень интересных и сложных атрибутов. Содержит регулярное выражение или, другим словами, шаблон написания данных в текущее поле.

```
<input type="text" name="text1" placeholder="имя" />
<input type="text" name="text2" required />
<input type="text" name="text3" pattern="\d{3}"/>
<input type="text" name="text4" disabled />
```

Сервисы обработки форм

https://formspree.io/
https://formden.com
https://google.com/forms/

Многих начинающих разработчиков может огорчить невозможность обработки созданной формы без знаний серверных языков программирования. На помощь могут прийти сервисы, которые обработают вашу форму и выполнят необходимые действия, например отправляют полученные от посетителя данные на электронную почту

Лабораторная работа

1. Откройте в текстовом редакторе файл contacts.html
2. Создайте форму связи используя элементы форм. Форма должна иметь вид как на изображении в конспекте

Тест

Пройти!

Боксовая модель

бокс/блок
содержимое
рамка - border
внутренние поля - padding
внешние отступы - margin

Каждому HTML-элементу на странице соответствует прямоугольная область. В CSS есть боксовая или блоковая модель, которая определяет размеры прямоугольных областей, их расположение. У каждого подобного бокса есть внешние отступы. Они определяются CSS-свойством `margin`. Есть рамка, которая задаётся свойством `border`. Есть внутренние поля, которые задаются свойством `padding`. Внутренние поля, это расстояние между содержимым элемента и его рамкой.

Во внутренней области элемента располагается текст и HTML-элементы. Именно внутренней области мы задаём ширину и высоту свойствами `width` и `height`. Для внутренней области задаются фоновое изображение или цвет. Размер внутренней области можно задавать также свойствами `mix-width`, `max-width`, `min-height`, `max-height`. Эти свойства отвечают за минимальные и максимальные размеры ширины и высоты.

CSS-свойства боксовой модели

Поля (`margin`), границы (`border`) и отступы (`padding`) могут быть указаны отдельно как верхний, правый, нижний и левый фрагменты, каждый из которых может независимо управляться своим соответствующим свойством.

Свойства рамки

`border-top` - свойства для верхней части рамки

`border-left-width` - толщина левой части рамки

`border-bottom-style` - стиль нижней части рамки

`border-right-color` - цвет правой части рамки

`border-radius` - закругление рамки

border-radius

`10px` - закругление каждого угла

`10px 20px` - закругление левого верхнего и правого нижнего угла с радиусом загругления 10px, остальных - 20px

`10px 20px 30px` - закругление левого верхнего на 10px, правого верхнего и левого нижнего - 20px, правого нижнего на 30px

`10px 20px 30px 40px` - закругление углов начиная с левого верхнего и по часовой стрелке

`10px 20px 30px 40px/50px` - до слеша - закругление углов с радиусом по x, после слеша - с радиусом по y

Рамка задаётся свойством `border`. Указывается ширина рамки. Например, два пикселя. Потом указывает тип рамки. Например, `solid` -сплошная одинарная, `dotted` - точечная, `dashed` -пунктирная, `double`-двойная. После типа рамки, указывается её цвет. Цвет указывается любым вариантом: шестнадцатиричным числом, функционалом `rgb` или `rgba`, названием цвета.

Свойство `border-radius` задаёт закругление у рамок элемента. Можно задавать одно или несколько числовых значений.

Внутренние отступы

`padding-top` - верхний
`padding-bottom` - нижний
`padding-left` - левый
`padding-right` - правый
`padding` - можно задать с любой стороны

Размеры внутренних отступов задаются для каждой отдельной стороны свойствами `padding-top`, `padding-right`, `padding-bottom`, `padding-left`. Соответственно верхний, правый, нижний и левый отступ. Или сокращённым свойством `padding`. При использовании сокращённого свойства, указываются от одного до четырёх значений. Если у свойства `padding` вы указали одно значение, то внутренние отступы будут одинаковы со всех сторон. Если указано два числа, то первое число определит отступы сверху и снизу, а второе - слева и справа. Если задано три числа, то первое задаёт отступ сверху, второе число - отступы слева и справа, а третье число - отступ снизу. Если свойство padding задано четырьмя числами, то отступы задаются для четырех сторон, в следующей последовательности - сверху, справа, снизу, слева.

Значения свойств `padding` не могут быть отрицательными величинами.

Пример отступами

```
div{
  width: 50px; height: 50px;
  border: 1px solid #000;
  padding-top: 0;
  padding-bottom: 0;
  padding-left: 0;
  padding-right: 0;
}

.d1{
  padding: 10px; /* со всех сторон */
}

.d2{
  padding: 10px 20px; /* сверху и снизу - 10px, справа и слева -  20px */
}



.d3{
   padding: 10px 20px 30px; /* 10px сверху, 20px справа слева, 30px снизу */
 }

/* 10px сверху, 20px справа, 30px снизу, 50px слева */
.d4{
  padding: 10px 20px 30px 50px;
}
```

Поля

`margin-top` - верхнее и т.д.
могут быть отрицательными
проценты отсчитываются от ширины родителя
значение `auto` задаёт одинаковые поля

Внешние поля, `margin`, задают расстояние от рамки элемента до других рамок или их внешних полей. Задаются свойствами `margin-top`, `margin-right`, `margin-bottom`, `margin-left` или сокращённым свойством `margin`. `margin`, как и `padding` может иметь от одного до четырех значений. Можно указывать отрицательные значения.

Отступы элементов, находящихся друг над другом или вложенных друг в друга, могут накладываться. У элементов друг над другом это называется схлопыванием `margin` или коллапсом. У вложенных это называется утеканием `margin`

У `margin` есть важное значение, `auto`. Задание значения `auto` для блочного элемента означает что внешние поля будут одинаковы. Это используется для блочных элементов для их центрирования относительно родительского элемента. Например, такое свойство задаётся первому главному контейнеру в элементе `body`

Контейнер страницы

```
div{
  width: 50px; height: 50px;
  border: 1px solid #000;
  margin-top: 0;
  margin-bottom: 0;
  margin-left: 0;
  margin-right: 0;
}

.d1{ margin: 10px; /* со всех сторон */  }
.d2{ margin: 10px 20px; /* 10px сверху снизу, 20px справа слева */ }

/* 10px сверху, 20px справа слева , 30px снизу */
.d3{ margin: 10px 20px 30px;}

/* 10px сверху, 20px справа, 30px снизу, 50px слева */
.d4{ margin: 10px 20px 30px 50px; }

/* поровну справа и слева */
.d5{ margin: 0 auto; }
```

Лабораторная работа

1. Откройте в текстовом редакторе файл css/style.css
2. Очистите внешние и внутренние отступы у элементов меню: ul, li
3. Задайте классу контейнеру ширину 1200px и автоматический расчет внешних отступов
4. Проверьте CSS-код на валидность

Эта практическая работа ориентирована на базовое понимание работы с CSS-свойствами боксовой модели. Откройте в текстовом редакторе файл style.css и очистите внешние и внутренние отступы у элементов меню: ul, li.
Потом проверьте CSS-код на валидность и убедитесь, что произошли изменения в меню

CSS обтекание

1. float: left или right
2. clear: left, right или both

CSS-свойство `float` берет элемент из нормального потока и помещает вдоль левой или правой стороны его контейнера, где текст и встроенные элементы будут обтекать его. Удобно использовать для выравнивания по левой или по правой стороне изображений в тексте. Ещё это свойство используется для создания нескольких колонок на странице. Правда стараются применяют `display:flex` и `display:grid`, но понимание `float` важно для работы.

Если в родительском элементе находится плавающие элементы, то родитель не знает какую высоту должен иметь. Потому после дочерних плавающих элементов вставляется элемент отменяющий обтекание. Отмена происходит при помощи свойства `clear` со значением `both`, `left` или `right`. Свойство `clear` указывается у элемента, идущего после плавающих.

Лабораторная работа

1. Откройте в текстовом редакторе файл pages/contacts.html
2. При помощи CSS отредактируйте внешний вид формы связи (как в конспекте)

В этой практической мы вспомним свойства `text-align`, `margin` и `padding`, `float`. Откройте в текстовом редакторе файл `pages/contacts.html`. При помощи CSS отредактируйте форму связи так, чтобы ее нешний вид совпадал с представленным в конспекте. Небольшая подсказка, для удобства работы оберните элементы формы в блок с классом `row`.

Тест

Пройти!

Пора! Пройти тестирование по свойствам боксовой модели и плавающим свойствам! Жмём на кнопку и убеждаемся что наши ответы верны! Иначе подсматриваем где именно закралась ошибка

Практическая работа

Вёрстка...

Медиа-правило

`@media` - at-rule

`screen`

`print`

Обычно вначале мы исходим из предположения, что наша страница будет показываться в браузере на экране компьютера. Но как оформить страницу, которая будет распечатана? Или страницу, которая должна отображаться на экране телевизора? И как подойти к созданию мобильной версии страницы?

Медиа-запросы

Медиа-запросы полезны, когда вы хотите изменить свой сайт или приложение в зависимости от общего типа устройства (например, печать или экран) или конкретных характеристик и параметров (таких как разрешение экрана или ширина области просмотра браузера).

Медиа-запросы используются для следующего:

Для условного применения стилей с помощью CSS @media и @import at-rules
Для назначения определенных медиа для элементов <style>, <link>, <source> и других HTML с атрибутом media=

Оформление для печати

```
<link media="print"
   rel="stylesheet" href="print.css" />
```

```
@media print {
 body { font-size: 26px; }
}
```

link

Если вы хотите указать на то, что подключаемый стилевой файл относится к стилям для печати, в атрибуте media элемента link укажите значение print. Именно эти стили будут применены к странице, помимо стилей по умолчанию. Также можно указать у атрибута media значение screen, это пригодится для дальнейшей работы по созданию мобильной версии страницы.

Пояснения

```
<link media="print"
      rel="stylesheet" href="print.css" />
```

```
@media print {
  body { font-size: 26px; }
}
```

Лабораторная работа

1. Откройте в текстовом редакторе файл index.html
2. Создайте удобным способом стили для печати страницы

media queries

```
@media screen and (max-width: 760px){
  /* стили для ширины экрана не более 760px*/
}
```

Если нужно указать тип носителя в разделе `style` прямо в CSS, или непосредственно в стилевом файле, используйте CSS директиву `@media` с последующим указанием типа носителя. В последующих фигурных скобках будут находится селекторы и правила, содержащие свои фигурные скобки. Это может быть непривычным вариантом, избегайте ошибок с лишними и недостающими фигурными скобками.

Пример media-запросов

```
@media screen and (max-width: 760px){
    /* стили для ширины экрана не более 760px*/
  }
```

Медиа-свойства

`width,height` - ширина, высота видимой части браузера

`max-width/min-width` - максимальная и минимальная ширина

`orientation` - расположение окна

`aspect-ratio` - соотношение сторон

Примеры медиа-свойства

```
width, max-width, min-device-width, orientation

@media screen and (orientation: landscape){  /* стили */  }
@media screen and (orientation: portrait){  /* стили */  }

@media screen (max-width: 575px) { }
@media screen (min-width: 576px) and (max-width: 767px){ }
@media screen (min-width: 768px) and (max-width: 991px){ }
@media screen (min-width: 992px) and (max-width: 1199px){ }
@media screen (min-width: 1200px){ }
```

Лабораторная работа

1. Откройте в текстовом редакторе файлы index.html и style.css
2. При помощи CSS напишите стили для страницы, чтобы при ширине до 768px страница имела мобильный вид

В этой практической работе, мы создадим оформление страницы для печати. Именно такой ее увидит посетитель сайта, нажавший `Ctrl + P` и распечатавший страницу на листке бумаги. Откройте в текстовом редакторе файл `index.html` и Создайте удобным способом стили для печати страницы. Проверьте CSS-код на валидаторе. Примечание: запуск страницы на печать `Ctrl + P`

Тест

Пройти!

Указание мета-информации

```
<meta name="author" content="Имя Фамилия" />
<meta name="generator" content="WordPress" />
<meta name="yandex-verification" content=“"5a6d0f07d131af7e" />
<meta name="google-site-verification"
              content="EE1o68asN4-hXTgJztpODDbqSsO12SPFwRtJY2kUBYo" />
<meta name="Keywords" content="компьютерные курсы, компьютерное обучение, учебный центр"/>
<meta name="Description" content="Учебный центр "СПЕЦИАЛИСТ" при МГТУ им. Н.Э. Баумана. Более 1000 курсов для начинающих и
профессионалов. Международный сертификат, служба трудоустройства!"/>
<meta name="viewport" content="width=device-width, initial-scale=1">
<meta http-equiv="refresh" content="N;url=_адрес цели перенаправления_"/>
<meta name="robots" content="index, follow"/>
```

Яндекс о meta

Мета-информация для робота

Яндекс о роботах

Оформление курсора

[документация Mozilla]

```
p{ cursor: cell; }
p.cursor-img{
  cursor: url(http://placehold.it/50x50?text=HTML) 4 12, auto;
}
```

Аудио и видео

```
<audio src="audio.ogg" controls></audio>

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

<video src="video.mp4" controls></video>

<video controls width="220" height="160">
  <source src="video.mp4" type="audio/mp4" />
  <source src="video.ogv" type="video/ogg" />
</video>
```

XML и sitemaps.xml

карта сайта

```
<?xml version="1.0" encoding="UTF-8"?>
  <urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url>
    <loc>http://www.example.com/</loc>
    <lastmod>2005-01-01</lastmod>
    <changefreq>monthly</changefreq>
    <priority>0.8</priority>
  </url>
</urlset>
```

Сайт в сети

доменное имя

регистратор доменных имён

хостинг

FTP

адрес
логин
пароль

Организация рабочего места

Apache, MySQL, PHP

WAMP/LAMP/MAMP

https://ospanel.io/

http://www.wampserver.com/ru/

Дальнейшее обучение

Настройка веб-сервера

Проектирование и создание базы данных

Основы JavaScript

JavaScript в браузере

PHP на сервере

PHP + MySQL

Продвижение сайта

Графика

Зачётный тест

Пройти!