Государственное бюджетное профессиональное образовательное учреждение

«Арзамасский коммерческо-технический техникум»

Саблукова Наталья Геннадьевна

Комплект лекций

по дисциплине «Основы сайтостроения»

для студентов специальности
09.02.07 Информационные системы и программирование

Арзамас
2022

Одобрено методическим объединением естественно-математических и информационных дисциплин
Протокол № 1 от 30.08.2022 г

Саблукова Н.Г.
Комплект лекций по дисциплине «Основы сайтостроения» для студентов специальности 09.02.07 Информационные системы и программирование – Арзамас: ГБПОУ АКТТ, 2022. – 60 с.

Комплект лекций содержат материал по дисциплине «Основы сайтостроения», изучаемой студентами данной специальности на первом курсе. Представленная информация может быть использована студентами как во время учебных занятий по данной дисциплине, а также в рамках самостоятельной работы во внеурочное время.

Содержание

Введение

Курс лекций составлен в соответствии с рабочей программой дисциплины «Основы сайтостроения» и требованиями федерального государственного образовательного стандарта среднего профессионального образования по специальности 09.02.07 Информационные системы и программирование.

Курс лекций раскрывает основные вопросы дисциплины «Основы сайтостроения»:

- Основы языка HTML.
- Основы CSS.
- Верстка сайтов.

Лекционный материал содержит краткую информацию по основным темам дисциплины. Каждая лекция содержит:

- план (перечень вопросов, рассматриваемых в рамках темы);
- изложение основных вопросов, в соответствии с представленным планом.

Учебное пособие предназначено для студентов очной формы обучения по специальности 09.02.07 Информационные системы и программирование и может быть использовано как во время учебных занятий по дисциплине «Основы сайтостроения», а также в рамках самостоятельной работы во внеурочное время.

Лекция № 1. Основные понятия сети Интернет. Теги и атрибуты языка HTML

План.

1. Основные понятия сети Интернет
2. Понятие сайта. Что такое HTML?
3. Понятие тега и атрибута языка HTML
4. Структура html-документа

1) Основные понятия сети Интернет

В свободной энциклопедии Википедия дается следующее определение:

Интернет (англ. *Internet*) – всемирная система объединённых компьютерных сетей, построенная на использовании протокола TCP/IP и маршрутизации пакетов данных. Интернет образует глобальное информационное пространство, служит физической основой для Всемирной паутины (WWW, World Wide Web) и множества других систем (протоколов) передачи данных.

Часто упоминается как «*Всемирная сеть*» и «*Глобальная сеть*», в обиходе иногда употребляют сокращённое наименование «*Инéт*». Другими словами, Интернет состоит из множества домашних и корпоративных сетей, принадлежащих различным пользователям, компаниям и предприятиям, работающих по самым разнообразным протоколам, связанных между собой различными линиями связи, которые могут передавать данные по телефонным проводам, оптоволокну, через спутники и радиомодемы.

Проще говоря, ***Интернет*** – это множество компьютеров по всему миру, объединенных в единую сеть, которые постоянно обмениваются какой-либо информацией.

Интернет не имеет никакого собственника, здесь нет и специального органа управления, который бы контролировал всю работу сети Интернет.

В сети есть Ваш компьютер. Компьютеры, которыми вы пользуетесь, называется **клиентским компьютером сети**. Пользователь Интернета платит только за подключение к некоторой региональной сети (провайдеру), которая в свою очередь платит за свой доступ сетевому владельцу государственного масштаба.

Компьютеры, с которых мы берем информацию, называются серверами.

Сервер – это компьютер, на котором лежит огромное количество документов.

Когда мы вводим в браузере адрес web-страницы, то мы обращаемся к серверу, путешествуя по проводам. На сервере есть электронный документ, который нас интересовал, и он у нас отображается.

Структура Интернет напоминает паутину, в узлах которой находятся компьютеры, связанные между собой линиями связи.

У каждого компьютера (точнее, у каждой точки доступа) в сети есть уникальный IP-адрес. Например, один IP адрес может раздавать роутер по нескольким компьютерам.

Существуют два типа IP-адреса:

1) ***Статический IP*** – закрепленные за определенным ПК, не меняется.

2) ***Динамический IP*** – присваивается в тот момент, когда пользователь соединяется с Интернетом. Т.е., когда вы подключились к интернету, то провайдер дает вам свободный IP адрес из своего набора IP.

Сервера имеют статические IP, чтобы можно всегда находить серверы. Клиенты могут иметь динамические адреса, потому что к ним никто не обращаются.

Структура IP-адреса устроена таким образом, что мы можем узнать, в какой стране и в каком городе находится компьютер пользователя. Таким образом, например, при настройке объявлений по контекстной рекламе, можно задавать регионы показа нашего объявления.

Но, как правило, мы не вбиваем в адресную строку этот адрес, а определенную строку. Для того, чтобы не запоминать сложные цифры IP-адреса, были придуманы так называемые доменные имена.

Есть специальная служба DNS – доменная система имен, по сути это таблица, в которой указано название сайта и соответствующий ему IP-адрес. То есть, вводя в адресную строку название сайта, и нажимая Enter, мы сначала обращаемся к DNS, чтобы узнать IP. И уже зная этот адрес, можно добраться до сервера.

Доменное имя - это уникальное имя, которое данный поставщик услуг избрал себе для идентификации, например: mail.ru или google.com

Доменное имя может иметь несколько уровней.

ru, com, net, рф – 1 уровень (обычно мы их видим в конце имени), это глобальные зоны, которые объединяют в себе большую группу сайтов по их принадлежности.

google.com - 2 уровень, когда в доменном имени появляется одна точка.

proglive.tiu.ru - 3 уровень

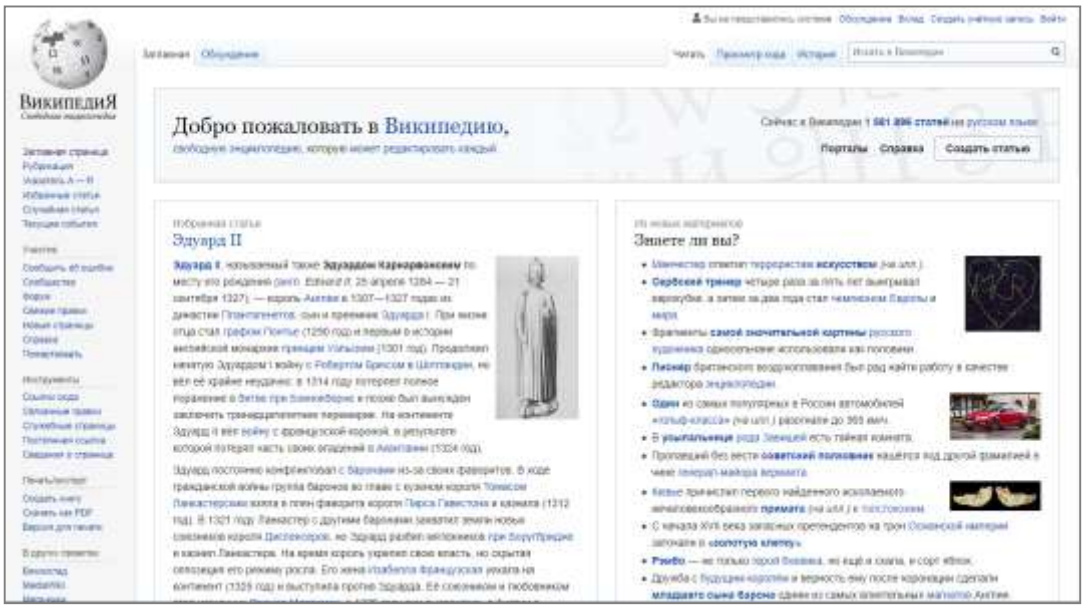
Так можно сделать бесплатный сайт, не покупая место на сервере. Можно зайти на некоторые сайты (народ.яндекс, tiu.ru) и просите там зарегистрировать свой сайт и вам дают уникальное имя. Таким образом, можно свой сайт делить.

Пользователи могут создавать домены со второго уровня, домены второго уровня всегда платные, третьего уровня могут быть бесплатные (размещаются на известных сайтах и на ваши сайты публикуют рекламу).

2) Понятие сайта. Что такое HTML?

Сайт – это совокупность электронных документов (web-страниц), объединенных под одним адресом (доменным именем), связанных между собой ссылками. Доступ к сайту осуществляется через браузер.

Чтобы понять, что такое HTML и зачем он нужен, можно открыть любую веб-страницу в браузере. На рис. 1 показана главная страница сайта ВикипедиЯ (https://ru.wikipedia.org/).



Главная страница сайта ВикипедиЯ

Вся графика и текст, которую Вы видите на этой странице, формируется при помощи команд языка HTML.

HTML – это аббревиатура, которая расшифровывается как HyperText Markup Language. В переводе на русский язык **HTML** означает **язык разметки гипертекста.**

То, что HTML является языком разметки значит, что он позволяет «разметить» веб-страницу на отдельные элементы: заголовок, абзац, картинка, таблица и т.д. и сообщить браузеру какой элемент чем является.

Пример:

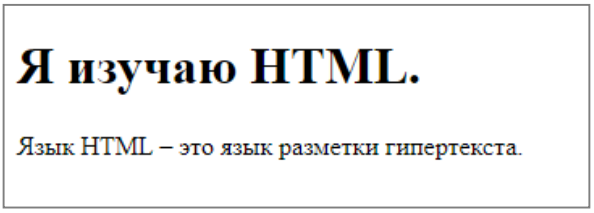
Возьмем следующую строку:

Я изучаю HTML. Язык HTML – это язык разметки гипертекста.

Для того, чтобы браузер вывел первое предложение как заголовок, а второе как абзац, необходимо заключить их в соответствующие теги – команды языка HTML.

<h1>Я изучаю HTML.</h1> Язык HTML – это язык разметки гипертекста.

В результате на веб-странице будут сформированы два элемента: заголовок – тег <h1> </h1> (имеет полужирное начертание, увеличенный шрифт и отступы) и абзац – тег   (имеет обычное начертание, отступы). Внешний вид такой веб-страницы представлен на рис. 2.



Пример веб-странице

HTML – это не язык программирования. Он не служит для выполнения логических и программных операций на странице, не позволяет обрабатывать данные. HTML отвечает за расположение в веб-документе ваших текстов, рисунков, таблиц. Заставить его посчитать, сколько будет дважды два невозможно, зато можно с его помощью красиво и легко выложить информацию о том, что дважды два будет четыре.

Язык HTML был изобретен **Тимом Бернерсом-Ли**, физиком из исследовательского института ЦЕРН в Швейцарии, в 1981-1991 гг. Он придумал идею интернет-гипертекстовой системы.

Гипертекст означает текст, содержащий ссылки на другие тексты.

Тим Бернерс-Ли опубликовал первую версию языка HTML в 1991 году, она состояла всего из 18 тегов. С тех пор каждая новая версия языка HTML содержит новые теги и атрибуты. Самым последним обновлением языка стало внедрение в 2014 году версии **HTML5**.

3) Понятие тега и атрибута языка HTML

HTML-документ – это файл, который имеет расширение **.html** или **.htm**. Вы можете просматривать его с помощью любого веб-браузера (например, Google Chrome, Safari или Mozilla Firefox).

Создавать HTML-документ можно в обычном текстовом редакторе Блокноте. Но удобнее использовать специализированные редакторы с подсветкой кода, такие как, Notepad++, Sublime Text, Visual Studio Code, Atom и др.

Как у любого языка, в HTML есть слова – команды языка и правила написания этих слов – синтаксис языка.

Команды языка HTML называются тегами. Теги записываются в угловых скобках: < >.

Теги могут быть одиночные и парные.

Парный тег:

<имя_тега> *контент* </имя_тега>

открывающийся тег содержимое тега закрывающийся тег

Парный тег представляет собой некоторый контейнер, который определяет область на веб-странице, в которой необходимо разместить какой-то контент.

Пример парного тега:

```
<h1> Сайт для программистов </h1>
```

В данном примере текст «Сайт для программистов» ограничен открывающимся тегом заголовка <h1> и закрывающимся тегом заголовка </h1>. Т.е. текст помещен между тегами, как в контейнер.

Одиночный тег:

```
<имя_тега>
```

Одиночный тег указывает конкретное место в документе, в котором необходимо разместить элемент или выполнить команду.

Примеры одиночных тегов:

 - тег перехода на новую строку;
<hr> - тег горизонтальной линии.

У тегов HTML могут быть атрибуты.
Атрибуты тегов определяют дополнительную информацию об элементе.
Атрибуты прописываются в открывающем теге элемента и содержат имя и значение.
Синтаксис тега с атрибутом:

```
<имя_тега атрибут = "значение_атрибута">
    содержимое тега
</имя_тега >
```

Примеры тегов с атрибутами:

```
<a href="contacts.html" > Контакты </a>
```

Парный тег <a> отображает слово Контакты, как ссылку.
У этого тега имеется атрибут href="contacts.html". В данном атрибуте указывается адрес веб-страницы contacts.html, на которую перейдет пользователь при щелчке по ссылке.

Внутри тега можно указать столько атрибутов, сколько необходимо. Несколько атрибутов у одного тега разделяются друг от друга пробелами.

```
<img src  = "book.png" width = "500">
```

Одиночный тег <img> отображает картинку на веб-странице.
У тега указаны два атрибута:
src = "book.png" – имя картинки, которую необходимо загрузить на веб-страницу.
width = "500" – ширина картинки равна 500 px.

Теги могут вкладываться друг в друга.
Примеры

```
<p> <i> Сайт для программистов </i> </p>
```

При вложении следует соблюдать порядок их закрытия **(принцип «матрёшки»):** теги должны закрываться в порядке, обратном тому, в котором они открывались.

Следующая запись будет неверной: Текст .

При написании тегов рекомендуется придерживаться следующих правил:

1. Записывать все теги и атрибуты строчными (маленькими) буквами.
2. Брать значения атрибутов в двойные или одинарные кавычки.
3. Парный тег обязательно закрывать.

4) Структура html-документа

Для того, чтобы объединить различные теги в целую html-страницу необходимо задать ее структуру.

Структура html-документа имеет следующий вид:

```
<!DOCTYPE html>
<html>
	<head>
		<title>Моя страница</title>
		<meta charset = "utf-8">
	</head>
	<body>

	</body>
</html>
```

Разберем каждую строку структуры html-документа:

1)<!DOCTYPE html>

Тег DOCTYPE определяет версию HTML. <!DOCTYPE html> означает, что будет использоваться самая последняя версия – HTML 5.

2)<html></html>

Элемент html является корневым элементом документа. <html> означает начало html-документа, </html> - его конец. Все остальные элементы должны располагаться внутри контейнера <html></html>.

3)<head></head>

Раздел head содержит служебную информацию о веб-странице (настройки для браузера): заголовок, кодировку, описание, ключевые слова для поисковых машин. В разделе head подключаются файлы стилей css и файлы скриптов.

Все что находиться в данном разделе не видно пользователю, за исключением содержимого тега <title>.

4) <title></title>

Обязательным тегом раздела <head> является тег <title>. Этот элемент устанавливает заголовок для веб-страницы, который является названием, появляющимся на вкладке браузера загружаемой страницы.

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

5) <meta charset="utf-8">

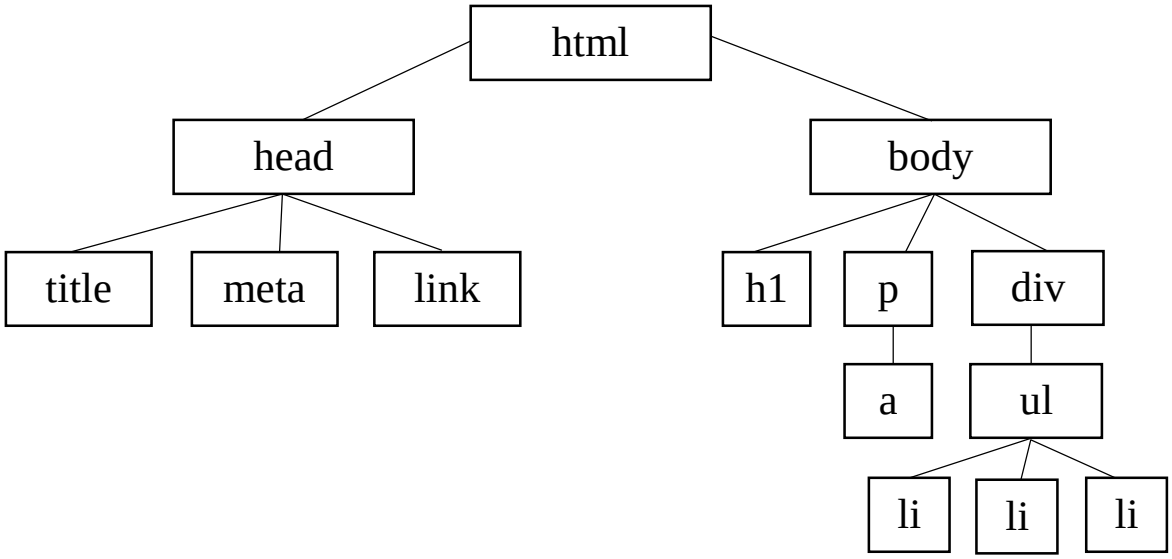
Тег meta с атрибутом charset="utf-8" устанавливает UTF-8 кодировку веб-страницы. Данная кодировка включает в себя большинство символов из всех известных человечеству языков.

6) <body></body>

В разделе body располагается все содержимое веб-страницы, которое видит пользователь (контент): текст, изображения, таблицы, видео, формы и т.д.

Элементы, находящиеся внутри тега <html>, образуют дерево документа, или так называемую **объектную модель документа DOM** (document object model).

На рис. 10 представлен пример структуры html-документа в виде дерева. Главный элемент html является корневым, у него имеется два главных раздела: head и body. В каждом из этих разделов содержатся теги, некоторые из них могут быть вложенными.



Пример структуры html-документа

В коде HTML могут использоваться комментарии.

Комментарии – это пометки для разработчика. Использование комментариев является хорошим тоном для веб-разработчиков, поскольку это ускоряет процесс изучения чужого кода.

```
<!-- текст комментария -->
```

Текст внутри комментария не отображается браузером на странице.
Комментарии обычно используются в следующих случаях:

- для комментирования кода, чтобы улучшить его читабельность.
- для временного отключения кода.

Лекция № 2. Основные теги языка HTML

План

1. Основные теги оформления текста
2. Гиперссылки
3. Изображения
4. Списки

1) **Основные теги оформления текста**

Заголовки

```
<h1> Заголовок 1 уровня - самый большой</h1>
<h2> Заголовок 2 уровня</h2> <h3> Заголовок 3 уровня</h3>
<h4> Заголовок 4 уровня</h4> <h5> Заголовок 5 уровня</h5>
<h6> Заголовок 6 уровня – самый маленький</h6>
```

Абзац (параграф)

```
<p>Здесь текст</p>
```

Теги выделения текста

```
<b>Полужирный текст</b>
<strong> Полужирный текст </strong>

<i>Текст курсивом</i>
<em>Текст курсивом</em>

<u>Подчеркнутый текст</u>

<s>Зачеркнутый текст</s>

<mark>Выделение текста с помощью подсветки фона</mark>
```

Верхний и нижний индексы

```
<sup> Верхний индекс </sup>
<sub> Нижний индекс </sub>
```

Текст с сохранением формата

```
<pre>
            Текст внутри тега <pre> выводится
                    с  тем       количеством          пробелов
      и с теми позициями строк, что были заданы в редакторе.
</pre>
```

Спецсимволы

Спецсимвол	*Описание*	*Спецсимвол*	*Описание*
<	Знак «меньше»	←	Стрелка влево
>	Знак «больше»	→	Стрелка вправо
©	Копирайт		Неразрывный пробел
"	Двойная кавычка		

2) **Гиперссылки**

```
<a href = "URL-адрес">Текст ссылки</a>
```

Внутрь парного тега <a> помещается текст, который будет отображаться на веб-странице. Обязательным атрибутом тега <a> является атрибут href, задающий URL-адрес веб-страницы, на которую необходимо перейти при щелчке на тексте ссылки

Абсолютные ссылки применяются для перехода на страницы внешнего сайта. Абсолютные адреса должны начинаться с указания протокола (http:// или https://) и содержать имя домена

```
<a href =  "http://yandex.ru" target =  "_blank">
        Перейти на сайт Яндекс
</a>
```

Атрибут target = "_blank" позволяет открывать страницу в новой вкладке.

Относительные ссылки применяются для перемещения по веб-страницам внутри сайта. Значение атрибута href зависит от исходного расположения файла.

а) файлы располагаются в одной папке (рис. 1).

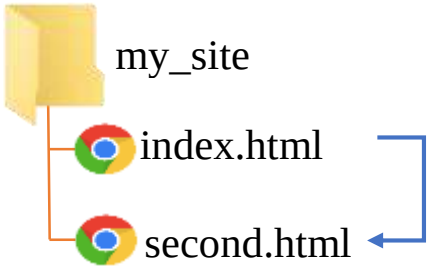


Рис. 1 Файлы в одной папке

```
<a href="second.html"> Перейти на вторую страницу </a>
```

б) Файлы размещаются в разных папках: исходный – в корне сайта (рис. 2).

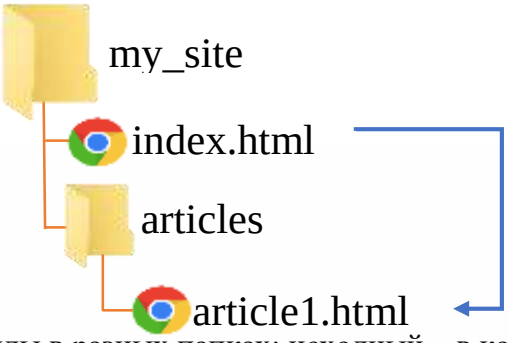


Рис. 2 Файлы в разных папках: исходный – в корне

```
<a href="articles/article1.html"> Читать статью </a>
```

Если файл находится внутри не одной, а двух папок, то путь к нему включает имена всех папок, записанные через /.

в) Файлы размещаются в разных папках: исходный – во внутренней папке (рис. 3).

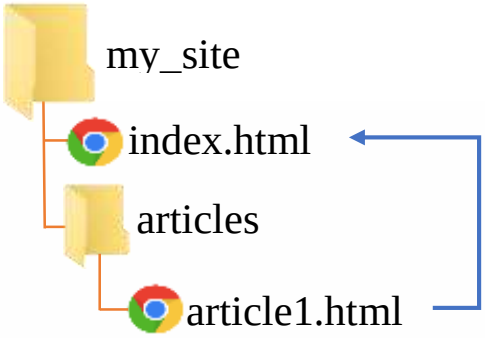


Рис. 3 Файлы в разных папках: исходный – во внутренней папке

```
<a href="../index.html"> Вернуться на главную </a>
```

Если исходный файл находится в двух вложенных папках и необходимо сослаться на документ в корне сайта, то конструкцию ../ следует записать два раза.

г) Каждый файл располагается в своей папке (рис. 4).

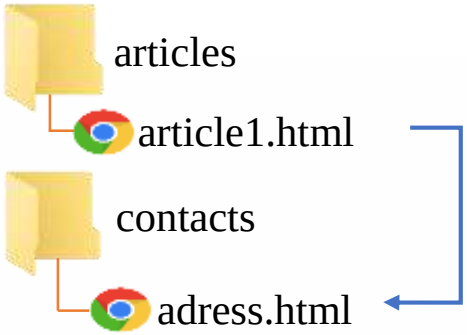


Рис. 4 Каждый файл располагается в своей папке

```
<a href="../contacts/adress.html">Адрес</a>
```

Якоря (внутренние ссылки) создают переходы на различные разделы текущей веб-страницы, позволяя быстро перемещаться между разделами.

```
Присвоение абзацу идентификатора id = "p10"
<p id = "p10"> десятый абзац </p>

Ссылка перехода к абзацу с id = "p10"
<a href =  "#p10">Перейти к 10-ому абзацу</a>
```

3) Изображения

```
<img src = "img/my_foto.jpg">
```

№	*Атрибут*	*Описание, принимаемое значение*
1.	src	Полное имя изображения *Синтаксис:* src = "img/my_foto.jpg"
2.	alt	Альтернативный текст для изображения (при отключенной графике). *Синтаксис:* alt="описание изображения"
3.	width	Ширина изображения в px. *Синтаксис:* width="500"
4.	height	Высоту изображения в px. *Синтаксис:* height="300"
5.	title	Универсальный атрибут, который можно использовать практически для любого тега. *Синтаксис:* title="описание изображения"

4) Списки

Маркированный список

```
<ul>
        <li>Красный</li>
        <li>Зеленый</li>
        <li>Синий</li>
</ul>
```

Нумерованный список

```
<ol>
        <li>Монитор</li>
        <li>Клавиатура</li>
        <li>Колонки</li>
</ol>
```

№	*Атрибут*	*Описание, принимаемое значение*
1.	start	Номер, с которого должен начинаться нумерованный список. Синтаксис: <ol start ="2">

2.	reversed	Атрибут reversed позволяет списку отображаться в обратном порядке. Синтаксис: <ol reversed>
3.	value	Применяется к пунктам в нумерованном списке, чтобы изменить его значение в списке. Синтаксис: <li value="7">

Список определений

```
<dl>
        <dt> Термин1 </dt>
        <dd> Определение термина1 </dd>
        <dt> Термин2 </dt>
        <dd> Определение термина2</dd>
 </dl>
```

Лекция № 3. Основы CSS

План

1. Что такое CSS? Синтаксис CSS
2. Подключение CSS к HTML
3. Селекторы CSS

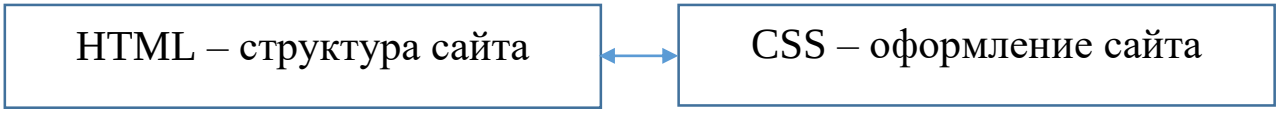
1) Что такое CSS? Синтаксис CSS

Рассмотренные в первой главе теги HTML позволяют задать структуру сайта и разместить различные элементы на веб-странице. Раньше, для того, чтобы оформить веб-страницу, т.е. добавить цвета, выравнивание, параметры шрифтов и т.п., использовали специальные атрибуты тегов. Однако у такого способа были ограниченные возможности в оформлении и сайты получались скучные и однотипные.

Пример того, как не нужно оформлять веб-страницу!

```
<body bgcolor="green">
        <h1 align="center">
                <font color="red">Заголовок</font>
        </h1>
</body>
```

В настоящее время для оформления веб-страниц используются каскадные таблицы стилей CSS. При помощи CSS можно менять цвет и шрифт у текста, изменять положение элементов на странице, их размеры, задавать элементам рамки, границы и отступы.

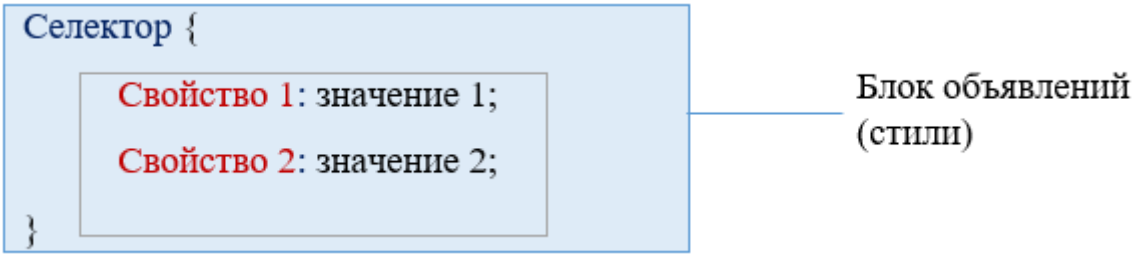


CSS (Cascading Style Sheets) – каскадные таблицы стилей, которые применяются для описания внешнего вида веб-документа, написанного при помощи языка разметки HTML.

Термин «каскадные таблицы стилей» был предложен Хоконом Ли в 1994 году. Совместно с Бертом Босом он стал развивать CSS. В настоящее время используется CSS версии 3.

CSS представляет из себя набор правил, описывающих форматирование (изменение внешнего вида) отдельных элементов на веб-странице. Правило, состоит из двух частей: **селектора** и следующего за ним **блока объявлений**.

Синтаксис CSS



Первым всегда указывается *селектор*, он сообщает браузеру, к какому элементу веб-страницы будет применен стиль. Селекторами могут быть любые теги, формирующие тело сайта, а также идентификаторы, классы и атрибуты тегов.

Далее следует *блок объявлений*, который начинается открывающейся фигурной скобкой { и заканчивается закрывающейся фигурной скобкой }. Между фигурными скобками указываются команды CSS – объявления.

Каждое *объявление* состоит из двух частей: свойства и его значения. Объявление всегда должно заканчиваться точкой с запятой.

Свойство – это команда форматирования, которая задает конкретный стиль для элемента. Каждое свойство имеет предопределенный набор значений. После имени свойства указывается двоеточие, которое отделяет название свойства от допустимого значения.

Пример CSS правила:

```
h1 {
        color: red;
        font-size: 20px;
}
```

В данном примере селектором является элемент h1 – заголовок первого уровня. Заметьте, что тег h1 записывается в CSS без угловых скобок. К заголовку применяются два стиля:

- color: red – свойства color со значение red задает красный цвет шрифта у заголовка;
- font-size: 20px – свойство font-size со значением 20px задает размер шрифта у заголовка.

Выше рассмотрен многострочный вариант расположения команд CSS. Он считается более наглядным, безопасным и удобным. Поэтому именно многострочный вариант чаще рекомендуется для использования. Однако иногда можно применять и однострочный вариант расположения команд.

```
h1 {color: red;font-size: 20px;}
```

Для задания комментарий в CSS используется конструкция /* … */. Такая конструкция позволяет комментировать одну или несколько строк CSS.

```
/* здесь пишется комментарий */
/*
и здесь можно
тоже указать комментарий*/
```

2) Подключение CSS к HTML

Для того, чтобы использовать стили CSS в веб-документе, необходимо их сначала подключить. В зависимости от способа подключения выделяют внешние таблицы стилей, внутренние и встроенные стили.

1) **Внешняя таблица стилей** представляет собой созданный в текстовом редакторе файл с расширением CSS, в котором находится весь CSS-код веб-страницы. Внутри файла содержатся только стили без HTML-разметки.

Внешняя таблица стилей (CSS-файл) подключается к html-документу с помощью тега **<link>**, расположенного внутри раздела <head></head>.

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

Тег <link> имеет два обязательных атрибута: href и rel. В атрибуте href указывается путь к внешнему CSS-файлу. Атрибут rel = "stylesheet" показывает тип ссылки. Значение stylesheet указывает на то, что подключается файл стилей CSS.

У тега <link> имеется также атрибут type = "text/css", но по стандарту HTML5 он не является обязательным, поэтому его можно не указывать.

Внешнюю таблицу стилей можно подключить ко всем страницам сайта.

К веб-странице можно присоединить несколько таблиц стилей, добавляя последовательно несколько тегов <link>, каждый из которых будет указывать на свой файл с таблицей.

Пример веб-страницы с внешней таблицей стилей (style.css)

Файл index.html

```
<!DOCTYPE html>
<html>
        <head>
                <meta charset="UTF-8">
                <title>Способы подключения CSS</title>
                <link href ="style.css" rel = "stylesheet">
        </head>
        <body>
                <p>Внешняя таблица стилей</p>
        </body>
</html>
```

Файл style.css

```
body{
        background-color: grey; /*цвет фоновой заливки: серый */
}
p{
        color: blue; /*цвет текста: синий*/
        font-size: 16px; /*размер текста: 16px*/
        text-align: center; /*выравнивание текста: по центру*/
}
```

В данном примере к файлу index.html подключается внешняя таблица стилей – файл style.css. В CSS файле задаются стили для тела сайта body и для абзаца p. В комментариях расшифрованы использованные свойства и их значения. В результате применения стилей CSS веб-страница будет оформлена следующим образом: на сером фоне по центру отображается синий текст с размером букв 16px.

Использование внешней таблицы стилей является самым удобным способом подключения CSS. Можно выделить следующие преимущества такого способа:

- меньший размер html-страницы и более чистая структура кода;
- быстрая скорость загрузки html-страницы;
- для разных веб-страниц может быть использован один css файл;
- удобство в редактировании стилей.

2) **Внутренние стили** являются частью кода html-страницы. Они располагаются внутри тега **<style></style>**, расположенного в разделе <head></head>.

Рассмотренный пример с использованием внутренних стилей будет выглядеть следующим образом:

Файл index.html

```
<!DOCTYPE html>
<html>
	<head>
		<meta charset="UTF-8">
		<title>Способы подключения CSS</title>
		<style>
			body{
				background-color: grey;
			}
			p{
				color: blue;
				font-size: 16px;
				text-align: center;
			}
		</style>
	</head>
	<body>
		<p>Внешняя таблица стилей</p>
	</body>
</html>
```

Внутренние стили имеют приоритет над внешними, но уступают встроенным стилям.

3) **Встроенные стили (inline-стили)** заключаются в использовании атрибута style непосредственно у того элемента, который необходимо стилизовать. В качестве значения атрибута указываются CSS-свойства.

```
<body style = "background-color: grey; ">
	<ul>
		<li style = "color: red; font-size: 18px; "> Монитор </li>
		<li> Клавиатура</li>
	</ul>
</body>
```

Минус от использования встроенных стилей заключается в том, что они создают дополнительные неудобства: смешивается структура документа с его оформлением, поиск и правка таких стилей занимает достаточно много времени. Однако встроенные стили обладают наибольшим приоритетом.

3) Селекторы CSS

Селекторы представляют структуру веб-страницы. С их помощью создаются правила для форматирования элементов веб-страницы. Селекторами могут быть теги, их классы и идентификаторы, а также атрибуты.

1) **Селектор тега**

В качестве селектора может выступать любой тег HTML.

В рассмотренных выше примерах использовались именно селекторы тегов.

Синтаксис CSS

```
тег {
        свойство 1: значение 1;
        свойство 2: значение 2;
}
```

Селектор тега применяется, когда необходимо задать один стиль для всех элементов с одинаковым тегом. Например, если нужно задать отступ у первой строки для всех абзацев или цвет всех заголовков второго уровня.

Однако, часто необходимо изменить цвет не всех заголовков на веб-странице, а только одного или двух. CSS предоставляет такую возможность с помощью селекторов id (идентификатор) и class (класс).

2) **Селектор id (идентификатор)** предназначен для применения стиля к уникальным элементам на веб-странице. **Каждый идентификатор может встречаться на странице только один раз.**

Чтобы задать данный селектор, следует для тега, к которому нужно применить стиль, добавить атрибут id и в его значении которого указать уникальное имя. В CSS коде селектор id начинается с символа #, после которого идет имя идентификатора.

Синтаксис CSS

```
#имя идентификатора {
        свойство 1: значение 1;
        свойство 2: значение 2;
}
```

Пример использования селектора id

Файл index.html

```
<p id = "first">Первый абзац </p>
<p> Второй абзац </p>
<p> Третий абзац </p>
<p> Четвертый абзац </p>
<p> Пятый абзац </p>
```

Файл style.css

```
#first {
        color: red; /*цвет текста: красный*/
        font-weight: bold; /*толщина текста: полужирное начертание*/
}
```

В данном примере указанные стили будут применены только к первому абзацу, для которого задан id = "first".

3) **Селектор class (класс)** позволяет применить стиль к нескольким подобным элементам на веб-странице.

Для использования селектора class следует для тегов, к которым необходимо применить стиль, добавить атрибут class. В качестве значения указать определённое имя класса. В CSS коде селектор class начинается с точки, после которого идет имя класса.

Также можно использовать классы и с указанием тега, к которому они применяются.

Синтаксис CSS

```
.имя класса {
        свойство 1: значение 1;
        свойство 2: значение 2;
}
```

```
тег.имя класса {
        свойство 1: значение 1;
        свойство 2: значение 2;
}
```

Имя идентификатора и класса всегда задается на английском языке и начинаться с буквы. Оно может содержать цифры, символ дефиса (-) и нижнего подчеркивания (_).

Пример использования селектора class

Файл index.html

```
<p id = "first" class= "odd">Первый абзац </p>
<p> Второй абзац </p>
<p class= "odd"> Третий абзац </p>
<p> Четвертый абзац </p>
<p class= "odd"> Пятый абзац </p>
```

Файл style.css

```
#first {
        color: red; /*цвет текста: красный*/
        font-weight: bold; /*толщина текста: полужирное начертание*/
}
.odd{
        background-color: #eee; /*цвет заливки: серый*/
}
```

В данном примере все нечетные абзацы с атрибутом class= "odd" будут выделены серым цветом.

Для уточнения элемента, к которому применяется стиль, в CSS можно указывать тег, у которого задан класс.

```
p.odd{
        background-color: #eee;
}
```

Классы можно задавать не только для одинаковых тегов. В следующем примере атрибут class="is_border" указан у разных тегов, служащих для отображения текстовой информации: у заголовка и у абзаца.

Файл index.html

```
<h1 class= "is_border">Заголовок с рамкой </h1>
<p class= "is_border"> Абзац с рамкой </p>
```

Файл style.css

```
.is_border{
        border: 1px solid #000; /*рамка: толщина 1px сплошная черного цвета*/
}
```

Для одного элемента иногда необходимо применить несколько классов, при этом их имена в значении атрибута следует указывать через пробел. Например, в следующем примере для заголовка заданы два класса с именами heading и logo. У каждого класса в CSS описаны свои стили.

Файл index.html

```
<h1 class= "heading logo">Заголовок</h1>
```

Файл style.css

```
.heading{
        text-transform: uppercase; /*все символы заглавные*/
        width: 200px; /*ширина текстового блока: 200px*/
}
.logo{
        border: 1px solid #abc;
}
```

4) **Универсальный селектор**

Часто возникает необходимость использования единого стиля одновременно для всех элементов веб-страницы. Для этого используется универсальный селектор. Стиль, определённый для универсального селектора, будет выполнен над всеми элементами.

Универсальный селектор записывается символом *.

Синтаксис CSS

```
* {
        свойство 1: значение 1;
        свойство 2: значение 2;
}
```

Лекция № 4. Основные свойства стилей

План

1. Основные свойства стилей CSS
2. Наследование, вложенность и группировка свойств
3. Псевдоклассы и псевдоэлементы

1) Основные свойства стилей CSS

При изучении способов подключения и селекторов CSS уже были рассмотрены некоторые примеры свойств стилей и их значений.

В данной теме будут представлены часто используемые свойства CSS и их возможные значения.

1. Фон элемента – свойство background

Свойство background позволяет установить характеристики фона для различных элементов веб-страницы.

Характеристики фона

а) Свойство ***background-color*** – устанавливает цвет фона у элемента.

Синтаксис CSS

```
background-color: red;
background-color: #00af45;
background-color: #a0c;
background-color: rgba(255, 128, 128, 0.5);
background-color: transparent;
```

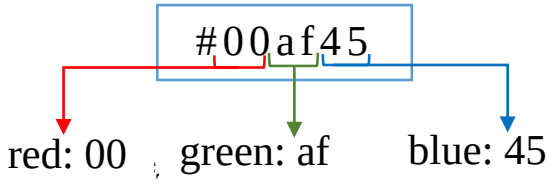
В примере выше показаны различные способы задания цвета на веб-странице:

– background-color: red; – используется название цвета на английском языке. При таком подходе есть ограничения, т.к. невозможно получить различные оттенки цветов.

– background-color: #00af45; – используется шестнадцатеричный код числа в модели RGB. Такой способ позволяет выбрать один из 16 млн. цветов.

Цифровая модель RGB расшифровывается как Red-Green-Blue, то есть Красный-Зеленый-Синий. Цвет в RGB представляется в виде трех пар шестнадцатеричных цифр (от 0 до 9, A, B, C, D, E, F), где каждая пара отвечает за свой цвет: две первые цифры – красный, две в середине – зеленый, две последние цифры – синий.

Чем больше значение цифры, тем сильнее данный оттенок проявит себя в цвете. Например, в цвете, заданном как #00af45, большее значение у зеленой составляющей (рис. 71). В результате получится некоторый оттенок зеленого цвета.

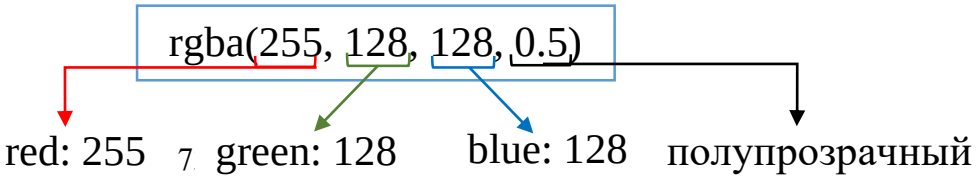


– background-color: #a0c; – сокращенная форма записи цвета в RGB. Используется, если совпадают два числа или две буквы одного и того же цвета.

Сокращенная форма записи цвета в RGB

– background-color: rgba(255, 128, 128, 0.5); – используется десятичный код числа в модели RGB и значение прозрачности.

Каждая из трех компонент RGB (красный, зеленый и синий) может принимать значение от 0 до 255. Последний параметр (альфа-канал) отвечает за прозрачность фона: значение 0 соответствует полной прозрачности, 1 – непрозрачность, промежуточное значение вроде 0.5 – полупрозрачности.



– background-color: transparent; – устанавливает прозрачный фон.

При подборе цвета лучше использовать любой графический редактор (Gimp, Photoshop). В нем можно выбрать нужный цвет из палитры, и редактор покажет шестнадцатеричный и десятичный код выбранного цвета.

Подобрать цвета также можно с помощью различных онлайн сервисов: https://colorscheme.ru/, https://color.romanuke.com/ и др.

б) Свойство ***background-image*** – устанавливает одно или несколько фоновых изображений для элемента.

Синтаксис CSS

```
background-image: url(img/foto.jpg); /*одно фоновое изображение*/
background-image: url(http://site.ru/rose.png);
background-image: url(t1.png), url(tr.png); /*несколько фоновых изображений*/
background- image: none; /*отменяет фоновое изображение*/
```

В качестве значения используется путь к графическому файлу, который указывается внутри конструкции url(). Путь к файлу при этом можно писать, как в кавычках (двойных или одинарных), так и без них.

Можно добавить несколько фоновых изображений для элемента, перечислив их параметры через запятую.

в) Свойство ***background-position*** – указывает, расположение фонового изображения.

Синтаксис CSS

```
background-position: left; /*слева*/
background-position: right; /*справа*/
background-position: top; /*сверху*/
background-position: bottom; /*снизу*/
background-position: center; /*по центру*/
```

Часто расположение фонового изображения задается сразу для двух осей, в этом случае значения свойства background-position указываются через пробел. Например, если изображение нужно вывести в правом верхнем углу, то команда будет выглядеть следующим образом:

```
background-position: top right;
```

Можно управлять расстоянием для фонового изображения от левого и верхнего краев элемента с помощью точных значений, указанных в единицах измерения CSS (например, в пикселях или %).

```
background-position: 10px 25px;
```

Такая запись означает, что фоновое изображение удалено от левого края на 10 пикселей и от верхнего края на 25 пикселей. Здесь важен порядок записи значений: первое значение отвечает за положение фонового рисунка по горизонтали, второе — по вертикали. Допустимы и отрицательные значения (например, они могут быть полезны в случае, если с левой или верхней стороны фоновой картинки есть область, которую необходимо спрятать).

г) Свойство ***background-repeat*** – определяет, нужно ли повторять фоновое изображение.

Синтаксис CSS

```
background-repeat: repeat-x; /*повтор по горизонтали*/
background-repeat: repeat-y; /*повтор по вертикали*/
background-repeat: repeat; /*повтор по вертикали и горизонтали*/
background-repeat: no-repeat; /*нет повтора*/
background-repeat: space; /*изображение повторяется столько раз, чтобы полностью заполнить область; если это не удается, между картинками добавляется пустое пространство*/
```

```
background-repeat: round; /*изображение повторяется так, чтобы в области поместилось целое число рисунков; если это не удается сделать, то фоновые рисунки масштабируются*/
```

д) Свойство ***background-size*** – определяет размер фонового изображения.

Синтаксис CSS

```
background-size: 300px; /*ширина изображения в px*/
background-size: auto 30%; /* ширина изображения вычисляется автоматически с сохранением пропорций, высота изображения – 30%*/
background-size: auto auto; /*исходные размеры изображения*/
background-size: cover; /*масштабирование изображения с сохранением пропорций на всю ширину или высоту элемента*/
background-size: contain; /*масштабирование изображения с сохранением пропорций, так чтобы изображение целиком поместилось в элемент*/
background-size: 6px, cover; /*размеры для двух фоновых картинок: у первой – ширина 6px, вторая – растянута на всю область фона*/
```

Если в значении свойства background-size указано одно значение, то это значение ширины изображения, высота при этом вычисляется автоматически, исходя из пропорций картинки (значение auto). Если в значении свойства background-size указано два значения, то первое значение – это ширина изображения, второе – высота.

Значение auto, заданное одновременно для ширины и высоты (background-size: auto auto), сохраняет исходные размеры фоновой картинки. Если auto установлено только для одной стороны картинки (background-size: auto 30%), то размер этой стороны вычисляется автоматически исходя из пропорций картинки.

Значения для нескольких фонов указываются через запятую.

В CSS используется несколько единиц измерения для определения размеров элементов. К ним относятся:

– пиксели (px) – единицы фиксированного размера;

– проценты (%) – масштабируют размер элемента в процентах относительно родительского элемента;

– относительная единица (em) – 1em равен текущему размеру элемента, 2em увеличивает размер в 2 раза и т.д. (em часто используется для размера шрифта).

д) Свойство ***background-attachment*** – определяет, будет ли изображение прокручиваться вместе с содержимым элемента.

Синтаксис CSS

```
background-attachment: scroll; /*фоновое изображение прокручивается вместе с текстом и другим содержимым */
background-attachment: fixed; /*фоновое изображение остается неподвижным*/
background-attachment: local; /*фоновое изображение прокручивается вместе содержимым элемента*/
```

Рассмотренные характеристики рекомендуется записывать в краткой форме, используя свойство background. Порядок написания значений может быть произвольным – браузер сам определит соответствие свойств и значений.

Краткая запись:

background: background-color background-image background-repeat background-position / background-size

```
background: #ff0 url(img/foto.jpg) repeat-x top/50px;
```

Как видно на рисунке и в примере, значения всех свойств указываются через пробел. Исключением являются свойства background-position и background-size – их необходимо разделять знаком слэша /.

2. Ширина и высота элемента – свойства width и height

Синтаксис CSS

```
width: 200px;
height: 50%;
```

В качестве значений принимаются любые единицы размеров, принятые в CSS – например, пиксели (px), проценты (%) и др. При использовании процентной записи ширина элемента вычисляется в зависимости от ширины родительского элемента. Если родитель явно не указан, то им выступает окно браузера.

3. Рамка вокруг элемента – свойство border

Свойство border позволяет установить характеристики рамок вокруг элемента на веб странице.

Характеристики рамки

а) Свойство ***border-color*** – цвет рамки.

Свойство задаёт цвет рамок всех сторон одновременно. С помощью уточняющих свойств можно установить свой цвет для рамки каждой стороны элемента.

б) Свойство ***border-style*** – стиль рамки.

Стиль рамки определяет ее отображение, без этого свойства рамка не будет видна вообще.

Синтаксис CSS

```
border-style: solid; /*сплошная рамка*/
border-style: dotted; /*точечная рамка*/
border-style: dashed; /*пунктирная рамка*/
border-style: double; /*двойная рамка*/
border-style: inset; /*трехмерная рамка с углублением*/
border-style: outset; /*трехмерная выпуклая рамка*/
border-style: groove; /*трехмерная рамка с прорезанным желобом*/
border-style: ridge; /*трехмерная рамка с оттиснутой бороздой*/
```

Значения свойства border-style

в) Свойство ***border-width*** – толщина рамки.

Толщину рамки можно задать для всех сторон одинаковую или для каждой из 4 сторон разную.

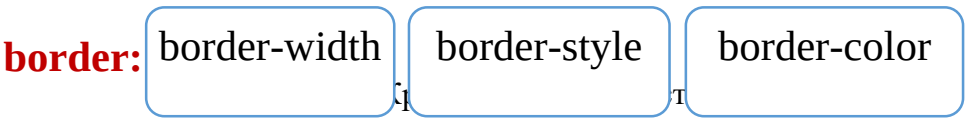
Синтаксис CSS

```
border-width: 1px; /*толщина всех сторон – 1px*/
border-width: 1px 2px; /*1px – верхняя и нижняя сторона, 2px – левая и правая*/
border-width: 1px 2px 3px; /*1px – верхняя, 2px – левая и правая, 3px – нижняя*/
```

```
border-width: 1px 2px 3px 4px; /*1px – верхняя, 2px – правая, 3px – нижняя, 4px – левая*/
```

Рассмотренные характеристики рамки рекомендуется перечислять в одну строку, разделяя пробелом, используя свойство border.

Краткая запись:




```
border: 1px solid black;
```

При этом заданные свойства будут применяться ко всем границам элемента одновременно. Также возможно каждую границу задавать отдельно. Для этого используются следующие свойства:

- border-top – стиль верхней границы;
- border-bottom – стиль нижней границы;
- border-left – стиль левой границы;
- border-right – стиль правой границы.

4. Цвет текста – свойство color

Цвет текста можно задавать любым способом, также как и цвет фона.

Синтаксис CSS

```
color: red;
color: #78fa2e;
color: RGB(34,21,56);
```

5. Шрифт – свойство font

Свойство **font** позволяет установить характеристики шрифта текста внутри элемента на веб странице.

Характеристики шрифта

а) Свойство ***font-family*** – устанавливает семейство шрифта, которое будет использоваться для оформления текста внутри элемента.

Список шрифтов может включать одно или несколько значений, разделенных запятой. Если в имени шрифта содержатся пробелы, например, Times New Roman, то оно должно заключаться в одинарные или двойные кавычки.

Синтаксис CSS

```
font-family: Cambria, "Times New Roman", serif;
```

В данном примере, браузер обрабатывает список шрифтов, заданный свойством font-family, следующим образом:

1) сначала он проверяет, установлен ли шрифт Cambria на компьютере, и, если да, использует его в качестве шрифта для текста внутри элемента;
2) если Cambria не установлен, то ищет шрифт Times New Roman, в случае успешного поиска использует его для шрифта;

3) если ни один из указанных шрифтов не найден, то применяется первый найденный браузером на компьютере шрифт из семейства serif.

В CSS все шрифты разделены на семейства, каждое семейство состоит из набора шрифтов, обладающих общими характеристиками. Существует всего пять семейств шрифтов:

– *sans-serif* – шрифты без засечек (Helvetica, Geneva, Arial, Verdana, Trebuchet, Univers), считается что они лучше читаются на экране компьютера, чем шрифты семейства serif;

– *serif* – шрифты с засечками (Times New Roman, Times, Garamond, Georgia), засечки – это декоративные штрихи и черточки по краям букв;

– *monospace* – семейство, состоящее из шрифтов, символы которых имеют одинаковую фиксированную ширину (Courier, Courier New, Andele Mono);

– *cursive* – шрифты, имитирующие рукописный текст (Comic Sans, Gabriola, Monotype Corsiva, Author, Zapf Chancery);

– *fantasy* – художественные и декоративные шрифты (Western, Woodblock, Klingon). Они не очень широко распространены, доступны не на всех компьютерах и редко используются в веб-дизайне.

б) Свойство ***font-weight*** – задает насыщенность шрифта.

Значение устанавливается от 100 до 900 с шагом 100. Сверхсветлое начертание, которое может отобразить браузер, имеет значение 100, а сверхжирное – 900. Нормальное начертание шрифта, которое установлено по умолчанию, эквивалентно 400, стандартный полужирный текст — значению 700.

Насыщенность шрифта также задаётся с помощью ключевых слов.

Синтаксис CSS

```
font-weight: normal; /*нормальный шрифт (по умолчанию)*/
font-weight: bold; /*полужирный шрифт*/
font-weight: bolder; /*насыщенность шрифта больше, чем у родителя*/
font-weight: lighter; /*насыщенность шрифта меньше, чем у родителя*/
font-weight: 500; /*средняя насыщенность шрифта*/
```

в) Свойство ***font-style*** – устанавливает стиль начертания шрифта.

Синтаксис CSS

```
font-style: normal; /*обычное начертание шрифта (по умолчанию)*/
font-style: italic; /*начертание курсивом*/
font-style: oblique; /*наклонное начертание*/
```

Разница между курсивом и наклонным начертанием заключается в том, что курсив вносит небольшие изменения в структуру каждого символа, в то время как наклонное начертание представляет собой наклонную версию прямого шрифта.

г) Свойство ***font-size*** – устанавливает размер (высоту) шрифта.

Существует несколько способов указания размера шрифта, наиболее распространенные из них:

– *px (пиксели);*

– *% (проценты);*

– *em* (размер шрифта изменяется относительно размера шрифта в родительском элементе);

– *ключевые слова*

CSS предлагает семь ключевых слов, которые позволяют назначить размер шрифта относительно размера по умолчанию (16px): *xx-small, x-small, small, medium, large, x-large и xx-large*.

Среднее значение medium - размер шрифта по умолчанию в браузерах. Остальные значения уменьшают или увеличивают размер шрифта с различными коэффициентами. Самый маленький размер шрифта xx-small равен примерно 9 пикселям, каждый последующий размер примерно на 20% больше предыдущего.

Другой набор ключевых слов: *larger* и *smaller* устанавливает изменение размера шрифта относительно родителя.

Синтаксис CSS

```
font-size: 20px;
font-size: 80%;
font- size: 0.8em;
font- size: small;
```

Указывать стиль шрифта можно при помощи сокращенной записи. В данном случае важен порядок следования значений:

Краткая запись:


```
font: bold 24px Arial, Verdana, sans-serif;
```

В качестве обязательных значений свойства font указывается размер шрифта и его семейство. Остальные значения задаются при желании.

6. Выравнивание текста по горизонтали – свойство text-align

Синтаксис CSS

```
text-align: left; /*выравнивание по левому краю*/
text-align: right; /*выравнивание по правому краю*/
text-align: center; /*выравнивание по центру*/
text-align: justify; /*выравнивание по ширине*/
```

7. Изменение регистра символов – свойство text-transform

Синтаксис CSS

```
text-transform: capitalize; /*каждое слово начинается с прописной (заглавной) буквы*/
text-transform: lowercase; /*все символы выводятся строчными буквами*/
text-transform: uppercase; /*все символы выводятся прописными буквами*/
```

8. Оформление текста – свойство text-decoration

Синтаксис CSS

```
text-decoration: line-through; /*перечёркнутый текст*/
text-decoration: overline; /*линия над текстом*/
text-decoration: underline; /*подчёркнутый текст*/
text-decoration: none; /*отменяет все эффекты, в том числе подчеркивание у ссылок*/
```

9. Оформление списков – свойство list-style

Свойство list-style позволяет установить стиль маркера у списка на веб странице.

Характеристики стиля маркера

а) Свойство ***list-style-type*** – изменяет тип маркера или удаляет маркер для маркированного и нумерованного списков.

Синтаксис CSS

```
list-style-type: none; /*маркер отсутствует*/

/*значения для маркированного списка*/
list-style-type: disc; /*маркер в виде точки (по умолчанию) */
list-style-type: circle; /*маркер в виде кружка*/
list-style-type: square; /*маркер в виде квадрата*/

/*значения для нумерованного списка*/
list-style-type: decimal; /*арабские числа (по умолчанию) */
list-style-type: lower-alpha; /*строчные латинские буквы*/
list-style-type: upper-alpha; /*заглавные латинские буквы*/
list-style-type: lower-roman; /* римские числа в нижнем регистре */
list-style-type: upper-roman; /*римские числа в верхнем регистре*/
```

б) Свойство ***list-style-position*** – определяет расположение маркера.

Синтаксис CSS

```
list-style-position: outside; /*маркеры размещаются за пределами блока с текстом*/
list-style-position: inside; /*маркер является частью текстового блока*/
```

в) Свойство ***list-style-image*** – позволяет установить в качестве маркера списка изображение. Путь к изображение указывается в скобках после слова url.

Синтаксис CSS

```
list-style-image: url(img/book.svg);
```

Все три свойства форматирования можно объединить в одно свойство list-style. Значения свойств могут располагаться в произвольном порядке, часть значений можно опустить.

Краткая запись:

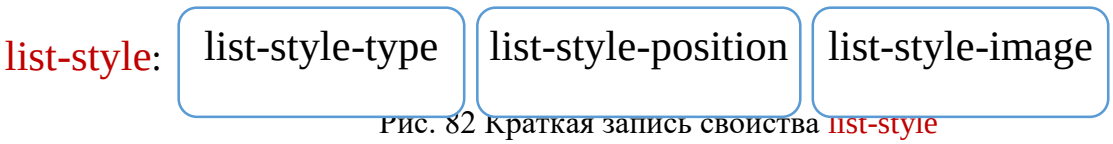


Рис. 82 Краткая запись свойства list-style

```
list-style: circle inside;
```

10. Отступы

Для создания промежутков между элементами на веб-странице используются отступы, которые могут быть внешними и внутрениими.

Свойство ***padding*** – внутренние отступы – расстояние от содержимого до края блока.
Свойство ***margin*** – внешние отступы (поля) – расстояние между блоками.
Значение свойств может быть указано в любых единицах CSS – px, %, em и т.д.

Синтаксис CSS

```
margin: 10px; /*поля вокруг блока с элементом10px */
padding: 5px; /* отступы от элемента до края блока 5px */
```

Величина отступа может быть независимо определена для каждой стороны элемента. Для этого используются свойства:

- padding-top – отступ сверху;
- padding-bottom – отступ снизу;
- padding-left – отступ слева;
- padding-right – отступ справа;

- margin-top – поле сверху;
- margin-bottom – поле снизу;
- margin-left – поле слева;
- margin-right – поле справа.

Отличие между свойствами padding и margin

Также можно задать поля или отступы для четырёх сторон, используя сокращенную запись margin и padding.

Синтаксис CSS

```
padding: 2px 4px 5px 10px; /*отступ сверху– 2px, справа – 4px, снизу – 5px, слева - 10px */
padding: 3px 6px 9px; /*отступ сверху– 3px, справа и слева – 6px, снизу – 9px*/
padding: 6px 12px; /*отступ сверху и снизу – 6px, справа и слева – 12px*/
```

Эти же правила относятся и к свойству margin.

2) Наследование, вложенность и группировка свойств

При изучении тегов HTML мы рассматривали, что можно вкладывать одни HTML теги в другие. А при помощи CSS есть возможность управлять различными вложенными конструкциями. Для управления вложенностью в CSS существуют несколько специальных селекторов. Рассмотрим эти селекторы на примерах.

Наследование стиля – это перенос стиля от родительского элемента к вложенным в него тегам.

```
<body>
	<h1>Заголовок</h1>
	<p id="p1">
		Первый абзац
		<strong> самый важный </strong>
	</p>
</body>
```

```
body{
	font-family: Calibri, sans-serif;
	color: #AA0000;
}
#p1{
	font-style: italic;
}
```

Весь текст будет темно-красного цвета шрифта Calibri, включая заголовок, абзац и строку с жирным цветом. Они наследует стиль css.

Ссылки не наследуют стили, чтобы они всегда на сайте выделялись. Можно в этом случае использовать вложенность или стили для ссылок.

```
<body>
	<h1>Заголовок</h1>
	<p id="p1">
		Первый абзац
		<strong> самый важный </strong>
		<a href="">Здесь будет ссылка</a>
	</p>
</body>
```

```
<body>
	<h1>Заголовок</h1>
	<p id="p1">
		Первый абзац
		<strong> самый важный </strong>
		<a href="" class="link1">Здесь будет ссылка</a>
	</p>
</body>
```

```
.link1{
	font-style: italic;
	font-family: Calibri, sans-serif;
	color:#aa0000;
}
```

Вложенность – когда один тег лежит внутри другого.

```
#p1 a{
	font-style: italic;
	font-family: Calibri, sans-serif;
	color:#aa0000;
}
```

Контекстные селекторы

```
<ul id = "nav-x">
	<li>Главная</li>
	<li>Товары</li>
	<li>Контакты</li>
</ul>

<ul id = "nav-y">
	<li>Главная</li>
	<li>Товары</li>
	<li>Контакты</li>
</ul>
```

Файл css

```
#nav-x li{
	border: 1px solid red;
	width: 100px;
}
```

Все теги li, которые лежат внутри списка nav-x, будут находиться в красной рамке.

Так можно через пробел писать несколько тегов. Пробел будет означать, что стили применяются ко всем тегам, перечисленным через пробел.

```
<ul id = "nav-x">
	<li><a href="">Главная</a></li>
	<li><a href="">Товары</a></li>
	<li><a href="">Контакты</a></li>
</ul>

<ul id = "nav-y">
	<li><a href="">Главная</a></li>
	<li><a href="">Товары</a></li>
	<li><a href="">Контакты</a></li>
</ul>

#nav-x li a{
	text-decoration: none;
	color: blue;
}
```

Горизонтальное меню

```
#nav-x li{
	border: 1px solid #aec55d;
	float:left;
	list-style: none;
	margin-right: 20px;
	margin-bottom:  40px;
	padding: 10px;
	background: #e9f0c0;
```

```
}
#nav-x li a{
	text-decoration: none;
	color: #2f1e0a;
}
#nav-y{
	clear:both;}
```

Дочерние селекторы

```
<ul id = "nav-y">
	<li><a href="">Главная</a>
		<ul>
			<li><a href="">О нас</a></li>
			<li><a href="">Слайдер</a></li>
			<li><a href="">Отзывы</a></li>
		</ul>
	</li>
	<li><a href="">Товары</a>
		<ul>
			<li><a href="">Одежда</a></li>
			<li><a href="">Обувь</a></li>
			<li><a href="">Компьютеры</a></li>
		</ul>
	</li>
	<li><a href="">Контакты</a>
		<ul>
			<li><a href="">Контактные данные</a></li>
			<li><a href="">Адрес</a></li>
			<li><a href="">Соц сети</a></li>
		</ul>
	</li>
</ul>

#nav-y>li>a{
	font-size: 20px;
}
```

Получаем только те теги li, которые только вложены в тег ul и потом только первые теги a. В результате будут большими буквами только названия списков первого уровня.

При помощи дочерних селекторов можно выбрать только те теги, которые являются прямыми потомками определенного элемента.

Группирование свойств – для группирования повторяющихся свойств. Теги, у которых повторяются свойства, записываются через запятую.

```
h1, h3, p {
	color: blue;
	text-align: center;}
```

3) Псевдоклассы и псевдоэлементы

Псевдоклассы и псевдоэлементы позволяют назначить CSS стили структурам, существование которых на веб-странице не обязательно, т.е. стили применяются к частям

документа не на основании той информации, которая содержится в его структуре, и даже изучив эту структуру, невозможно понять, что к этим элементам применяются какие-либо стили.

Псевдокласс в CSS – *это ключевое слово, добавленное к селектору,* которое задает стиль для его состояния.

- hover – состояние, когда навели мышку.
- active – элемент активен, нажат элемент.
- Visited – посещенная ссылка
- link – непосещенная ссылка
- focus – для разных элементов, например input

Синтаксис

```
тег: псевдокласс {
        свойство: значение;
}
```

```
#nav-x li a:hover{
        text-decoration: underline;
        color: #dcd294;
}
```

Псевдоэлементы

- first-letter – первая буква
- first-child – первый ребенок, первый пункт
- nth-child (2n) – n ребенок (в скобках формулы, odd, even)

```
li: first-letter {
        color: #454;
}
#cat li: nth-child (2n){
        color: #454;}
```

Лекция № 5. Таблицы и формы в HTML

План

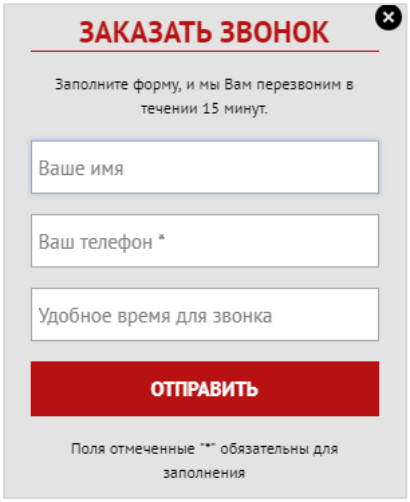
1. Формы на сайте. Тег <form>
2. Элементы (поля) форм
3. Таблицы

1) Формы на сайте. Тег <form>

Форма – один из самых необходимых элементов современных сайтов. Формы позволяют получать информацию от пользователя, отправлять данные на сервер или администратору на почту.

Примерами форм на сайте являются формы авторизации и регистрации, формы обратной связи, формы для заказа товара, формы подписки, формы голосования и т.д.

Форма авторизации



Форма обратной связи

Тег <form>

Для размещения формы на веб-странице используется контейнер <form> </form>. Тэг **<form>** может содержать атрибуты, описанные в таблице 1.

Таблица 1. Атрибуты тега <form>

№	*Атрибут*	*Описание, принимаемое значение*
1.	action	**Обязательный атрибут**. Определяет файл на сервере, который будет обрабатывать данную форму. Синтаксис: <form action = "action.php"> </form> Если значение атрибута не указано, то данные формы не будут отправлены и после перезагрузки страницы элементы формы примут значения по умолчанию.
2.	method	Определяет, каким образом данные из формы будут переданы обработчику. Допустимые значения: method="post" и method="get". Если значение атрибута не установлено, по умолчанию предполагается method="get".
3.	enctype	Определяет, каким образом данные из формы будут закодированы для передачи обработчику. enctype ="multipart/form-data" – данные не кодируются, это значение применяется при отправке файлов.
4.	name	Задает имя формы, которое будет использоваться для доступа к ее элементам.

Пример записи атрибутов тега <form> может быть следующим:

```
<form action = "index.php" method = "post" enctype ="multipart/form-data">

</form>
```

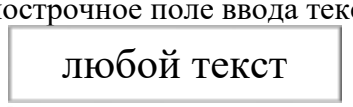
В данном случае обработка данных осуществляется в файле index.php, в котором используется серверный язык программирования php.

2) Элементы (поля) форм

Наиболее часто в формах используется тег **<input>**, позволяющий вводить и отправлять на сервер различные данные. Вид выводимого поля и тип передаваемых данных зависит от атрибута type.

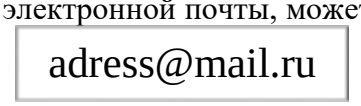
1) Текстовые поля

а) **текст: <input type="text">** - однострочное поле ввода текста;



б) **пароль: <input type="password">** - поле ввода пароля, символы заменяются точками;

в) **email: <input type="email">** - поле ввода электронной почты, может показывать предупреждение, если введён некорректный email;



г) **число <input type="number">** - поле ввода числа.

Для поля ввода числа также можно добавить атрибуты, описанные в таблице 2.

Таблица 2. Атрибуты тега <input type="number">

№	*Атрибут*	*Описание, принимаемое значение*
1.	min	Минимальное значение числа
2.	max	Максимальное значение числа
3.	step	Шаг приращения числа. Может быть целым (2) или дробным (0.5)
4.	value	Начальное число, которые выводится в поле
5.	size	Ширина поля

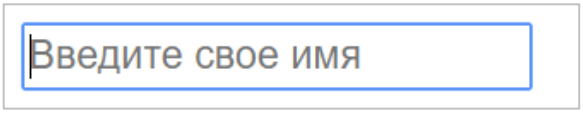
Для многих элементов формы, в том числе и для тега <input>, используются универсальные атрибуты, описанные в таблице 3.

Таблица 3. Универсальные атрибуты элементов форм

№	*Атрибут*	*Описание, принимаемое значение*
1.	**name**	**Имя поля, по которому обработчик формы идентифицирует элемент.**
2.	size	Ширина поля в символах.
3.	**value**	– **Начальный текст, отображаемый в поле.** – **Значение, отправляемое на сервер.**
4.	required	Обязательное поле. Выводит сообщение о том, что пользователь должен ввести данные или выбрать значение. Атрибут используется без указания значения.
5.	disabled	Поле, недоступное для ввода. Атрибут используется без указания значения.
6.	autofocus	Поле автоматически получает фокус после загрузки страницы. Атрибут используется без указания значения.

Текстовые поля могут отображать **подсказку**, которая исчезнет, как только будет введён некоторый текст. Для этого используется атрибут placeholder.

```
<form action = " ">
        <input type="text" placeholder="Введите свое имя">
</form>
```

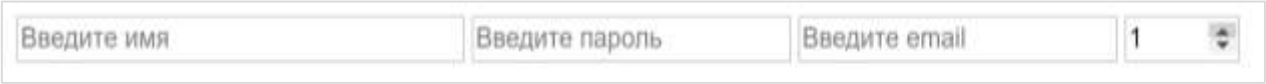


Текстовое поле с атрибутом placeholder

Если начать набирать данные в текстовом поле, то текст «Введите своё имя» исчезнет.

В следующем примере показан html-код формы с различными текстовыми полями и их атрибутами.

```
<form action = " ">
        <input type="text" size = "30" placeholder="Введите имя" >
        <input type="password" size = "20" placeholder="Введите пароль" >
        <input type="email" size = "20" placeholder="Введите email" >
        <input type="number" size = "5" min="1" max="20" step="1" value="1">
</form>
```



Результат в браузере

Текстовые поля, как и многие другие элементы формы, являются строчными элементами, поэтому для переноса поля на следующую строку следует использовать тег
 или свойства CSS. Также можно каждое текстовое поле обернуть в тег абзаца <p>.

Чтобы создать к элементам формы подсказки, видимые в любой момент, используются **текстовые метки <label> </label>**.

Метка <label> может быть связана с полем двумя способами:

1) Тег с полем помещается внутрь контейнера **<label> </label>.**

```
<label> Имя
        <input type="text">
</label>
```

2) С помощью атрибута for, значение которого соответствует значению id (идентификатор) у поля.

```
<label for="first_name">Имя </label>
<input type="text" id="first_name">
```

Текстовое поле с меткой

При щелчке по метке фокус переходит к текстовому полю и помещает курсор внутрь него. Эта связь особенно полезна при работе с флажками и переключателями.

2) Переключатели и флажки

Флажок («чекбокс») используют, когда необходимо выбрать любое количество вариантов из предложенного списка.

Для создания флажков используется тег: <input type="checkbox">.

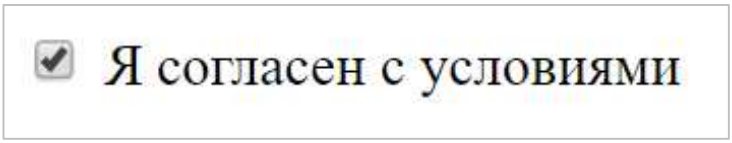
По небольшому флажку часто бывает сложно щелкнуть мышкой, поэтому рекомендуется помещать флажок и его описание внутрь метки <label>.

```
<label>
        <input type="checkbox"> Я согласен с условиями
</label>
```

В данном примере можно щёлкнуть по тексту «Я согласен с условиями», чтобы переключить флажок.

По умолчанию флажок выключен. Чтобы пометить его по умолчанию включенным, используется атрибут checked.

```
<label>
        <input type="checkbox" checked> Я согласен с условиями
</label>
```



Флажок

Переключатели используют, когда необходимо выбрать один единственный вариант из нескольких предложенных. Переключатели являются **взаимоисключающими**, т.е. выбор одного из вариантов исключает выбор другого.

Переключатели создаются с помощью тега: **<input type="radio">**.

Чтобы переключатели относились к одной группе, необходимо чтобы у них был одинаковый атрибут name, иначе их можно будет нажать одновременно.

По умолчанию пункты переключателя выключены. Чтобы пометить какой-либо пункт включенным, применяется атрибут checked.

Для переключателей и флажков используется атрибут value, который показывает, какое значение будет отправлено на сервер (какой пункт был выбран пользователем).

```
<label>Выберите Ваш пол</label>
<label>
        <input type="radio" name="gender" value="men" checked> Мужской
</label>
<label>
        <input type="radio" name="gender" value="women"> Женский
</label>
```

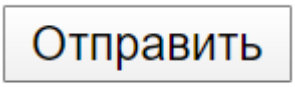
В данном примере оба переключателя используют одинаковое значение атрибута name (значение gender), выбор одного из вариантов исключит выбор другого варианта. При выборе пункта *Мужской* на сервер будет отправлено значение men, при выборе пункта *Женский* – значение women.

3) Кнопки

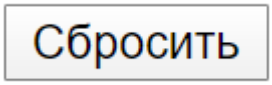
Кнопку на веб-странице можно создать двумя способами – с помощью тега **<input>** и парного тега **<button></button>**.

При добавлении кнопки через <input> возможно создание следующих типов кнопок:

а) кнопка отправки данных на сервер: **<input type="submit">**;



б) кнопка сброса введенных данных: **<input type="reset">**;



в) обычная кнопка, которая не имеет поведения по умолчанию: **<input type="button">**.

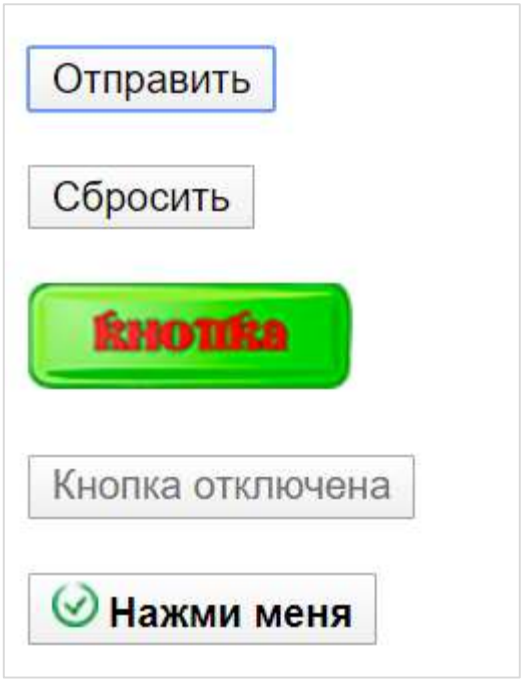
Такую кнопку можно использовать для запуска сценария JavaScript.

Второй способ создания кнопки основан на использовании контейнера **<button></button>**. Для корректного отображения элемента <button> разными браузерами у него нужно указывать атрибут type, например:

```
<button type="submit"> </button>
```

В следующем примере показан html-код создания различных кнопок.

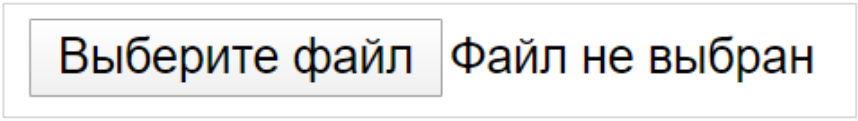
```
<input type="submit" name = "button1" autofocus> <br>
<input type="reset" name = "button2"> <br>
<input type="image" name = "button3" src="knp.png"> <br>
<input type="button" name = "button4" value="Кнопка отключена" disabled> <br><br>
<button type = "button" name = "button5">
    <img src = "photo.png">
    <b>Нажми меня</b>
</button>
```



Кнопки

4) Поле загрузки файла

Чтобы загружать на сервер один или несколько файлов используется специальное поле: **<input type="file">.**



Вид поля для загрузки файла в Chrome

При нажатии на кнопку открывается окно для выбора файла, где можно указать, какой файл пользователь желает использовать.

Чтобы поле заработало и браузер смог предать выбранный файл на сервер, необходимо для тега формы:

1. задать метод отправки данных POST (method="post");
2. установить у атрибута enctype значение multipart/form-data.

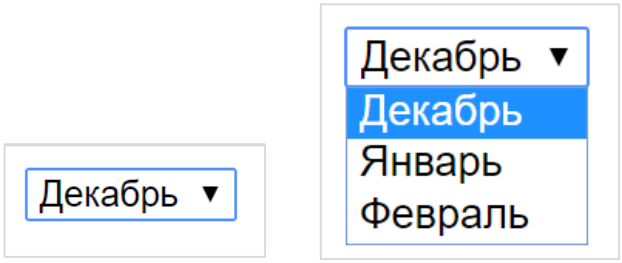
Далее рассмотрим другие теги для создания полей формы.

5) Выпадающий список

Выпадающий список используется, если количество вариантов для выбора достаточно много. Выпадающий список работает подобно переключателям, отличаясь от них только внешним видом.

Для создания выпадающего списка используется контейнер **<selest></select>**. Элементы выпадающего списка указываются в контейнерах <option></option>.

```
<select>
        <option> Декабрь </option>
        <option> Январь </option>
        <option> Февраль </option>
</select>
```



Выпадающий список

По умолчанию в поле списка отображается его первый элемент. Атрибуты тегов <select> и <option> приведены в таблицах 6 и 7.

Таблица 6. Атрибуты тега <select>

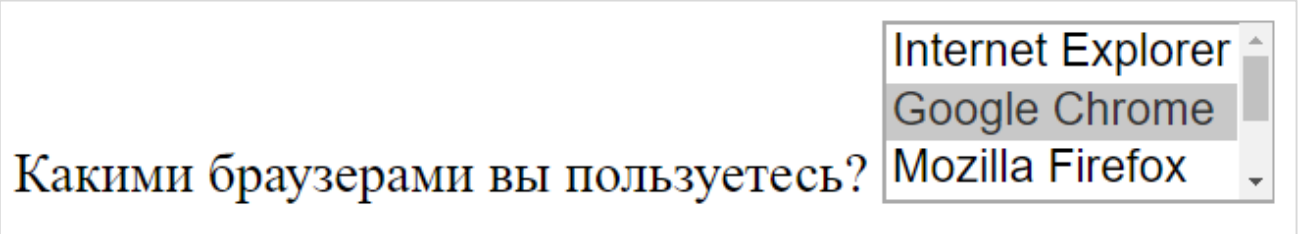
№	*Атрибут*	*Описание, принимаемое значение*
1.	multiple	Дает возможность выбора нескольких пунктов. При выборе нужно нажать и удерживать нажатой клавишу Ctrl. Атрибут используется без указания значения.
2.	size	Задает количество отображаемых строк списка. Значение атрибута задается целым положительным числом.

Таблица 7. Атрибуты тега <option>

№	*Атрибут*	*Описание, принимаемое значение*
1.	selected	Отображает выбранный элемент списка по умолчанию при загрузке веб-страницы браузером. Атрибут используется без указания значения.
2.	value	Указывает значение, отправляемое на сервер

Пример использования тега <select>:

```
<label>Какими браузерами вы пользуетесь? </label>
<select multiple size = "3">
        <option value ="explorer"> Internet Explorer </option>
        <option value ="chrome" selected> Google Chrome </option>
        <option value ="firefox"> Mozilla Firefox </option>
        <option value ="opera"> Opera </option>
        <option value ="safari"> Safari </option>
</select>
```



Выпадающий список множественного выбора

6) Текстовая область

Если необходимо создать многострочное поле для ввода большого текста, то вместо тега <input> используется элемент **<textarea></textarea>.**

Таблица 8. Атрибуты тега <textarea>

№	*Атрибут*	*Описание, принимаемое значение*

1.	cols	Ширина текстового поля в символах.
2.	rows	Высота текстового поля в строках текста.
3.	wrap	Определяет параметры переноса строк в тексте, при отправке данных на сервер: – wrap = "hard" – сохраняет перенос, обязательным условием использования значения hard является установленный атрибут cols; – wrap = "soft" – не сохраняет перенос (значение по умолчанию).
4.	maxlength	Максимальное число символов, которое можно ввести в текстовое поле.
5.	readonly	Указывает, что текстовое поле будет доступно только для чтения, т.е. текст невозможно будет изменить, но будет возможность его скопировать. Атрибут используется без указания значения.
6.	disabled	Отключает возможность редактирования и копирования содержимого поля. Атрибут используется без указания значения.

3) Таблицы

Для чего нужны таблицы

Основное назначение таблиц в HTML - это представление табличных данных, таких как: просмотр информации о пользователях, просмотр заказанных товаров в интернет-магазине, просмотр отчетов о продажах и многое другое.

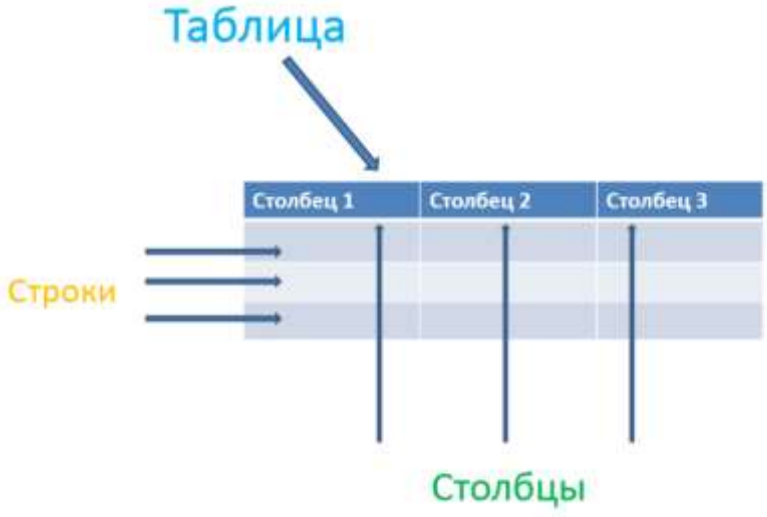
Второе назначение таблиц в html – это то, что при помощи таблиц, можно верстать веб-страницы. А верстать - это значит разбивать нашу страницу на элементы, т.е. формировать различные блоки нашего сайта, такие как шапка, меню, контент, подвал и другие элементы. Табличный способ верстки в настоящее время считается неправильным и устаревшим, потому что таблицы используются не по назначению, но многие сайты либо уже сверстаны при помощи таблиц, либо продолжают верстаться этим способом.

Создание таблиц

Таблица в html внешне никак не будет отличаться от таблиц, созданных в Word или Excel.

Таблица в html, состоит из некоторого количества столбцов и строк, которые формируют ячейки в таблице.

Структура таблицы



Структура таблицы в HTML

```
<table>
	<tr>
		<td> </td>
		<td> </td>
	</tr>
</table>
```

<table> </table> - контейнер таблицы
<tr> <tr> - строка таблицы
<td> <td> - столбец (или ячейка) таблицы
<th> <th> - то же самое, что и тег <td>, только текст в этой ячейке будет полужирным и выровненным по центру. Тег <th> часто используется для создания шапки в таблице.

В следующем примере показан html-код таблицы с 2 строками и 3 столбцами.

```
<table>
	<tr>
		<td>Столбец 1</td>
		<td>Столбец 2</td>
		<td>Столбец 3</td>
	</tr>
	<tr>
		<td>Столбец 1</td>
		<td>Столбец 2</td>
		<td>Столбец 3</td>
	</tr>
</table>
```

Если просмотреть данный пример в браузере, то Вы не увидите границ таблицы.

Столбец 1 Столбец 2 Столбец 3
Столбец 1 Столбец 2 Столбец 3

Результат в браузере

Для задания границ таблицы необходимо к html-файлу подключить css-файл со следующими стилями:

Файл style.css

```
table {
	border: 2px solid black;
	border-collapse: collapse; /*убираем двойные рамки*/
}
td {
	border: 1px solid black;}
```

Столбец 1	Столбец 2	Столбец 3
Столбец 1	Столбец 2	Столбец 3

Результат в браузере

Чтобы вставить еще одну строку в таблицу, нужно добавить контейнер **<tr> </tr>** с таким же количеством столбцов, что было ранее. Чтобы вставить еще один столбец в таблицу, нужно добавить контейнер **<td> </td>** в каждую строку.

При помощи тега **<caption> <caption>** формируется название таблицы. Этот тег помещается в контейнер таблицы <table>. Название по умолчанию располагается над таблицей и выравнивается посередине

В следующем примере показан html-код таблицы с заголовком, состоящей из 3 строк и 4 столбцов. Первая строка таблицы оформлена полужирным шрифтом с выравниванием по центру.

```
<table>
	<caption>Пример таблицы</caption>
	<tr>
		<th>Столбец 1</th>
		<th>Столбец 2</th>
		<th>Столбец 3</th>
		<th>Столбец 4</th>
	</tr>
	<tr>
		<td>Столбец 1</td>
		<td>Столбец 2</td>
		<td>Столбец 3</td>
		<td>Столбец 4</td>
	</tr>
	<tr>
		<td>Столбец 1</td>
		<td>Столбец 2</td>
		<td>Столбец 3</td>
		<td>Столбец 4</td>
	</tr>
</table>
```

Пример таблицы

Столбец 1	**Столбец 2**	**Столбец 3**	**Столбец 4**
Столбец 1	Столбец 2	Столбец 3	Столбец 4
Столбец 1	Столбец 2	Столбец 3	Столбец 4

Результат в браузере

Объединение ячеек

Для объединения ячеек используются следующие атрибуты:

rowspan – объединение строк	**colspan** – объединение столбцов	

В значении этих атрибутов указывается целым положительным числом количество объединяемых ячеек.

Рассмотрим пример с 3 столбцами и 3 строками. У первой ячейки добавим атрибут rowspan="2", объединяющий две строки.

```
<table>
	<tr>
		<td rowspan="2">Объединение строк</td>
		<td> Столбец 2 строки 1 </td>
		<td> Столбец 3 строки 1 </td>
	</tr>
	<tr>
		<td> Столбец 1 строки 2</td>
		<td> Столбец 2 строки 2</td>
		<td> Столбец 3 строки 2 </td>
	</tr>
	<tr>
		<td> Столбец 1 строки 3</td>
		<td> Столбец 2 строки 3 </td>
		<td> Столбец 3 строки 3 </td>
	</tr>
</table>
```

Объединение строк	Столбец 2 строки 1	Столбец 3 строки 1	
	Столбец 1 строки 2	Столбец 2 строки 2	Столбец 3 строки 2
Столбец 1 строки 3	Столбец 2 строки 3	Столбец 3 строки 3	

Результат в браузере

Результат в браузере получился не совсем правильным: ячейки второй строки вышли за пределы таблицы. Чтобы это исправить, необходимо во второй строке удалить первую ячейку:

```
<table>
	<tr>
		<td rowspan="2">Объединение строк</td>
		<td> Столбец 2 строки 1 </td>
		<td> Столбец 3 строки 1 </td>
	</tr>
	<tr>
		<td> Столбец 2 строки 2</td>
		<td> Столбец 3 строки 2 </td>
	</tr>
	<tr>
		<td> Столбец 1 строки 3</td>
		<td> Столбец 2 строки 3 </td>
		<td> Столбец 3 строки 3 </td>
	</tr>
</table>
```

Объединение строк	Столбец 2 строки 1	Столбец 3 строки 1
	Столбец 2 строки 2	Столбец 3 строки 2
Столбец 1 строки 3	Столбец 2 строки 3	Столбец 3 строки 3

Результат в браузере

Аналогично можно сделать объединение столбцов:

```
<table>
	<tr>
		<td rowspan="2">Объединение строк</td>
		<td colspan="2">Объединение столбцов</td>
	</tr>
	<tr>
		<td> Столбец 2 строки 2</td>
		<td> Столбец 3 строки 2 </td>
	</tr>
	<tr>
		<td> Столбец 1 строки 3</td>
		<td> Столбец 2 строки 3 </td>
		<td> Столбец 3 строки 3 </td>
	</tr>
</table>
```

Объединение строк	Объединение столбцов	
	Столбец 2 строки 2	Столбец 3 строки 2
Столбец 1 строки 3	Столбец 2 строки 3	Столбец 3 строки 3

Результат в браузере

Вложенные таблицы

В любую ячейку таблицы можно вложить еще одну таблицу, соблюдая ту же структуру.

```
<table>
	<tr>
		<td>
			<table>
				<tr>
					<td>Столбец 1 вложенной таблицы</td>
					<td> Столбец 2 вложенной таблицы</td>
				</tr>
			</table>
		</td>
		<td> Столбец 2</td>
		<td> Столбец 3</td>
	</tr>
	<tr>
		<td>Столбец 1</td>
		<td>Столбец 2</td>
```

```
            <td>Столбец 3</td>
        </tr>
</table>
```

Лекция № 6. Строчные и блочные элементы. Свойства display и float

План

1. Строчные и блочные элементы
2. CSS-свойство display
3. Обтекание элементов

1. Строчные и блочные элементы

Блочным называется элемент, который отображается на веб-странице в виде прямоугольника. Такой элемент занимает всю доступную ширину, высота элемента определяется его содержимым, и он всегда начинается с новой строки

- Заголовки <h1>
- Абзац <p>
- Таблица <table>
- Список
- Формы <form>
- Блок (контейнер)

Блочные элементы предназначены для **структурирования** основных частей веб-страницы.

Ширина **строчных** элементов равна их содержимому. Они автоматически не переходят на новую строку.

- Ссылка <a>
- Жирный текст <b>
- Курсив <i>
- Изображение <img>
- Поле ввода <input>
- Строчный элемент

Строчные элементы предназначены, чтобы разграничить часть текста и придать ему определённую функцию или смысл.

Блочно-строчные элементы: img, input – можно задавать размеры как блочным элементам, но при этом они не переходят на новую строку.

Тег ** - блок, используется для блочной верстки сайта.**

Пример 1 (блок, внутри него заголовок, текст и 2 ссылки) – показать размеры элементов, отступы.

Пример 2 – добавить пустой div и задать для него ширину, высоту и заливку.

Пример 3 – сброс стилей, свойство box-sizing: border-box;

Пример 4 – span вокруг нескольких слов.

```html
<div class="block1">
    <h1>заголовок</h1>
    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.
    <span>Harum nobis ipsum fuga</span> odio veritatis temporibus earum molestias
    ducimus sapiente nesciunt corporis, enim suscipit fugit quaerat. Quod ea provident impedit!
    Repudiandae!</p>
    <a href="">Ссылка1</a>
    <a href="">Ссылка2</a>
</div>
<div class="block2">  </div>
```

```css
*{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
.block1{
    width: 100%;
    height: 200px;
    background: rgb(216, 0, 6);
    padding: 50px;
}
.block2{
    width: 100%;
    height: 200px;
    background: rgb(216, 136, 136);
}
span{
    color: blue;
}
```

2. CSS-свойство display

Свойство display позволяет изменять тип элемента.

Значения свойства display:

- display: none – выключает блок из документа
- display: block – блочное отображение элемента (строчное делает блочным)
- display: inline – строчное отображение элемента (блочное делает строчным)
- display: inline-block – строчно-блочное отображение (блочное делает строчным, но позволяет задавать размеры)
- display: flex – адаптивный flex-блок для дочерних элементов
- display: grid – блочный grid-контейнер в виде сетки
- display: table-cell – отображение как у ячейки таблицы

Пример 1 с меню (inline и inline-block).
Пример 2 из ссылки сделать кнопку.

3. Обтекание элементов

Свойство float позволяет задать обтекание у элемента.

Значения свойства float:

- float: left – обтекание слева
- float: right – обтекание справа
- overflow: hidden; - прячет перекрытие
- clear: both; - отмена обтекания

Пример с картинками и текстом.

Лекция № 7. Позиционирование элементов. Формирование блочной модели сайта

План

1. CSS-свойство position - позиционирование элементов

2. Блочная верстка сайта

1) CSS-свойство position - позиционирование элементов

Позиционирование позволяет точно определить, где появятся блоки относительно другого элемента.

Значения свойства Position:

- position: static – статическое позиционирование – элементы могут обтекать друг друга (значение по умолчанию). Чтобы подвинуть элемент нужно задать для него отступ от других элементов или краев.
- position: relative – относительное позиционирование. Элемент можно смещать относительно его позиции с помощью свойств смещения.
- position: absolute – абсолютное позиционирование. Элемент можно смещать относительно левого верхнего края страницы или родительского элемента с помощью свойств смещения.
- position: fixed – фиксированное позиционирование

Абсолютное позиционирование – если посредине блок вынуть, то его можно вставить в любое место страницы, а блоки при этом сомкнутся.

Свойства смещения:

- **Left – смещение слева**
- **Top – смещение сверху**
- **Right – смещение справа**
- **Bottom – смещение снизу**

Можно задавать у смещений положительные и отрицательные значения.

Свойство z-index: -1; /*перемещение блока под элемент*/

Статичное позиционирование – смещение с помощью отступов

```
<div class="block1"></div>
<div class="block2"></div>
<div class="block3"></div>

.block1{
   width: 400px;
   height: 400px;
   background: rgb(83, 129, 86);
}
.block2{
   width: 250px;
   height: 250px;
   background: rgb(241, 239, 82);
   margin-left: 500px;
}
.block3{
   width: 100px;
   height: 100px;
   background: rgb(101, 92, 230);
}
```

Относительное позиционирование – относительно начальной позиции

```
.block2{
   width: 250px;
   height: 250px;
   background: rgb(241, 239, 82);
   position: relative;
   left:500px;
   top:200px;
}
```

Абсолютное позиционирование – относительно начала координат

```
.block2{
   width: 250px;
   height: 250px;
   background: rgb(241, 239, 82);
   position: absolute;
   left:500px;
   top:200px;
}

<div class="block1">
      <div class="block2"></div>

      <div class="block3"></div>
</div>
```

Абсолютное позиционирование – относительно родительского блока

```
.block1{
   width: 400px;
   height: 400px;
   background: rgb(83, 129, 86);
   position: relative;
   left: 500px;
}
.block2{
   width: 250px;
   height: 250px;
   background: rgb(241, 239, 82);
   position: absolute;
   left:100px;
   top:100px;
}
.block3{
   width: 100px;
   height: 100px;
   background: rgb(101, 92, 230);
}
```

2) Блочная верстка сайта

Верстка сайта – это создание структуры сайта.

Основным тегом для верстки является тег ** ** , с помощью которого можно делить web-страницу на блоки. Внутри данного тега можно разместить другие HTML теги, которые требуются для создания содержимого сайта (ссылки, текст, изображения и тд.).

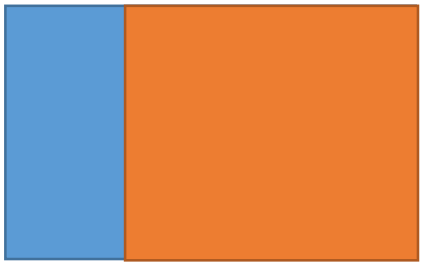
Примеры блочной верстки сайта

Верстка сайта в 2 колонки:

```
b
o
d
y
        <div id = "left"> Это левый блок

        <div id = "right"> Это правый блок


clear"> Отмена обтекания </div>
</body>
```



```
#left  {
        float: left;
        width: 30%;}
#right  {
        float: right;
        width: 70%;}
#clear {
        clear: both; }
```

Верстка сайта в 3 колонки:

```
b
o
d
y
        <div id = "left"> Это левый блок

        <div id = "right"> Это правый блок


        <div id = "center"> Это центр </div>
</body>
```

```
#left  {
        float: left;
        width: 30%;}
#right  {
        float: right;
        width: 20%;}
#center  {
        overflow: hidden;
        width: 50%;}
```

Пример использования обтекания


```
<body>
        <div id = "header">
                <img src="logo.png" id="logo">

d
</body>
h1>Мой сайт</h1>
v
```

```
#logo  { float: left; }
```

Пример использования позиционирования


```
<body>
        <div id = "header">
                <input type="submit"
                value="Регистрация">
d
i</body>
```

```
#header {
        position: relative;}

#header input {
        position: absolute;
        top: 10px;
        right: 20px;
}
```

Способы создания горизонтального меню

```
<body>
        <div id = "nav">
                <ul>

                   <li><a href="">Home</a></li>

                   <li><a href="">Work</a></li>

                   <li><a href="">About</a></li>

                   ...

                </ul>

d
```

2 способ – float: left

```
ul {
        list-style: none;}
ul li {
        display: inline;
}
```

```
ul {
        list-style: none;}
ul li {
        float: left;
}
```

Лекция № 8. Верстка сайта на FlexBox

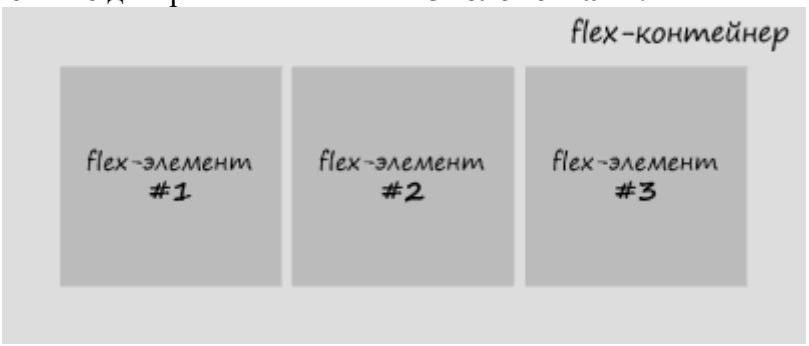
План
1. Назначение Flexbox. Схема устройства flex-контейнера
2. Свойства родительских элементов (flex-container)
3. Свойства дочерних элементов

1) Назначение Flexbox. Схема устройства flex-контейнера

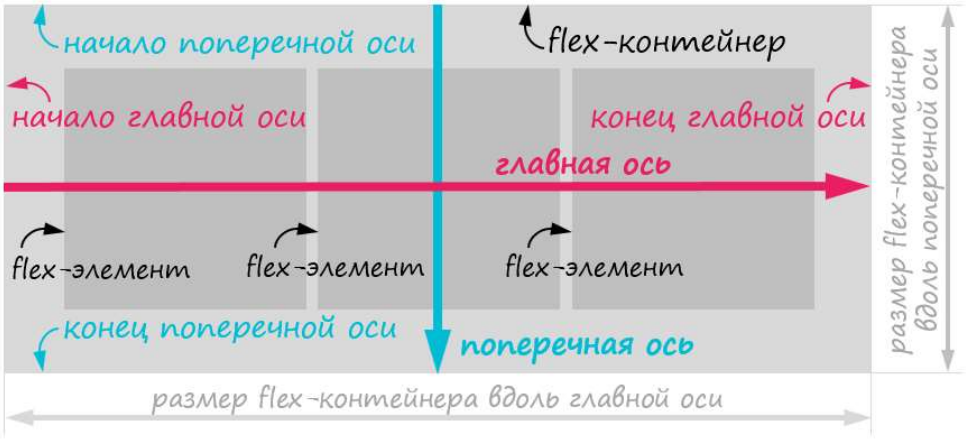
CSS Flexbox предназначен для создания гибких макетов. С помощью этой технологии можно очень просто и гибко расставить элементы в контейнере, распределить доступное пространство между ними, и выровнять их тем или иным способом даже если они не имеют конкретных размеров. CSS Flexbox позволяет создать адаптивный дизайн намного проще, чем с использованием Float и позиционирования.

Создание CSS разметки с помощью Flexbox начинается с установки необходимому HTML элементу CSS-свойства **display** со значением **flex** или **flex-inline**.

После этого данный элемент становится **flex-контейнером**, а все его **непосредственные** дочерние элементы – **flex-элементами**.



На рисунке представлена схема устройства flex-контейнера:



2) Свойства родительских элементов (flex-container)

1) display - создаёт flex контейнер, инлайновый или блочный, в зависимости от заданного значения.

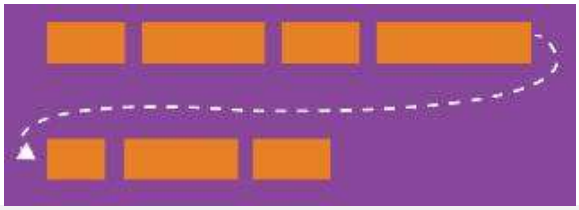
```
.container {
  display: flex; /* или inline-flex */
}
```

2) flex-direction - устанавливает направление главной оси и определяет направление flex элементов размещенных в flex контейнере

Значения:

row (стандартное положение) — слева направо.
row-reverse — элементы располагаются справа налево.
column — тоже самое, что и row, только сверху вниз.
column-reverse — тоже самое, что и row-reverse, но снизу вверх.

3) flex-wrap - дает возможность дочерним элементам при необходимости переходить на другую строку



Значения:

nowrap — это значение по-дефолту, при котором все flex элементы будут выстраиваться в одну линию.
wrap — flex элементы будут переноситься на несколько строк, от верха к низу.
wrap-reverse — flex элементы будут переноситься на несколько строк снизу вверх.

4) justify-content – выравнивание дочерних элементов вдоль главной оси

Значения:

flex-start — дефолтное состояние, при котором элементы расставляются от начала строки.
flex-end — состояние, в котором элементы размещены с конца строки.
center — элементы центрированы вдоль строки.
space-between — элементы равномерно распределены по строке, первый элемент находится

вначале строки, последний в конце.
space-around — элементы равномерно распределены по строке с равным местом вокруг них.
space-evenly — элементы распределены таким образом, что свободное пространство между любыми двумя элементами равномерно, как и место до границы края контейнера.

5) align-items – выравнивание дочерних элементов вдоль поперечной оси
Значения:
flex-start — всё размещается с начала поперечной оси
flex-end — все элементы размещаются с конца поперечной оси
center — элементы центрируются по поперечной оси
baseline — элементы выравниваются по базовой линии
stretch — это дефолтное состояние, при котором элементы заполняют контейнер, с учетом min-width и max-width.

6) align-content - выравнивает и распределяет строки контейнера, когда есть свободное пространство в поперечной оси. Это свойство не приносит эффекта, когда есть только одна строка flex элементов

```
.container {
  align-content: flex-start | flex-end | center | space-between | space-around | stretch;
}
```

3) Свойства дочерних элементов

- **Order** - определяет порядок, в котором будут располагаться дочерние элементы. Задается целым числом и по умолчанию равно 0.
- **Flex-grow** - указывает, насколько отдельный элемент будет больше соседних элементов (по умолчанию 0).
- **Flex-shrink** - определяет способность flex-элемента сокращаться в случае недостатка свободного места. По умолчанию равен 1.
- **Flex-basis** - базовый размер отдельного элемента, заменяет свойство width
- **Flex** является сокращенным свойством для задания свойств flex-grow, flex-shrink и flex-basis

  flex: 0 1 auto; /*по умолчанию*/
- **Align-self** - выравнивание отдельно взятого flex-блока по поперечной оси.

Лекция № 10. Кроссбраузерность. Основы адаптивной верстки. Публикация сайта в сети Интернет

План
1. Кроссбраузерность
2. Новые теги HTML 5 и стили CSS 3
3. Основы адаптивной верстки
4. Публикация сайта в интернете

1. Кроссбраузерность

В процессе своей работы любой веб-разработчик сталкивается с таким понятием, как кроссбраузерность. Это понятие означает, что любой сайт должен выглядеть одинаково во всех браузерах.

Кроссбраузерность – это свойство сайта отображаться и работать во всех популярных браузерах одинаково.

Главным виновником этой проблемы является браузер Internet Explorer (IE).

Пути решения проблемы кроссбраузерности:

а) Тестировать во всех основных браузерах

При этом, лучше использовать последнюю версию HTML 5. Для этого в начале кода пишется строка: <!DOCTYPE html>

Необходимо проверять, как выглядит сайт во всех популярных браузерах с некоторой периодичностью во время верстки. Тогда вы сможете оперативно устранять несоответствия.

б) Сбросить в начале работы стили css.

Некоторые теги HTML, например, заголовки, параграфы, списки, изначально имеют определенный набор свойств и значений. Эти свойства могут определяться каждым браузером по-разному. Для того, чтобы вид HTML страницы не зависел от того, с помощью какого браузера ее просматривают, используется сброс стилей CSS.

Обычно, если используется сброс стилей, то их определяют в отдельном файле (reset.css), который подключают на веб-страницу до основного файла со стилями. Готовый файл reset.css, в котором через запятую перечислены все теги, которые нужно обнулить, можно найти в сети Интернет.

в) Объявлять условный комментарий

Условный комментарий представляет собой обычный HTML комментарий, только в квадратных скобках указывается условие, которое понимает только браузер IE, а всеми остальными браузерами игнорируется.

Internet Explorer читает код в комментариях, поэтому для ранних версий можно писать следующие строки:

```
<!-- [if (IE 6) I (IE 7)]>
<link href = “css/style_ie.css” />
<![endif] -->
```

То, что находится между if и endif Internet Explorer выполнит. В данном случае для IE 6 или 7 подключится файл новый стилей CSS. В этом файле можно писать только те стили CSS, которые будут относиться к IE.

2. Новые теги HTML 5 и стили CSS 3

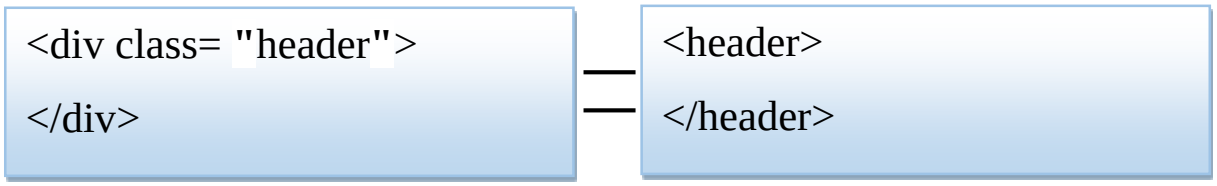
А) Семантические теги

В HTML 5 используются специальные семантические теги, которые ведут себя как тег блока div.

Семантические теги – это теги, которые предназначены для того чтобы компьютерные программы (поисковые системы, сборщики информации, речевые браузеры и т.д.), понимали какой тип информации заложен в данных тегах.

Примеры семантических тегов:

<header> </header> - шапка
<nav> </nav> - горизонтальное меню
<aside> </aside> - левый или правый блок
<section> </section> - контент
<footer> </footer> - подвал



Разницы между тегами ** ** и **<header>** при верстке сайта нет. Однако, например, для поисковых систем, разница будет большой. Поисковые машины или роботы не понимают ** **, для них это типовой тег разметки – замени его на ** ** и смысл не поменяется. Другое дело **<header>**, робот, обнаружив этот тег, воспринимает его именно как шапку сайта или раздела.

Б) Новые атрибуты у тегов форм

<input type = "email"> - ввод электронной почты
<input type = "date"> - календарь
<input type = "source"> - поиск
<input type = "number"> - переключатель
<input type = "range"> - ползунок

В) Теги для дополнительных технологий и API

<audio> </audio> – воспроизведение аудиофайлов без использования сторонних программ (плагинов, расширений).
<video> </video> – воспроизведение видеофайлов, без использования сторонних программ.
<source /> – указание пути к аудио/видео файлам, находится внутри тегов audio и video.
<canvas> </canvas> – создание специальной области на сайте, в которой можно создавать или добавлять изображения и, с помощью языка программирования JavaScript, манипулировать ими.

Новые свойства CSS3

1. **Закругление углов у блока** - свойство **border-radius**

```
border-radius: 2px; /*закругление для всех углов 2px*/
border-radius: 2px 5px 5px 2px; /*разные закругления у углов*/
border-radius: 5px/10px; /*закругление со скосом*/
```

2. **Градиентная заливка –** использование у свойства **background** или **background-image** значений: **linear-gradient()** или **radial-gradient()**

```
background: linear-gradient(to right, aqua, yellow); /*линейный градиент вправо от голубого к желтому*/
```

3. **Прозрачность элемента** - свойство **opacity**

```
opacity: 0; /*полностью прозрачный элемент*/
opacity: 1; /*полностью непрозрачный элемент*/
```

Удобный эффект, находит массу применений - полупрозрачный фон, изменение прозрачности при наведении и т.п.

4. **Тени**

Тени имеют следующие параметры: смещение по горизонтали, по вертикали, размытие и цвет
Свойство **box-shadow – тени у блоков**
Свойство **text-shadow – тень у текста**

```
box-shadow: 10px 5px 5px #000; /*тень смещена по оси X на 10px, по оси Y на 5px, радиус размытия 5px, черного цвета*/
```

5. **Плавные переходы –** свойство **transition**

Свойство **transition** включает 4 параметра:

- **transition-property** — указывает, для какого стиля будет действовать переход;
- **transition-duration** — определяет длительность анимации;
- **transition-timing-function** — скорость хода анимации;
- **transition-delay** — время ожидания перед началом перехода.

```
transition: all 1s ease 0s; /*переход для всех свойств длительностью 1s,  начало анимации медленное, к середине скорость повышается и в конце вновь снижается, задержка анимации 0s*/
```

6. **Трансформация объектов –** свойство **transform**

Перемещение объекта – transform: translate()

```
transform: translate(10px, 5px); /*объект сдвинется на 10px вправо и на 5px вниз*/
```

Чтобы сдвинуть в противоположную сторону, надо добавить значениям знак "-" (например, -10px)

Поворот объекта – transform: rotate()

```
transform: rotate(90deg); /*поворот объекта по часовой стрелке на 90^0*/
```

Если поставить со знаком "-", то поворот будет против часовой

Изменение размера у объекта – transform: scale()

```
transform: scale(1.5); /*увеличение объекта в полтора раза*/
```

Если поставить "-", то объект уменьшится на указанное значение

Новые свойства нужно использовать с префиксами (приставками).

Префиксы для браузеров:

-moz - мазилла
-webkit – вебкитовские браузеры (Chrome, Safari)
-ms – интернет эксплорер
-o – опера

```
- webkit-transform: rotate(90deg);
-moz-transform: rotate(90deg);
transform: rotate(90deg);
```

3) Основы адаптивной верстки

Адаптивная верстка – html-верстка веб-страниц, при которой внешний вид, расположение и структура ее элементов зависят от типа и свойств устройства на котором отображается страница.

Как сделать адаптивный дизайн сайта из фиксированного макета?

1. Необходимо добавить мета-тег ***viewport.*** Данный мета-тег прописывается в блоке <head> сайта.

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

2. Ширину основного контейнера необходимо задать с помощью свойства ***max-width.***

3. Необходимо перевести все статические единицы измерения в относительные единицы измерения: ***px необходимо перевести в %, а шрифты задать в em.***

Перевод ширины контейнера или элемента из px в % осуществляется по формуле:

Размер контейнера (px) / размер основного контейнера (родителя) в (px) * 100% = результат **(%)**.

Пример
Размер основного контейнера 960px, в нем имеется контейнер 720px. Для него получим размер в % следующим образом: 720/960*100=75%.

Перевод шрифта из px в em осуществляется по формуле:
Размер шрифта (px) /16px (стандартный размер) = размер шрифта (em)
Пример
Размер шрифта 32px, тогда 32/16=2em.

4. Необходимо использовать **медиа-запросы**.

Медиа-запросы включают в себя медиа-тип (принтеры, смартфоны, экраны, телевизоры, проекторы и др.) и условия, которое может принимать в свою очередь истину или ложь (true, false). В зависимости от того верный ли медиа-тип и выполняется ли условие будут применяться различные стили css. Если условие верно, то будут применяться те стили, которые прописаны в этом медиа-запросе, если же будет ложным, то будут применяться обычные стили css.

Благодаря таким запросам и создаются различные отображения сайта для мобильных, планшетов и экранов мониторов. Медиа-запросы поддерживаются всеми современными браузерами.

Медиа-запрос записывается следующим образом:

```
@media screen and (max-width: 1000px) {
.class {
        свойство: значение;
        }
}
```

Здесь:
@media – медиа-запрос;
screen – медиа-тип (также называют тип носителя);
max-width: 1000px – условие, которое должно выполняться (в данном примере стили будут применяться, если ширина окна меньше ширины 1000px);

4) Публикация сайта в интернете

Хостинг — услуга по предоставлению места для физического размещения информации на сервере, постоянно подключенном к интернету.

Если необходимо разместить сайт – надо найти для нее место (как для книги на полке), в интернете такие места предоставляют специальные службы – **хостеры**. Они предоставят место для сайта на своем сервере – машине, на которой установлены специальные программы, и которая постоянно подключена к сети. Собственно, интернет и состоит из множества таких машин объединенных между собой.

Существует два способа размещения сайта в интернете: бесплатный и платный.

Бесплатный хостинг позволяет бесплатно выбрать доменное имя третьего уровня для своего сайта и выложить сайт в интернет. Бесплатный хостинг предполагает размещение на сайте рекламы со стороны хостинг-провайдера, которая «немного» искажает дизайн вашего сайта, и возможности у бесплатного варианта ограничены. Вы никогда не будете иметь большую посещаемость ресурса, расположенного на бесплатном хостинге, отсюда следует и невозможность заработать на таком сайте, если, конечно, вы ставите перед собой такую цель.

Доменное имя сайта — это имя, которое вы указываете в поисковой строке браузера, например,www.my1site.ru или my1site.ru. Доменным именем первого уровня в данной записи является окончание .ru, второй уровень — это предыдущее название web-сайта: my1site. Доменное имя третьего уровня, которое предлагает бесплатный хостинг может выглядеть, к примеру, так: photo.my1site.ru или mypage.my1site.ru. Третьим уровнем доменного имени в данном случае является имя, которое предшествует my1site.ru

Платный хостинг дает возможность выбора доменного имени первого и второго уровня. Платный хостинг — это, когда вы платите деньги за место для своего web-сайта в сети интернет, но при этом являетесь полноправным хозяином своего ресурса, то есть никто не станет завешивать ваш сайт рекламой, пренебрегая дизайном.

ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ

Основная литература:

Полуэктова Н. Р. Разработка веб-приложений : учебное пособие для среднего профессионального образования / Н. Р. Полуэктова. — Москва : Издательство Юрайт, 2022. — 204 с. (*электронно-библиотечная система* https://urait.ru/)

Интернет-ресурсы:

1. Учебники по HTML и CSS [Электронный ресурс]. Форма доступа: http://htmlbook.name.
2. <MyRusakov.ru/>. Уроки и статьи по созданию сайтов [Электронный ресурс]. Форма доступа: http://MyRusakov.ru.
3. Ruseller.com. Частная коллекция качественных материалов для тех, кто делает сайты [Электронный ресурс]. Форма доступа: http://ruseller.com/