Лабораторная работа 1.1.

# HTML 5. Основы синтаксиса и семантики.

## 1. Элементы HTML5

Веб-страница состоит из ряда элементов. Некоторые элементы пусты, другие содержат текст, а третьи включают в себя другие элементы (либо и элементы и текст). Элемент состоит из открывающего тега и как правило из атрибутов. Например, на рис 1 представлен элемент, вставляющий гиперссылку в HTML-документ.

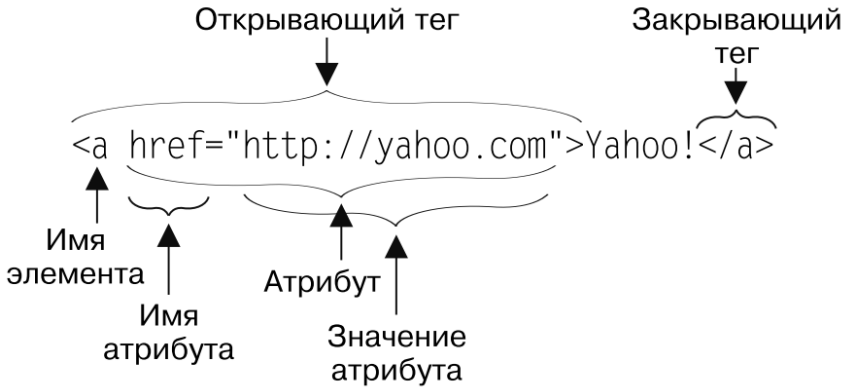



**Рис.1**

Все элементы могут иметь атрибуты. У некоторых элементов есть обязательные атрибуты. В качестве примера атрибута рассмотрим href. Он сопровождает элемент <a> (см. рис. 1). Как правило, атрибуты представляют собой пары «имя/значение».

Атрибуты предоставляют графическим движкам дополнительную информацию об элементе. Они ставятся у открывающего, а не у закрывающего тега. Чтобы завершить (закрыть) элемент используется закрывающий тег (ставится левая угловая скобка и слеш, за которым следует имя элемента — то самое, которое уже было записано в открывающем теге. Затем идет правая угловая скобка).

Между открывающим и закрывающим тегами помещается содержимое элемента (контент). Контент может включать в себя другие элементы и/или текстовые узлы. Соблюдайте правила вложения элементов! Если вы включаете один элемент в другой в качестве дочернего, то дочерний элемент должен быть как открыт, так и закрыт до окончания родительского элемента, пример вложения элементов представлен ниже:

```
<div id="content">
    <p class="files"> О направлении «Бизнес-информатика» ЮУрГУ
        <a href="http://business-inform.susu.ru/"> в Интернете </a>
    </p>
</div>
```

Обратите внимание: элемент-якорь <a> и открывается и закрывается в пределах элемента <p>. Элемент <p>, в свою очередь, полностью находится внутри элемента .

Закрывающие теги есть у всех элементов, кроме тех, которые называются самозакрывающимися (синоним — пустые элементы). Пустые элементы не могут содержать вложенных элементов или текста. Будучи самозакрывающимися, они не имеют конечного (закрывающего) тега. К самозакрывающимся (пустым) элементам относятся:

- <img/> — изображение;
- <br/> — переход на новую строку;
- <hr/> — горизонтальная разделительная линия;
- <link/> — ссылка;

 <col/> — столбец таблицы;
 <input/> — элемент формы;

## 2. Необходимые компоненты

2.1. Веб-страница должна содержать всего пять обязательных компонентов:

 объявление типа документа (DTD - Document Type Declaration);
 корневой элемент HTML — <html></html>;
 заголовок документа, прямой потомок html — <head></head>;
 заглавие документа, находящееся в заголовке, — <title></title>;
 тело документа, прямой потомок html — <body></body>.

Иными словами, вот минимальный код, который должен присутствовать в документе HTML5 (созданный файл сохраните в папку Лабораторная работа 1 с именем пример1.html):

```
<!DOCTYPE html>
<html>
        <head>
        <meta charset="utf-8"/>
        <title>О стандарте HTML5</title>
        </head>
                <body>
                С 28 октября 2014 года Консорциум Всемирной паутины
                официально рекомендует при создании использовать HTML5,
                который является языком для структурирования содержимого
                всемирной паутины
                </body>
</html>
```

A) Первая строка любого HTML-документа начинается с **DOCTYPE** (**Document Type Declaration DTD** – определение типа документа). **DOCTYPE** выполняет две функции: во-первых, он помогает валидаторам определять какие правила применять при проверке валидности кода, во-вторых, он заставляет браузеры перейти в режим соответствия стандартам (Обратите внимание, что DOCTYPE нечувствителен к регистру). До появления HTML5 выглядело следующим образом:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/ TR/html4/loose.dtd">

Однако теперь, когда у нас есть HTML5, определение типа документа выглядит так:

**<!DOCTYPE html>**

B) <HEAD> и </HEAD> - теги секции заголовка. Элемент <head> в документе содержит важную информацию. Но она, в отличие от содержимого элемента <title>, не отображается прямо в окне браузера2. Как правило, содержимое элемента <head> не выводится на экран — в большинстве браузеров пользователь видит лишь содержимое элемента <title>, обязательного в ва лидном документе HTML5.

C) Элемент <title> является обязательным и должен сопровождаться закрывающим тегом </title>. Ваша страница может пройти валидацию без тегов <head>, <body> и даже <html>, но не без <title>. Более того, если пропустить закрывающий тег этого элемента, то синтаксический разбор страницы также не удастся выполнить.

D) <body> Весь контент, который вы собираетесь отображать на странице, должен находиться в теле страницы, то есть его нужно заключить в элемент <body>.

Создайте документ, согласно структуре, приведенному в примере 1 (созданный файл сохраните в папку Лабораторная работа 1 с именем **Nast.html** ):

```
<!DOCTYPE html>
<html>
        <head>
        <title> Настольный теннис </title>
        </head>
                <body>
                Моя первая Web-страница о настольном теннисе будет написана с
                использованием стандарта разметки написания гипертекстовых страниц
                HTML 5.
    </body>
    </html>
```

Сохраните документ с именем **Nast.html** (в меню "Файл" команда "Сохранить"). Перейдите в окно браузера и откройте созданный файл.

### 3. Кодировка

Кодировка сообщает браузерам и валидаторам, какой набор символов использовать при отображении веб-страницы. Если в HTML-документе кодировка не объявлена, то браузер сначала попытается определить ее через заголовок ответа вашего HTTP-сервера (в частности, заголовок Content-iype). Кодировка, заданная в заголовке ответа, предпочтительнее, нежели указанная в документе, поэтому заголовки — основной способ предоставления такой информации. Однако существует возможность контролировать, какие заголовки будет отправлять ваш сервер, — для этого необходимо объявить кодировку в HTML-документе. Это оптимальный вариант. Если кодировка не определена ни в документе, ни в ответе заголовка, то браузер попробует выбрать ее сам, что может не соответствовать потребностям сайта.

Для задания кодировки в своем документе поместите внутри элемента head элемент meta и укажите в нем такой набор символов: <meta charset="UTF-8" /> (или windows-1251)

**Windows-1251** – набор символов и кодировка, являющаяся стандартной 8-битной кодировкой для всех русских версий Microsoft Windows. Пользуется довольно большой популярностью. Windows-1251 выгодно отличается от других 8-битных кириллических кодировок (таких как CP866, KOI8-R и ISO 8859-5) наличием практически всех символов, использующихся в русской типографике для обычного текста; она также содержит все символы для близких к русскому языку языков: украинского, белорусского, сербского и болгарского.

**UTF-8** – в настоящее время распространённая кодировка, реализующая представление Юникода, совместимое с 8-битным кодированием текста. Нашла широкое применение в операционных системах и веб-пространстве. Текст, состоящий только из символов Юникода с номерами меньше 128, при записи в UTF-8 превращается в обычный текст ASCII. Остальные символы Юникода изображаются последовательностями длиной от 2 до 6 байт.

Unicode (UTF-8) представляет собой универсальную кодировку, удовлетворяющую потребностям большинства веб разработчиков, большинство современных веб-платформ по умолчанию работают именно на ней.

*Примечание: Согласно стандарту HTML4.1 вводили <meta http-equiv="Content-Type" content="text/html; charset»UTF-8" />*

### 4. Язык веб-страницы

Для определения основного языка веб-страницы добавьте в открывающий тег элемента html атрибут lang и присвойте ему значение: <html lang="ru">

В HTML5 (как и в любой другой из предыдущих версий) необязательно объявлять основной язык документа. Однако это хорошая практика.

### 5. Метатеги для мобильной среды

Существует ряд метатегов (Метатеги (англ. meta tags) – HTML-теги, предназначенные для предоставления дополнительной информации о веб-странице), специально приспособленных для использования на мобильных устройствах. К ним, в частности, относятся метатеги, которые приказывают окну браузера занять всю доступную область просмотра.

Метатег "viewport" позволяет задавать логические размеры и масштабирование для окна браузера, соответствующего области просмотра на мобильном устройстве. В нашем приложении CubeeDoo мы воспользуемся следующим метатегом:

<meta name="viewport" content="user-scalable=no, width=device-width, initial-scale=1.0"/>

Метатег, описывающий область просмотра, поддерживается на всех смартфонах и других мобильных устройствах с операционными системами iOS, Android, webOS, Windows Phone и Opera Mobile. Устанавливая ширину области просмотра равной device-width, мы приказываем браузеру сделать ширину документа равной ширине экрана на устройстве. Метатег области видимости поддерживает список команд, разделенных запятыми. С их помощью мы можем задавать ширину, масштаб и высоту области просмотра в браузере. Можно запретить браузеру применять масштабирование либо выполнять масштабирование в диапазоне от максимального до минимального

- width=<num>|device-width. Как правило, ключевой термин device-width приходится использовать в случаях, когда ширина области просмотра должна быть равна ширине экрана устройства, но числовые значения здесь также допустимы. Значение, задаваемое по умолчанию, в разных браузерах варьируется, но, как правило, оно близко к 980. Минимальное значение — 200, максимальное — 10000.
- height=<num>|device-height. Это значение, устанавливаемое в device-height или, например, равное 480 у iPhone 4, определяет высоту области просмотра. Как правило, это значение опускается, задается только значение ширины. Для справки: минимальное значение в данном случае 223.
- user-scalable=no|yes. Определяет, может ли пользователь увеличивать или уменьшать размеры контента, то есть масштабировать область просмотра. Чтобы разрешить масштабирование, задайте здесь yes, чтобы запретить — no. По умолчанию действует значение yes. При этом значении данные не прокручиваются. Если пользовательское масштабирование допускается, его можно ограничить с помощью значений minimum-scale и maximum-scale, присваиваемых метатегу области просмотра.
- initial-scale=<float>. Задает исходный коэффициент (множитель) масштабирования, используемый при просмотре веб-страницы. По умолчанию задается значение 1.0, при котором веб-страница отображается без масштабирования.
- maximum-scale=<float>. Задает максимальный лимит пользовательского масштабирования, если параметр user-scalable не установлен в no. Максимальное значение — 10, но это может быть любое дробное значение от 0.25 и выше minimum-scale=<float>. Задает минимальный лимит пользовательского масштабирования веб-страницы. Минимальное значение — 0.25.

В большинстве браузеров ширина области просмотра по умолчанию составляет 980 пикселов. Устанавливая здесь значение device-width, пользователю не приходится увеличивать загруженную страницу, поскольку мы выдали ему картинку шириной 980 пикселов, тогда как ширина экрана устройства — всего 320 пикселов.

Мы могли бы задать для области просмотра значение 320, а не device-width, но в таком случае страница будет корректно отображаться лишь на устройствах именно с такой шириной экрана. Оптимальный вариант — приравнивать ширину окна к ширине экрана

устройства, так как страница будет масштабироваться пропорционально устройству. При этом для автора сайта не имеет значения, с каким именно устройством работает пользователь.

Однако такая тактика оптимальна при работе не с любыми сайтами, а только с мобильными сайтами и веб-приложениями. Кроме того, когда активирована команда user-scalable, пользователь может увеличить страницу, чтобы ее легче было читать и людям с ослабленным зрением.

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

## 6. Структурные элементы

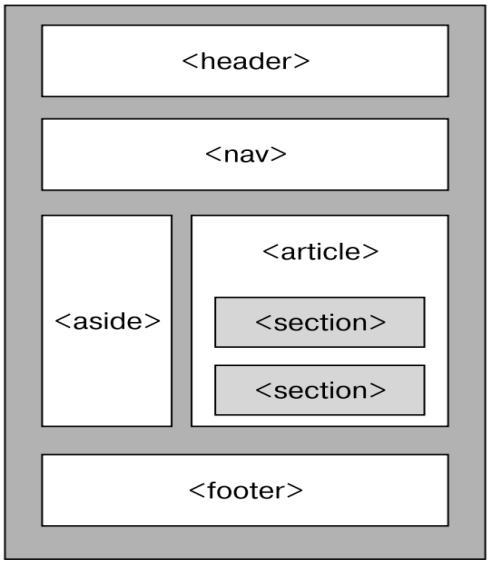
Начальным элементом, открывающим «тело» HTML-документа, предназначенного для вывода на экран браузера пользователю, является тег <body>. Другие структурные элементы появились еще в HTML 4 и включены в спецификацию HTML5. Таковы, например, теги создания разноуровневых заголовков от <h1> до <h6>. Заголовок обозначает начало раздела документа. В стандарте определено 6 уровней заголовков: от H1 до H6. Текст, окруженный тегами <H1></H1>, получается большим — это основной заголовок. Если текст окружен тегами <H2></H2>, то он выглядит несколько меньше (подзаголовок); текст внутри <H3></H3> еще меньше и так далее до <H6></H6>.

Внесите изменения в файл Nast.html:

```
<!DOCTYPE html>
<html>
    <head>
    <title> Настольный теннис </title>
    <meta name="viewport" content="user-scalable=yes, width=device-width, initial-
        scale=1.0"/>
    </head>
        <body>
            <H1> Приветствую Вас на web-страничке о настольном теннисе</H1>
            <H3> Виды соревнований </H3>
            <H3> Участники соревнований </H3>
            <H4> Правила игры </H4>
            Моя первая Web-страница о настольном теннисе будет написана с
        использованием стандарта разметки написания гипертекстовых страниц HTML 5
        </body>
</html>
```

В HTML5 добавлено несколько новых секционирующих элементов, в частности:

- section;
- article;
- nav;
- aside;
- header;
- footer.




Типичный макет веб-страницы, в котором используются структурные элементы HTML5 представлен на рисунке:

**– header** используется для хранения заголовков страницы (<h1> … <h6>) и/или разделов section (section — основной из новых структурных элементов, контент которых может быть сгруппирован тематически или связан между собой). Может содержать дополнительную информацию, например, логотипы и навигацию, т.е. определяет «шапку» сайта или раздела.

– **nav** включает основные навигационные ссылки на страницы и зачастую содержится в header.

– **aside**, который хранит информацию, связанную с окружающим контентом, но существующую независимо, например, на боковой панели (содержаться выносные цитаты, мысли, дополнительные ссылки).

– **article**, который определяет автономный контент, который может использоваться независимо от страницы в целом, подобно записи в блоге (задаёт содержание сайта вроде новости, статьи, записи блога, форума или др.). Он идеально подходит для написания основного контента сайта.

– **section** основной из новых структурных элементов, контент которых может быть сгруппирован тематически или связан между собой. Это различная информация, не относящаяся к области навигации, боковым панелям, заголовкам или нижним колонтитулам и относится к основному контенту страницы.

Секционирующий элемент определяет область применения заголовков и нижних колонтитулов страницы, например, относится этот заголовок лишь к части страницы или ко всему документу.

– **footer** содержит информацию о странице, например, информацию об авторе, ссылки на использованные источники, а также сведения об авторских правах. Т.е. задаёт «подвал» сайта или раздела, в нем обычно располагается имя автора, дата документа, контактная и правовая информация.

Внесите изменения в файл Nast.html, описав структуру HTMLстраницы:

```
<!DOCTYPE html>
<html>
<head>
<title> Настольный теннис </title>
<meta name="viewport" content="user-scalable=yes,
width=device-width, initial-scale=1.0"/>
</head>
  <body>
          <article>
        <header>
            <h1>Приветствую Вас на web-страничке о настольном
       теннисе</h1>
        </header>
     <p>В настольный теннис играют миллионы людей во всем
  мире, и не случайно. Этот популярный вид спорта, который по-
  другому называется пинг-понг</p>
     <footer>Источник: http://www.takzdorovo.ru</footer>
     </ article >
<H3> Виды соревнований </H3>
<H3> Участники соревнований </H3>
<H4> Правила игры </H4>
Моя первая Web-страница о настольном теннисе будет написана с
использованием стандарта разметки написания гипертекстовых
страниц HTML 5
</body>
</html>
```

Вышеперечисленные теги ориентированы на робота-поисковика. Поисковые системы начинают лучше индексировать сайт, потому что чётко отделяют контент страницы от вспомогательных элементов. Речевые браузеры, предназначенные для слепых людей, пропускают заголовок и переходят непосредственно к содержимому. Сайты могут

автоматически обмениваться контентом и другой информацией между собой. Все эти возможности называются семантикой и позволяют представить данные в удобном для роботов виде.

Когда следует применять эти новые элементы? Опять же, необходимо сосредоточиться на контенте и подумать о семантике каждого элемента. Например, многие веб-страницы включают область, которая считается «заголовком» и, возможно, хранит логотип, название компании и слоган. Это, безусловно, хороший повод использовать **header**. Элементы **section** или **aside** могут определяться в документе со своими заголовками и навигацией, которые также могут быть помещены в **header**.

Все это справедливо и для **footer**. На данный момент контент большинства страниц подходит для нового элемента **footer** — в него можно добавить сведения об авторе, авторском праве, а также сопутствующую информацию. Элементы **sections, articles и asides** могут содержать ту же информацию и, в свою очередь, включать в себя **footer**.

В сообществе авторов спецификаций нередко поднимается вопрос о том, что два этих элемента слишком похожи. Но следует применять именно <article>, а не <section>, если заключенный в нем контент отличается значительной самостоятельностью. Зачастую такое решение бывает субъективным. Рассуждая о сходствах, различиях и функциональности двух этих элементов, проведем аналогию с воскресной газетой. В воскресной газете есть несколько разделов: передовица, новости, информация о недвижимости, раздел объявлений, еженедельное приложение, кроссворд и т. д. В большинстве разделов есть статьи. На сайте, как и в воскресной газете, статьи и разделы могут быть вложены внутрь других разделов. Например, в элементе <article> могут находиться запись для форума, газетная или журнальная статья, пост для блога, пользовательский комментарий, интерактивный виджет или гаджет, а также любой другой независимый элемент содержимого.

Внесите изменения в файл Nast.html, описав структуру HTMLстраницы:

```
<!DOCTYPE html>
<html>
<head>
<title> Настольный теннис </title>
<meta name="viewport" content="user-scalable=yes, width=device-width, initial-scale=1.0"/>
</head>
    <body>
    <nav role="navigation">
      <ul>
      <li><a href="/">На главную</a></li>
      <li><a href="/Prava1/"> Права и обязанности участника</a></li>
      <li><a href="/ Prava2/"> Права и обязанности представителя (тренера) и капитана команды</a></li>
      </ul>
    </nav>
        <article>
          <header>
              <h1>Приветствую Вас на web-страничке о настольном теннисе</h1>
          </header>
```

В настольный теннис играют миллионы людей во всем мире, и не случайно. Этот популярный вид спорта, который по-другому называется пинг-понг

<section>
<h2>Все о видах соревнований по настольному теннису</h2>
В системе спортивной подготовки выделяют: 1) контрольные соревнования; 2) подготовительные; 3) отборочные; 4) подводящие; 5) главные соревнования.
1. Контрольные соревнования - проводятся с целью контроля за уровнем подготовленности спортсмена.
2. Подготовительные соревнования – основной целью их является адаптация спортсменов к условиям соревновательной борьбы и т.д.
<footer>Источник: http://opace.ru/a/vidy_sorevnovaniy>
</section>

<section>
<h2>Все об участниках соревнований </h2>
 К участию в соревнованиях допускаются спортсмены, включенные в именную заявку, спортивная квалификация которых соответствует уровню соревнований, и получившие разрешение врача.
 Спортсмены младших возрастных категорий допускаются к соревнованиям старших возрастных категорий с разрешения тренерского совета соответствующего уровня 
<footer>Источник: http://ttfr.ru/rus/1/ >
</section>

Моя первая Web-страница о настольном теннисе будет написана с использованием стандарта разметки написания гипертекстовых страниц HTML 5

<footer>Источник: http://www.takzdorovo.ru</footer>
</ article >
</body>
</html>

Ниже представлен пример более сложной схемы страницы с использованием вышеперечисленных тегов (структурных элементов) (рис. 3.).

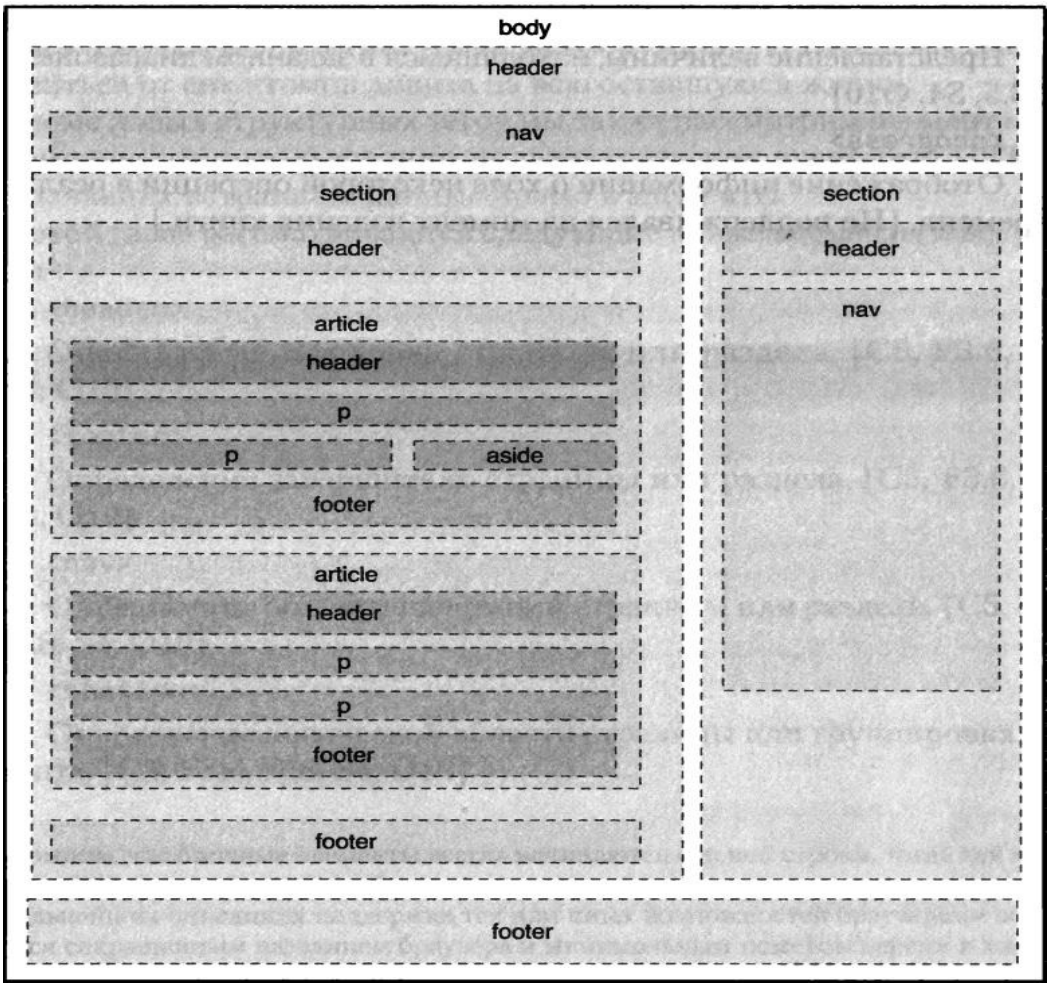

body
header
nav
section
header
article
header
p
p
aside
footer
article
header
p
p
footer
footer
section
header
nav
footer

Рис 3