МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
"САМАРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ"

К а ф е д р а "Информатика и системы управления"

М.И. ПОДНЕБЕСОВА

WEB-ПРОГРАММИРОВАНИЕ

Учебное пособие

Самара
Самарский государственный технический университет
2017

Печатается по решению редакционно-издательского совета СамГТУ

УДК 004.55

Поднебесова М.И.

Web-программирование: учеб.пособие /*М.И. Поднебесова*. – Самара: Самар. гос. техн. ун-т, 2017. - 94 с.: ил.

В издании приведен теоретический материал, охватывающий основное содержание разделов дисциплины "Web-программирование", а также практические задания по приобретению навыков создания web-страниц с использованием языка HTML, каскадных таблиц стилей (CSS) и языка JavaScrip и материал для самоконтроля.

Пособие предназначено для студентов направления бакалавриата 44.03.01 "Профессиональное обучение", профиль "Информатика, вычислительная техника и компьютерные технологии" и может быть использовано при изучении дисциплины "Web-программирование".

УДК 004.55

Р е ц е н з е н т ы : доцент кафедры ЭПА филиала ФГБОУ ВО "Сам ГТУ" в г. Сызрани, к.т.н. Тамьяров А.В.;

Зам. начальника кафедры АРЭО филиала ВУНЦ ВВС "ВВА" в г. Сызрань, к.т.н. Алексеев Э.О.

ПРЕДИСЛОВИЕ

Данное пособие представляет собой учебно-теоретическое издание, освещающее содержание основных разделов дисциплины "Web-программирование" согласно федеральным государственным образовательным стандартам по подготовке бакалавров по направлению "Профессиональное обучение", профиль "Информатика, вычислительная техника и компьютерные технологии".

Целью учебного пособия является формирование у студента представления об основах современных технологий создания web-страниц и приобретение теоретических и практических навыков создания грамотно оформленных web-сайтов. Учебное пособие направлено на формирование профессиональной компетенции, связанной со способностью проектировать и оснащать образовательно-пространственную среду для теоретического и практического обучения.

Методика изложения материала учитывает последовательность изучения разделов дисциплины в соответствии с рабочей программой и предусматривает приобретение практических навыков создания web-документов с помощью элементарных заданий по ходу изложения материала, выполнение которых способствует наиболее быстрому и качественному усвоению материала. Для каждого раздела предлагаются контрольные вопросы, позволяющие оценить степень усвоения пройденного материала.

Предлагаемое издание предоставляется студентам всех форм обучения по направлению "Профессиональное обучение", профиль "Информатика, вычислительная техника и компьютерные технологии" как помощь при изучении теоретического материала по дисциплине "Web-программирование".

ВВЕДЕНИЕ

Педагогу в области информационных технологий помимо знаний в области психолого-педагогических дисциплин, необходимо обладать высоким уровнем знаний в области вычислительной техники, а также активно применять информационные технологии в учебном процессе и при подготовке методических материалов. Эффективным средством проектирования методических материалов могут служить web-технологии (например, для разработки электронных учебников, обучающих сайтов, контролирующих материалов и т.п.). Применение современных web-технологий позволит создать интерактивную среду обучения, рационально организовать познавательную деятельность обучаемых, повысить эффективность процесса обучения.

К web-программированию относят отдельное направление программирования, которое применяется для разработки web-приложений. Высокие темпы развития сетевых и Internet технологий обуславливают многообразие существующих и появление новых направлений в области web-программирования. Как наука web-программирование не является сформированной и в большей степени является набором существующих программных технологий, используемых для разработки web-страниц и web-сайтов.

При создании современных сайтов подразумевается использование трех основных технологий: языка разметки гипертекста HTML5, каскадных таблиц стилей CSS3 и языка сценариев JavaScript.

Основная задача данного пособия заключается в формировании в рамках предложенного теоретического материала знаний, умений и навыков, характеризующих определенный уровень профессиональных компетенций, связанных со способностями проектировать и оснащать образовательно-пространственную среду для теоретического и практического обучения с применением web-технологий.

Первый раздел учебного пособия освещает вопросы, связанные с разработкой web-страниц с помощью языка разметки гипертекста HTML (Hyper Text Markup Language). Приведено описание структуры HTML-документа и основных элементов языка HTML. Рассмотрены

примеры использования элементов с приведение кода документа и визуального отображения документа в браузере.

Во втором разделе показаны возможности применения каскадных таблиц стилей CSS (Cascading Style Sheets) для визуального представления web-страниц. Приведено описание синтаксиса стилевых правил и различных селекторов CSS с примерами их использования.

Третий раздел пособия посвящен основам языка сценариев JavaScript и способам его применения для добавления интерактивности на web-страницы. Приведено краткое описание синтаксиса языка, рассматриваются понятия объектной модели браузера и объектной модели документа, а также основные объекты, их свойства и методы.

В пособии параллельно с изложением материала студентам предоставляются элементарные практические задания, позволяющие приобрести практические навыки создания web-документов с использованием описанных в каждом разделе web-технологий. Для закрепления теоретических знаний в учебном пособии приведены контрольные вопросы по каждому разделу.

1. Основы HTML

1.1. Язык разметки гипертекста HTML

Язык разметки текста в информационных технологиях представляется набором символов или последовательностей, которые вставляются в текст с целью передачи информации о его выводе или строении. Текстовый документ, созданный с использованием языка разметки, содержит в себе помимо текста дополнительные данные о различных его элементах - например, указание на заголовки, выделения, списки и т. д. Наряду с этим с помощью языка появляется возможность добавления в документ интерактивных элементов и содержания других документов.

Различают *логическую* и *визуальную* разметки. ***Логическая разметка*** указывает, какую роль играет данный участок документа в его общей структуре (например, "данная строка является заголовком"). ***Визуальная разметка*** определяет, как именно будет отображаться этот элемент (например, "данную строку следует отображать жирным шрифтом"). Идея языков разметки состоит в том, что визуальное отображение документа должно автоматически получаться из логической разметки и не зависеть от его непосредственного содержания. Это упрощает автоматическую обработку документа и его отображение в различных условиях [1].

Язык **HTML (Hyper Text Markup Language)** - это независимый от платформ язык разметки гипертекста. *Гипертекст* - это текст, в котором имеются ссылки для автоматического перехода на другие тексты – *гиперссылки*. HTML является стандартным языком, предназначенным для создания гипертекстовых документов. HTML-документы доступны для просмотра в различных web-браузерах и в различных операционных системах.

Хотя HTML-документы и представляют собой файлы с расширением .htm или .html, на самом деле это обычные текстовые файлы формата ASCII. Поэтому любая программа, позволяющая создавать текстовые документы формата ASCII, может служить редактором

HTML. Так же HTML-документы можно создавать и при помощи специализированных HTML-редакторов.

На протяжении 13 лет (с 2004 г.) официальным стандартом языка была версия 4.01 и, в октябре (28) 2014 г. был опубликован стандарт пятой версии языка HTML (1 ноября 2016 г. – версия 5.1). Несмотря на схожесть названий, HTML5 не является продолжением HTML4.

HTML5 - это не продолжатель языка разметки гипертекста, а новая открытая платформа, предназначенная для создания web-приложений использующих аудио, видео, графику, анимацию и многое другое, которая в маркетинговых целях названа знакомой аббревиатурой и построена на базе HTML.

Чтобы не нарушать основы языка HTML, предназначенного для структурной и логической разметки гипертекста, была создана отдельная система для возможности описания визуального оформления HTML-документов - CSS (Cascading Style Sheets), каскадные таблицы (листы) стилей. С помощью таблиц стилей CSS можно задавать параметры визуального представления для любого элемента HTML [1].

1.2. Основные понятия языка HTML

Исходный код HTML-документа представляет собой текст, внутри которого располагаются ***элементы*** разметки. При просмотре web-страницы в браузере элементы разметки не видны, но наблюдается результат их воздействия на текст.

Элементы разметки состоят из заключённых в угловые скобки (**<…>**) **дескрипторов - тегов(tags)** и их **атрибутов**. **Тег** – оформленная единица HTML-кода. Теги бывают *начальными* (открывающимися) и *конечными* (закрывающимися). Существуют теги, не требующие закрывающего дескриптора. Например, тег разрыва строки
. Совокупность открывающего (**< >**) и закрывающего (**< / >**) тегов называется ***контейнер***. Пара начального и конечного тегов и участок документа между ними, на который распространяется их влияние, является **элементом** разметки.

Каждый HTML-документ должен начинаться с элемента **<!DOCTYPE>,** который предназначен для указания типа текущего документа - **DTD** (document type definition, описание типа документа), что в свою очередь сообщает серверу способ обработки документа и то, какие дескрипторы могут находиться на web-странице. В HTML5 существует один тип документа, который переводит браузер в стандартный режим: `<!DOCTYPE html>`.

Общая схема исходного кода HTML-документа выглядит следующим образом:

```
<!DOCTYPE>
<html>
    <head>
  служебные элементы
    </head>
        <body>
    текст и прочие элементы
        </body>
</html>
```

Элемент **<html>** сообщает web-браузеру, что данный документ написан с использованием HTML, и определяет границы документа. Внутри элемента **<HTML>** располагается собственно весь документ. Документ, обозначенный тегами **<html>** и **</html>** дополнительно делится на *голову (заголовок)* - **<head>** и *тело (основную часть)* - **<body>**.

!! *Запустите программу* **Блокнот** *и введите в окно документа следующий текст:*

<!DOCTYPE html>
<html>
<head>
<title>Моя первая страница</title>
</head>

<body>

<p align="center">Добро пожаловать на мою первую страничку!

<p align="center">Меня зовут (Фамилия, Имя). А это моя страничка, созданная на занятиях по web-программированию!

</body>

</html>

Сохраните файл в предварительно созданной папке под своей фамилией, присвоив ему имя ***First.html****.*

Примечание. В дальнейшем указания на выполнение заданий будут помечаться знаком **!!** и выделяться *курсивом*.

!! *Запустите браузер и откройте в нем созданный вами файл* ***First.html****. Просмотрите результат.*

1.3. Элементы <head> и <body>

Область, обозначаемая тегом **<head>**, определяет общую структуру документа и его глобальные свойства. Располагающиеся в данном разделе документа данные относятся к служебной информации и требуются браузеру пользователя для идентификации и адекватного отображения документа. В элементе <head> допускается вложение следующих элементов: **<title>, <base>, <link>, <meta>, <style>**.

Элемент <title>

Создает название web-страницы, отображаемое в заголовке окна браузера. При отсутствии названия документа, браузер может сгенерировать его из названия файла либо отобразить сообщение "Документ без имени" или "Untitled Document". Название рекомендуется создавать минимальной длины и в тоже время оно должно быть информативным.

Элемент <link>

С помощью данного элемента описываются взаимосвязи документа с остальными документами сайта, определяющие его место в иерархической структуре сайта. Элемент не имеет конечного тега.

Обычно элемент **<link>** используется для указания ссылки на внешние таблицы стилей.

Например:

```
<LINK REL=stylesheet HREF="style.css" TYPE="text/css">,
```

где **style.css** файл, содержащий определения таблицы стилей.

Элемент <meta>

В данном элементе располагаются дополнительные сведения о способе обработки документа, а также данные, позволяющие поисковым машинам идентифицировать и классифицировать документ без его загрузки. Элемент не имеет конечного тега.

Атрибуты:

name – определяет тип данных, используется браузером для получения дополнительной информации о документе. Данный атрибут часто заменяют атрибутом *http-equiv*, который используется сервером для создания дополнительных полей при выполнении запроса. В этом случае элемент <meta> с атрибутом будет включен в заголовок ответа браузера.

http-equiv – конвертирует метатег в заголовок протокола HTTP. Использование атрибута *http-equiv* со значением "refresh" позволяет организовать принудительное обновление страницы с некоторым промежутком времени или загрузку нескольких страниц с определенным интервалом. Значение атрибута *content* в этом случае означает промежуток между обновлениями в секундах, так если оно равно нулевому значению - страница будет обновляться непрерывно.

content – определяет содержание данных, указанных в атрибуте *name* или *http-equiv*.

charset – определяет кодировку документа.

Например:

Описание типа и характеристик документа.

```
<meta http-equiv="content-type" content="text/html"
charset="utf-8">
```

описание языка документа

```
<meta http-equiv="content-language" content="en">
```

имя автора документа

```
<meta name="author" content="Поднебесова М.И.">
```

набор ключевых слов для поиска

```
<meta name="keywords" content="HTML, web, язык разметки, тег, гипертекст">
```

!! *Добавьте в заголовок (<head>) документа First.html описание типа и характеристик документа, а так же имя автора.*

Элемент <style>

Данный элемент применяется для внедрения в документ таблицы стилей (CSS – Cascade Style Sheet).

Атрибуты:

type – обязательный атрибут, определяющий MIME-тип внедряемой таблицы стилей. Обычно, атрибут принимает значение "text/css".

Синтаксис: `<style type="text/css">`, `<style type = "text/javascript">`

Элемент **<body>** необходим для определения основной части документа, которая будет отображаться в браузере пользователя. Обязательно имеет начальный и конечный теги.

1.4. Элементы тела документа

Элементы тела документа (основной части) определяют отображаемое в окне браузера содержимое HTML-документа. В теле документа могут находиться ссылки на другие документы, текст, изображения и другие данные.

Все элементы, встречающиеся в теле HTML-документа можно разделить на две группы: блочные и строчные.

Блочные элементы

Блочным называется элемент, который отображается на web-странице в виде прямоугольника (параграфы, абзацы, списки, таблицы и т.п.). Такой элемент занимает всю доступную ширину окна, вы-

сота элемента определяется его содержимым, и он всегда начинается с новой строки. Блоки располагаются по вертикали друг под другом. Запрещено вставлять блочный элемент внутрь строчного. К блочным элементам относятся следующие.

Абзац <p> (paragraph)

В HTML-документе обычно игнорируются символы возврата каретки (нажатие клавиши Enter). Физический разрыв абзаца может находиться в любом месте исходного текста документа (для удобства его читаемости). Однако браузер отображает абзацы только при наличии тега <p>.

Атрибуты:

align – определяет горизонтальное выравнивание текста в абзаце. Может принимать значения: *left*, *center, right* (по левому краю, по центру и по правому краю соответственно). По умолчанию имеет значение *left*. Несмотря на выбранный способ выравнивания в текущем абзаце, в каждом последующем абзаце используется значение по умолчанию.

Листинг 1. Применение элемента <p>.

```
<!DOCTYPE html>
<html>
<head>
<title>Пример абзаца</title>
</head>
<body>
<p align="center">В отличие от большинства текстовых процессоров, в HTML-документе обычно игнорируются символы возврата каретки (нажатие клавиши Enter). Физический разрыв абзаца может находиться в любом месте исходного текста документа (для удобства его читаемости). Однако браузер разделяет абзацы только при наличии тега &lt;P&gt;.
<p>Атрибуты:
<p align="right">ALIGN - определяет способ горизонтального выравнивания параграфа. Возможные значе-
```

```
ния: left, center, right (по левому краю, по центру и
по правому краю соответственно). По умолчанию имеет
значение left.
   <p align="left">Каждый следующий абзац игнорирует
заданное для предыдущего абзаца значение атрибута
ALIGN.
   </body>
   </html>
```

Результат листинга на Рис. 1.

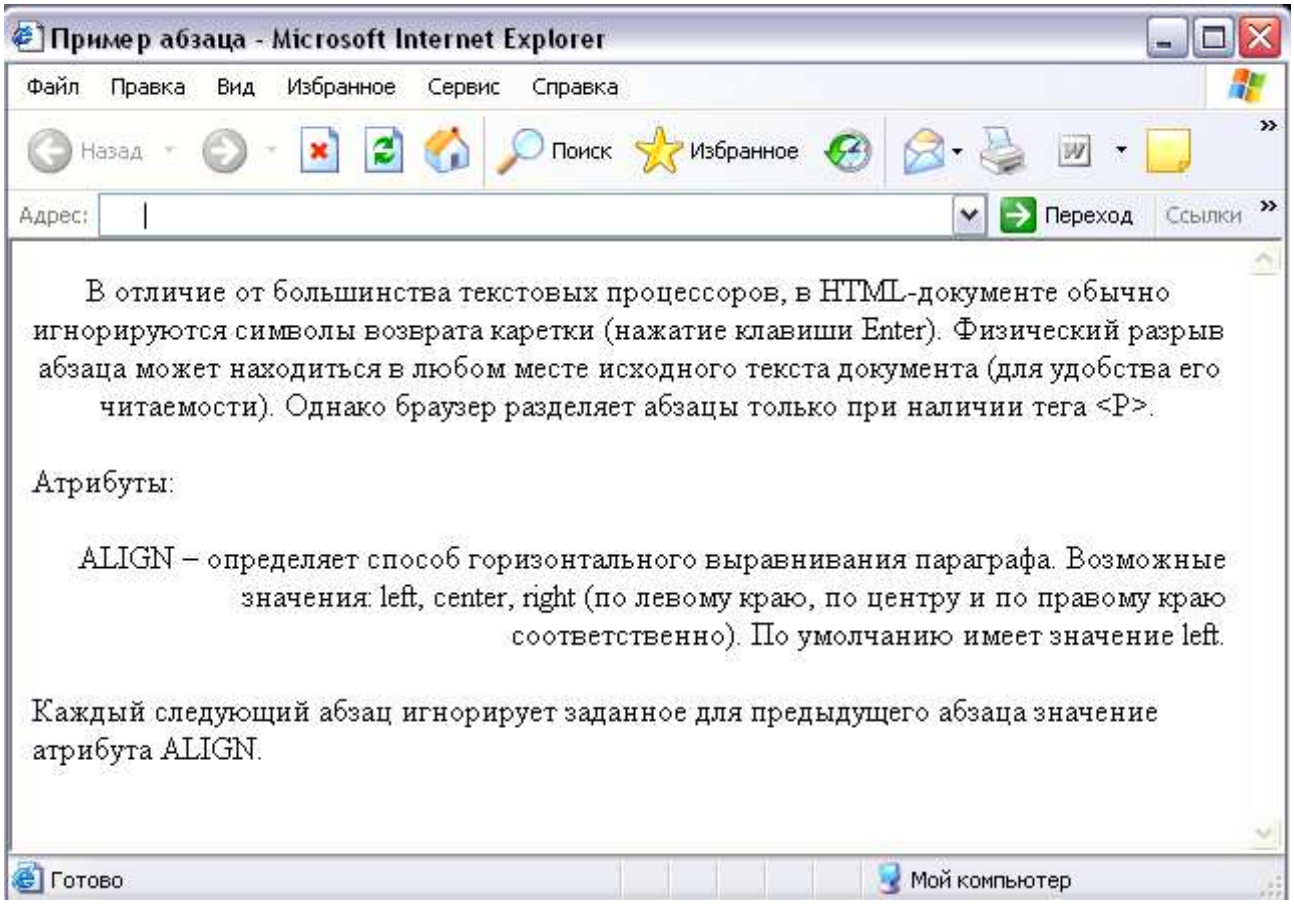

Рис. 1. Применение элемента <p>

!! *Создайте в* ***Блокноте*** *новый документ, введите в него содержание Листинга 2 и сохраните файл в своей папке под именем* ***Primer1.html****. Измените выравнивание абзацев по своему усмотрению. Просмотрите результат в браузере.*

Уровни заголовков <hn> (headers)

Данный элемент применяется для создания в документе заголовков, выделяемых полужирным начертание текста и отделяемых от остального текста отступами перед и после абзаца. Существует шесть уровней заголовков, которые различаются размером шрифта. Первый уровень заголовков (самый крупный шрифт) обозначается цифрой 1, следующий - 2, и т.д.

Атрибуты:

align – определяет горизонтальное выравнивание. Может принимать значения: *left*, *right, center* (слева, справа и по центру соответственно).

Листинг 2. Уровни заголовков.

```
<!DOCTYPE html>
<html>
<head>
   <title>Уровни заголовков</title>
</head>
<body>
<h1 align="center">Пример заголовка 1</h1>
<h2 align="center">Пример заголовка 2</h2>
<h3 align="left">Пример заголовка 3</h3>
<h4 align="left">Пример заголовка 4</h4>
<h5 align="right">Пример заголовка 5</h5>
<h6 align="right">Пример заголовка 6</h6>
</body>
</html>
```

Результат листинга на Рис. 2:

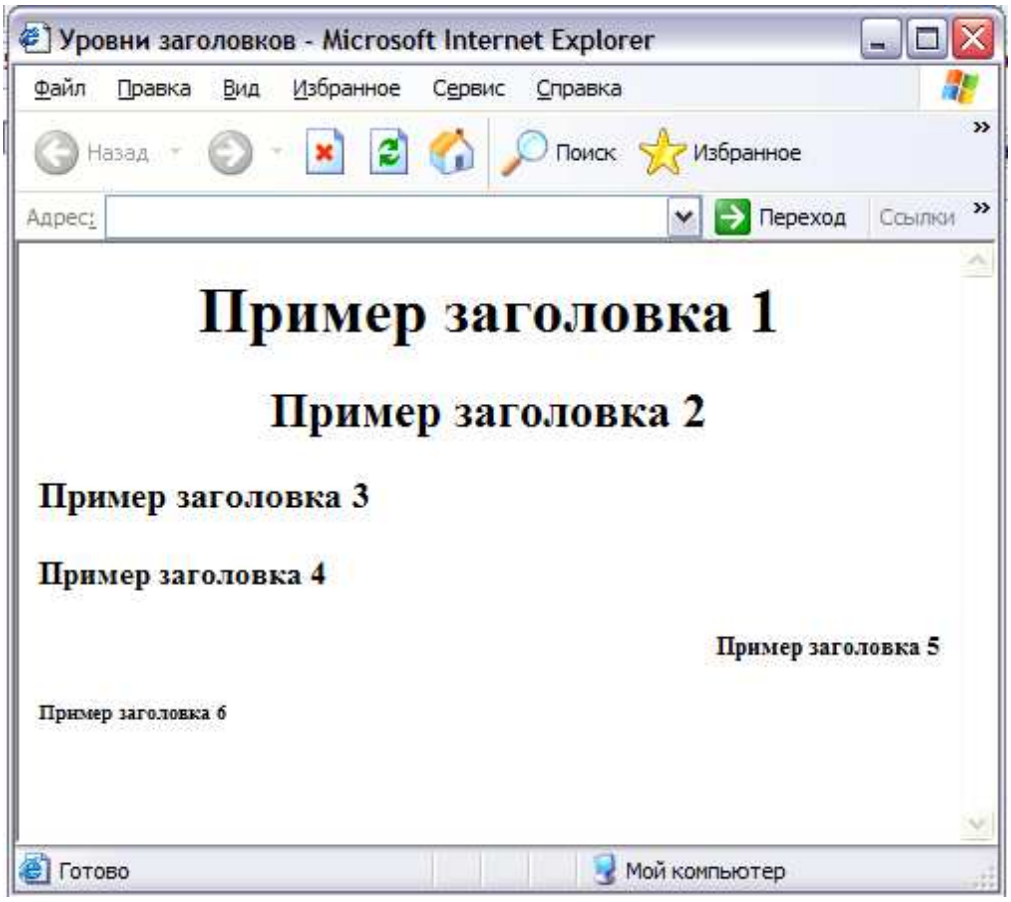

Рис. 2. Примеры заголовков.

!! *Добавьте в текст документа* ***Primer1.html*** *заголовок "Элемент Pharagraph" и выделите его, выбрав один из шести уровней, и задайте способ выравнивания по центру. Просмотрите результат в браузере.*

Горизонтальная линия <hr> (Horisontal Rule)

Позволяет внедрить в текст горизонтальную разделительную линию.

Атрибуты:

width – определяет длину линии в пикселях или процентах от ширины окна браузера.

size – определяет толщину линии в пикселях.

align – определяет горизонтальное выравнивание линии. Возможные значения: *left, right, center* (используется по умолчанию).

noshade – позволяет отобразить линию без трехмерных эффектов. Атрибут не требует указания значения. Без данного атрибута линия отображается объемной.

Листинг 3. Пример горизонтальной линии.

```
<html>
<head>
<title>Горизонтальная линия</title>
</head>
<body>
<hr width="100%">
<hr noshade width="250" align="right">
<hr size="7" width="50%">
<hr noshade size="15" width="100" align="left">
</body>
</html>
```

Результат листинга на Рис. 3.

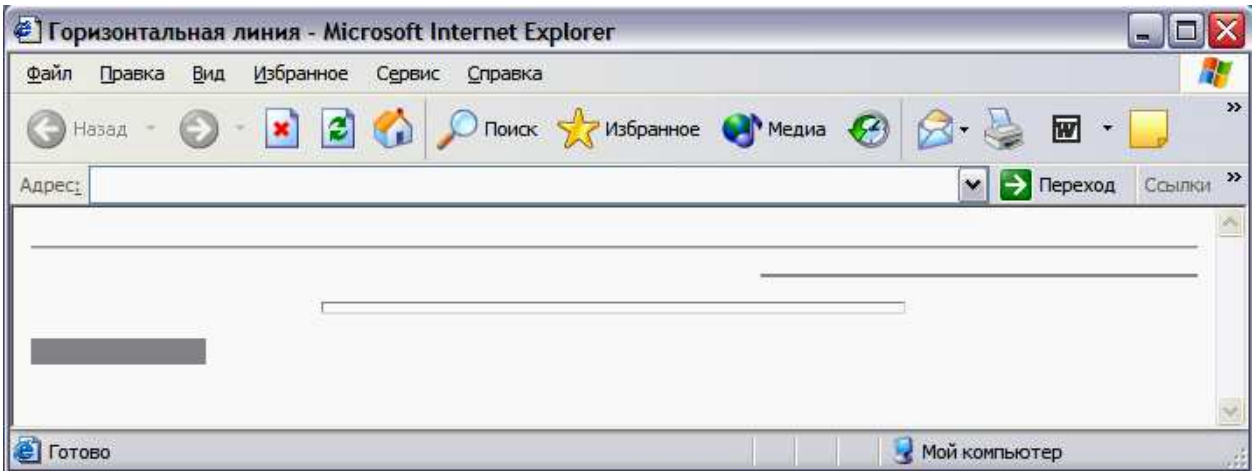

Рис. 3. Примеры горизонтальных линий.

!! *Отделите заголовок от остального текста страницы горизонтальной линией. Размер, толщину и выравнивание линии выберите по своему усмотрению. Просмотрите результат.*

Элемент (division)

Данный элемент применяется для логического выделения фрагмента HTML-документа. Используется как удобный контейнер для объектов страницы, которым легко динамически манипулировать – перемещать, включать/выключать, создавать слои, регулировать от-

ступы и т.п. Содержимое элемента (текст или другие элементы) по умолчанию оформляются как отдельный абзац. Используется как основной элемент блочной верстки web-страниц. Внешний вид блока определяется стилями.

Атрибуты:

align – определяет горизонтальное выравнивание содержимого элемента. Возможные значения: *left*, *right*, *center* .

Синтаксис: `<div align="left">Фрагмент текста</div >`

Элемент <pre> (preformatted text)

Данный элемент применяется для внедрения в документ блока предварительно форматированного текста. Браузер отображает такой текст с помощью моноширинного шрифта со всеми пробелами и символами конца строки.

Листинг 4. Пример отформатированного текста.

```
<html>
<head>
    <title>Preformatted text</title>
</head>
<body>
Используется для включения в документ уже отформа-
тированного текста.
<pre> Браузеры воспроизводят содержимое
     этого элемента с помощью моноширинного шрифта,
 сохраняя пробелы и
     символы конца строки.</pre>

</body>
</html>
```

Результат листинга на Рис. 4:

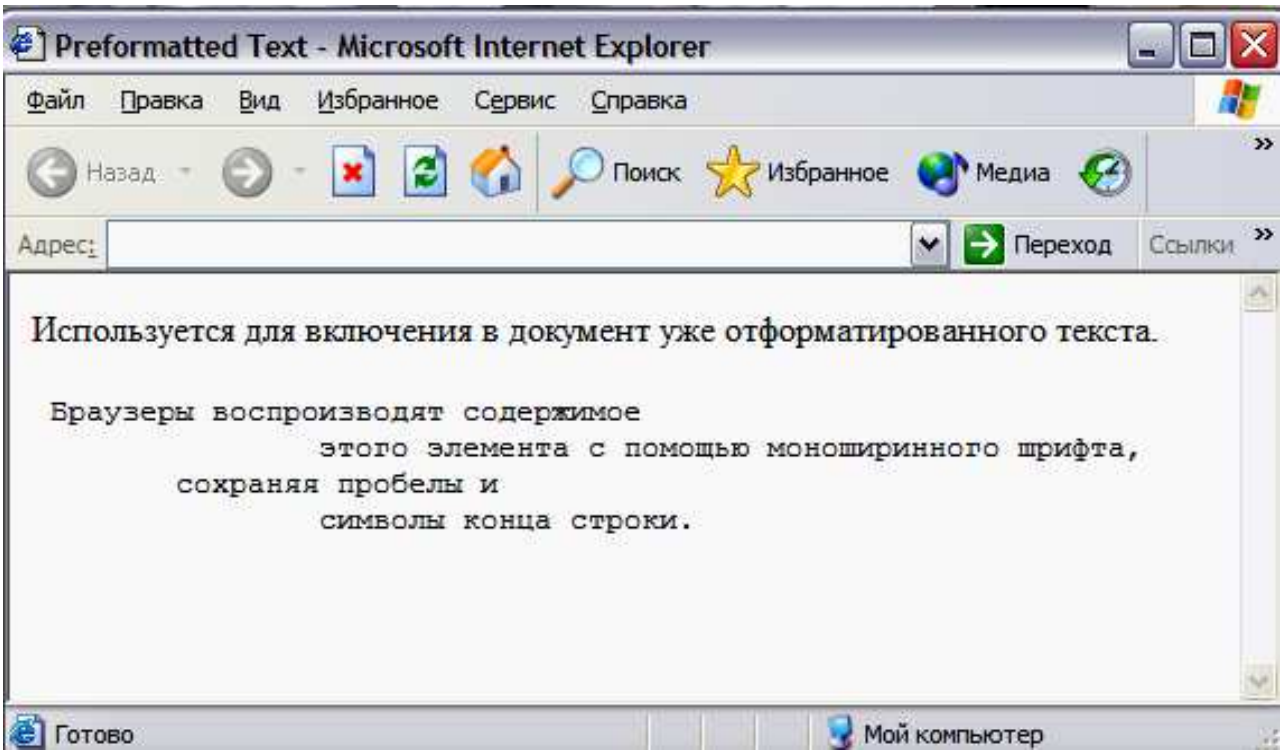

Рис. 4. Пример отформатированного текста.

!! *Добавьте в конец документа отформатированный текст из Листинга 4. Просмотрите результат.*

Также к блочным элементам относятся элементы для создания различных видов списков, таблиц и форм. Данные элементы будут описаны позднее.

Строчные элементы

Строчными называются элементы документа, которые являются непосредственной частью строки. Используются для изменения вида и стиля текста, в частности, отдельных символов и слов. Некоторые строчные элементы описаны ниже.

**Элемент **

Позволяет выделить некоторое количество текста для последующего его форматирования, но в отличие от элемента не начинает новый абзац. При помощи элемента можно выделить часть текста внутри другого элемента и применить к нему определенный стиль.

Синтаксис: `<span>Текст для форматирования</span>`

**Разрыв строки
 (break)**

Этот элемент задает разрыв текста с переходом на новую строку (перенос). Элемент не имеет конечного тега.

!! *Разорвите текст первого абзаца на три части. Просмотрите результат.*

Элементы <b> и <i>

Данные элементы позволяют отобразить размещенный между тегами текст жирным и курсивным начертанием соответственно. Использование этих тегов современной спецификацией HTML не осуждается, однако инструменты CSS дают больше возможностей для модификации начертания текста. В тоже время поисковые системы повышают рейтинг слов выделенных жирным начертанием с помощью тега <b>.

!! *Оформите в документе жирным шрифтом слово "Атрибуты:", а курсивом название атрибута "ALIGN".*

Комментарии

Для добавления комментария в текст документа используется тег <!-- -->. Он позволяет скрыть от пользователя комментарии к исходному коду, а так же сценарии JavaScript от браузеров, которые их не поддерживают.

Синтаксис: `<!-- Текст комментария -->`.

!! *Добавьте в текст документа перед применением тега <pre> комментарий – краткое описание назначения элемента <pre>.*

Управляющие последовательности

Некоторые символы являются управляющими символами в HTML и не могут напрямую использоваться в документе: левая угловая скобка "<", правая угловая скобка ">", амперсанд "&", двойные кавычки. Для использования этих символов в документе, требуется заменить их escape-последовательностями (Таблица 1). для обозначения специальных символов используется большое количество escape-последовательностей.

Таблица 1

Управляющие последовательности для отображения специальных символов

Символ	Escape-последовательность
<	<
>	>
&	&
"	"
©	©
®	®

Примечание. Escape-последовательности чувствительны к регистру: нельзя использовать < вместо <.

1.5. Организация списков

В HTML-документе возможна организация списков трех типов:

- упорядоченные (нумерованные);
- неупорядоченные (маркированные);
- список определения (описаний).

Отличаются они способом оформления. Существует возможность создания вложенных списков, с использованием используя различных тегов списков или повторяя одни списки внутри других.

Упорядоченные (нумерованные) списки

В нумерованном списке браузером автоматически нумеруются элементы списка. При удалении одного или нескольких элементов упорядоченного списка, номера остальных элементов автоматически обновляются. Упорядоченный список организуется элементом **<ol>** (Ordered List). Каждый пункт нумерованного списка начинается с тега **<li>.**

Атрибуты:

start – указывает первое число нумерации элементов (только целые числа);

type – определяет стиль нумерации элементов списка. Может принимать значения: "A" – заглавные латинские буквы A, B, C ...; "a" – строчные латинские буквы a, b, c ...; "I" – большие римские числа I,

II, III ...; "i" – маленькие римские числа i, ii, iii ...; "1" – арабские числа 1, 2, 3 ... (используется по умолчанию).

Элемент <li> упорядоченного (нумерованного) списка может иметь атрибут *value*. Данный атрибут изменяет порядок нумерации пунктов списка. В качестве значения указывается порядковый номер пункта.

Листинг 5. Нумерованный список.

```
<html>
<head>
<title> Упорядоченный список </title>
</head>
<body>
<ol type="a" start="3">
<li> Элемент списка под буквой C. </li>
<li value="5"> Элемент списка под буквой E.</li>
<li> Элемент списка под буквой F. </li>
</ol>
</body>
</html>
```

Результат листинга на Рис. 5.

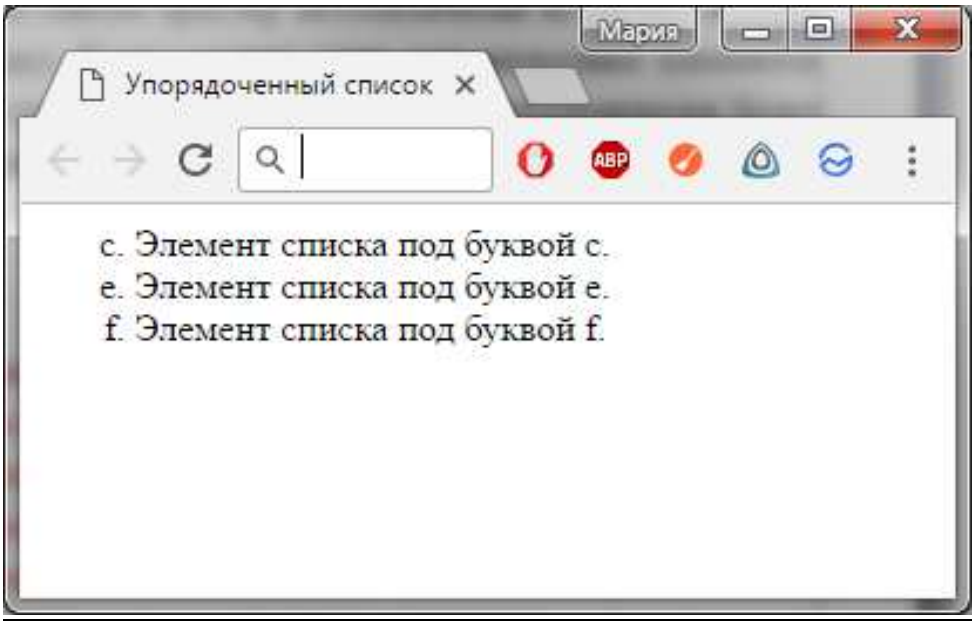

Рис. 5. Упорядоченный список.

Неупорядоченные (маркированные) списки

Маркированный список организуется элементом **<ul>** (Unsorted List). Каждый пункт неупорядоченного списка начинается с тега <li>. Для неупорядоченных списков браузером используются маркеры для пометки элементов списка. Вид маркера может быть указан разработчиком либо настраиваться пользователем браузера.

Атрибуты:

type - определяет внешний вид маркера. Может иметь значение: *disc* (вид по умолчанию), *circle* (круглый) или *square* (квадратный).

Список определений

Данный тип списка необходим для создание списков типа "термин"-"описание". Список организуется с помощью элемента **<dl>** . Каждый термин начинается тегом **<dt>** , а описание - тегом **<dd>.**

Листинг 6. Список определений.

```
...<body>
<dl>
<dt> <b> HTML (Hyper text markup language)</b>
<dd> независимый от платформ язык разметки гипер-
текста
<dt> <b> Гипертекст </b>
<dd> Текст, имеющий ссылки для автоматического пе-
рехода на другие тексты – гиперссылки.
</dl>
</body>...
```

Результат листинга на Рис. 6.

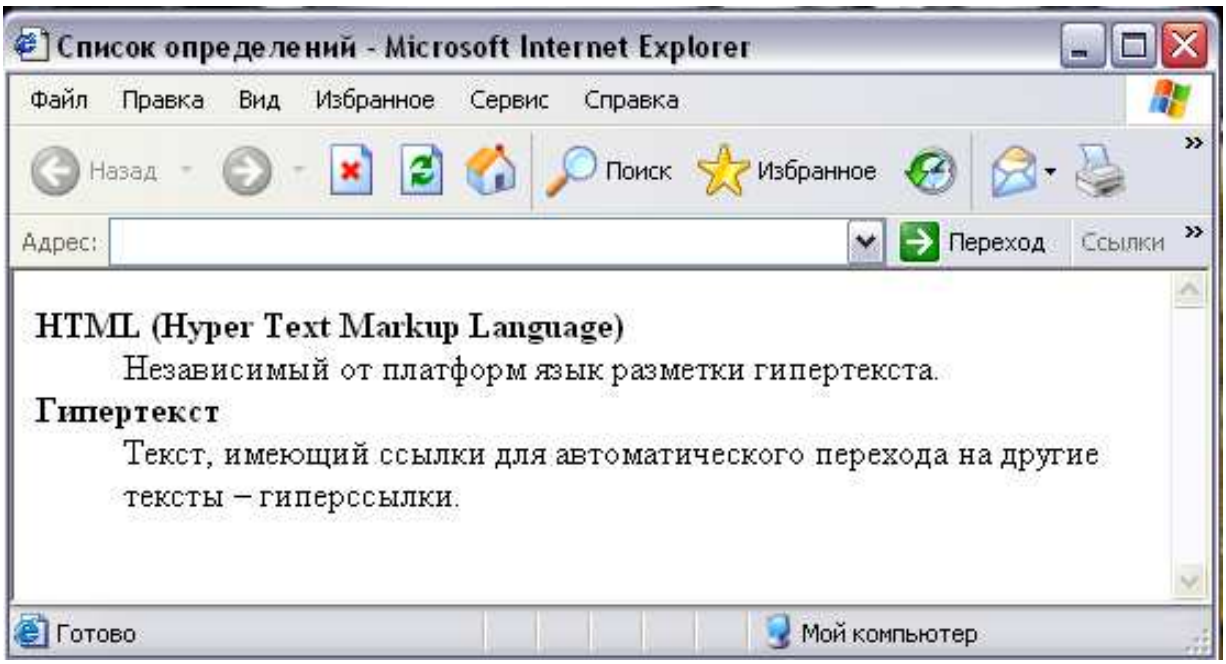

Рис. 6. Список определений.

!! *Создайте в* ***Блокноте*** *новый документ и сохраните его в своей папке под именем* ***Primer2.html.*** *Добавьте в текст документа следующее содержание:*

Списки в HTML

Язык HTML предоставляет несколько возможностей создания списков информации. В каждом списке должен быть один или несколько элементов списков. Списки могут содержать:

Неупорядоченную информацию. Упорядоченную информацию. Определения.

Упорядоченный список, создаваемый с помощью элемента ol, может содержать информацию, в которой важен порядок, например, рецепт:

Тщательно смешать сухие ингредиенты. Влить жидкость. Смешивать 10 минут. Выпекать в течение часа при температуре 300 градусов.

!! *Оформите соответствующим образом заголовок документа, абзацы, маркированный и нумерованный список, выбрав любой тип маркера и стиль нумерации. Просмотрите результат в браузере.*

!! *Создайте список определений по образцу Листинга 6. Просмотрите результат в браузере.*

1.6. Организация гипертекстовых ссылок

Гипертекстовые ссылки являются ключевым компонентом, делающим всемирную паутину привлекательной для пользователей. С добавлением гипертекстовых ссылок, набор документов становится связанным и структурированным, что позволяет пользователю получать необходимую ему информацию максимально быстро и удобно. Гиперссылка может указывать на другой HTML-документ, определенное место в данном документе или выполнять другие функции.

Для создания и использования гипертекстовых ссылок используется **элемент <a>** (от англ. "anchor" - якорь).

Атрибуты:

href – определяет адрес документа для перехода, преобразуя находящийся внутри элемента текст или изображение как гипертекстовую ссылку. Может принимать значения:

http://... – для создания ссылки на документ Всемирной паутины;

ftp://... – для создания ссылки на ftp-ресурс;

mailto:... – для создания ссылки на e-mail;

name – используется для определения маркера (якоря)внутри web-страницы, определяя находящееся содержимое элемента как возможный объект для ссылки. В качестве значения указывается имя маркера, применяемого в данном документе.

target – указывает окно (фрейм), в котором следует открывать документ. В качестве значения задается имя существующего фрейма, либо зарезервированное имя:

_self – для отображения документа в текущем окне;

_parent – для отображения документа во фрейме-родителе текущего окна;

_top – для отображения документа в полном окне браузера;

_blank – для отображения документа в новом окне.

Создание ссылок на внешний документ или ресурс WWW

При создании таких ссылок для элемента <a> обязательным является атрибут href. Текст и изображения, размещенные между тегами <a> ... , становятся активной зоной документа, чувствительной к щелчку мыши, который вызывает загрузку целевого документа. Текст гиперссылки выделяется подчеркиванием и цветом.

Например:

Ссылка `<a href="docs/title.html">документация</a>` будет ссылаться на файл *title.html* в подкаталоге *docs* (относительно текущего).

При создании ссылки на ресурс WWW необходим полный URL документа, а не только путь и имя файла.

Например:

```
<a href="http://www.yahoo.com / Arts/ Performin
Arts/Circuses/"> Список цирков Yahoo</a>
```

!! *Откройте в **Блокноте** файл **Primer1.html**, сохраненный в вашей папке. В конце документа создайте ссылку "Организация списков в HTML" на документ **Primer2.html** (так как оба файла находятся в одной папке, достаточно указать только имя файла без указания пути к файлу). Проверьте действие гиперссылки.*

!! *В документе **Primer1.html** создайте ссылку, открывающую документ в новом окне, на спецификацию HTML 5.1 (адрес ресурса* https://www.w3.org/TR/2016/WD-html51-20160310*). Проверьте действие гиперссылки.*

!! *В документе **Primer1.html** создайте ссылку на свой e-mail адрес. Проверьте действие гиперссылки. Закройте документ.*

Создание ссылок на точки внутри документа

Для удобства работы с большим (по длине) HTML-документом, в нем создаются ссылки на различные его участки или разделы с использованием специального скрытого ***маркера (якоря)*** для этих разделов. Такие маркеры позволяют перемещаться внутри документа без использования вертикально прокрутки окна браузера. При щелчке ле-

вой кнопки мыши по ссылке браузер перемещается в указанное место документа, в котором помещен маркер.

Для создания такой ссылки необходимо выполнить следующие шаги:

1. Создать маркер (якорь) раздела:

```
<a id="named_anchor"> Первая строка раздела или заголовок раздела </a>
```

2. Создать ссылку на данный якорь:

```
<a href="#named_anchor"> Текст ссылки </a>
```

Листинг 7. Организация ссылок внутри документа.

```
...<body>
<h2 align=center>Организация гипертекстовых ссылок</h2>
<a href="#ex1">Создание ссылок на внешний документ или ресурс www</a><br>
<a href="#ex2">Создание ссылок на точки внутри документа</a><br>
<p>Гипертекстовые ссылки являются ключевым компонентом...
...
<a id="ex1"><b><i>Создание ссылок на внешний документ или ресурс www</i></b></a>
<p>При создании таких ссылок для элемента <a>...
...
<a id="ex2">Создание ссылок на точки внутри документа
<p>Если html-документ слишком большой, то ...
...
</body>...
```

Результат листинга и действия ссылки на Рис. 7.

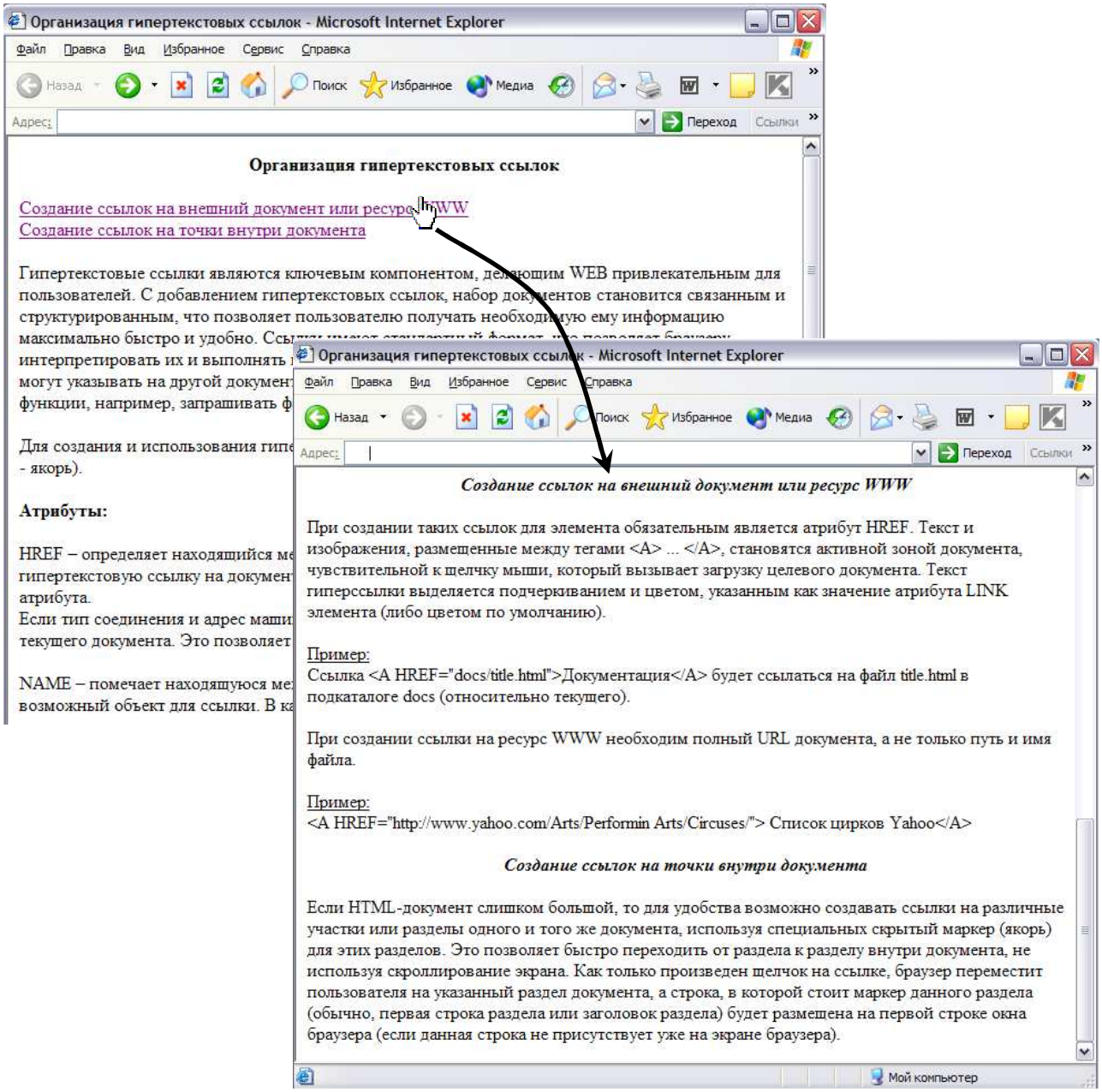

Рис. 7. Пример использования "якорей"

Символы "#ex1" и "#ex2" сообщают браузеру, что необходимо найти в данном HTML-документе якоря с именами "ex1" и "#ex2". Когда пользователь щелкнет мышью на ссылке *"Создание ссылок на внешний документ или ресурс WWW "*, браузер перейдет сразу к этому разделу.

Примечание. Маркер раздела может быть поставлен как в том же документе, который просматривается в текущий момент, так и в другом документе. Во втором случае браузер осуществит загрузку другого документа и перейдет к указанному для него разделу.

!! *Создайте в* ***Блокноте*** *новый документ и сохраните его в своей папке под именем* ***Primer3.html****. Добавьте в текст документа два небольших рассказа или стихотворения такого объема, чтобы для просмотра второго произведения требовалось прокручивать документ в браузере. В начале документа создайте ссылки на оба произведения, используя якоря для указания начала каждого произведения. В конце документа создайте ссылку на начало документа. Проверьте действие гиперссылок.*

1.7. Создание таблиц

Таблицы в HTML формируются нетрадиционным способом – построчно. Сначала с помощью элемента <tr> необходимо создать ряд таблицы, в который затем элементом <td> помещаются ячейки. В HTML таблицы используются не только для отображения таблиц как таковых, но и для дизайна. С помощью таблиц можно создать невидимый "каркас" страницы, помогающий расположить текст и изображения так, как вам нравится.

Элемент <table>

Элемент для создания таблицы. Обязательно должен иметь начальный и конечный теги. По умолчанию таблица печатается без рамки, а разметка осуществляется автоматически в зависимости от объема содержащейся в ней информации. Ячейки внутри таблицы создаются с помощью элементов <tr>, <td>, <th> и <caption>.

Атрибуты:

align – определяет горизонтальное выравнивания таблицы. Возможные значения: *left*, *center, right*. Значение по умолчанию – *left*.

border – определяет ширину внешней рамки таблицы (в пикселях). При значении равном нулю или при отсутствии этого атрибута рамка не отображается.

cellpadding – определяет расстояние (в пикселях) между рамкой каждой ячейки таблицы и содержащимся в ней материалом.

cellspacing – определяет расстояние (в пикселях) между границами соседних ячеек.

width – определяет ширину таблицы. Значение может задаваться в пикселях или в процентах по отношению к ширине окна браузера. По умолчанию значение атрибута определяется автоматически в зависимости от объема содержащегося в таблице материала.

height – определяет высоту таблицы. Значение может задаваться в пикселях или в процентах по отношению к высоте окна браузера По умолчанию значение атрибута определяется автоматически в зависимости от объема содержащегося в таблице материала.

bgcolor – определяет цвет фона ячеек таблицы.

background – позволяет заполнить фон таблицы рисунком. В качестве значения необходимо указать URL рисунка.

Элемент <caption>

Определяет заголовок таблицы в виде текста. Недопустимо применение в заголовке блочных элементов.

Атрибуты:

align – определяет вертикальное выравнивание заголовка таблицы. Может принимать значения:

top – размещение заголовка над таблицей (по умолчанию);

bottom – размещение заголовка под таблицей.

Элемент <tr> (table row)

Создает новый ряд (строку) ячеек таблицы. Ячейки в строке создаются с помощью элементов <td> и <th>.

Атрибуты:

align – определяет горизонтальное выравнивание содержимого всех ячеек строки. Может принимать значения: *left*, *center, right*.

valign – определяет вертикальное выравнивание содержимого всех ячеек строки. Может принимать значения: *top*, *bottom, middle*.

bgcolor – определяет цвет фона для всех ячеек строки.

Элементы <td> и <th> (table data & table head)

Элемент <td> создает ячейку с данными в текущей строке. Элемент <th> также создает ячейку, но определяет ее как ячейку-заголовок. Такое разграничение позволяет браузерам оформлять содержимое ячейки-заголовка и ячеек с данными разными шрифтами.

Атрибуты:

align – определяет горизонтальное выравнивание содержимого ячейки. Может принимать значения: *left, center, right*. По умолчанию выравнивание определяется значением атрибута *align* элемента <tr>. Если же и он не задан, то для <td> выполняется выравнивание по левому краю, а для <th> – центрирование.

valign – определяет вертикальное выравнивание содержимого ячейки. Может принимать значения: *top, bottom, middle*. По умолчанию происходит выравнивание по центру (valign="middle"), если значение этого атрибута не было задано ранее в элементе <tr>.

width – определяет ширину ячейки. Ширина задается в пикселях или в процентном отношении к ширине таблицы.

height – определяет высоту ячейки. Высота задается в пикселях или в процентном отношении к высоте таблицы.

colspan – задает объединение соседних ячеек строки в одну большего размера. По умолчанию имеет значение 1.

rowspan – задает объединение соседних ячеек столбца в одну большего размера. По умолчанию имеет значение 1.

nowrap – блокирует автоматический перенос слов в пределах текущей ячейки.

bgcolor – определяет цвет фона ячейки.

Листинг 8. Таблица с объединенными ячейками.

```
   ...<table border="4" cellspacing="0" align = "cen-
ter">
     <caption>заголовок таблицы </ caption >
          <tr>
   <td bgcolor="white">заголовок 1
   <td bgcolor="white">заголовок 2
           <tr>
   <td rowspan=3 bgcolor="white">ячейка 1
   <td>ячейка 2
           <tr>
```

```
   <td>ячейка 3
            <tr>
   <td>ячейка 4
           <tr>
   <td colspan=2 bgcolor="white" align = "center">
ячейка 5
   </table>...
```

Результат листинга и действия ссылки на Рис. 8.

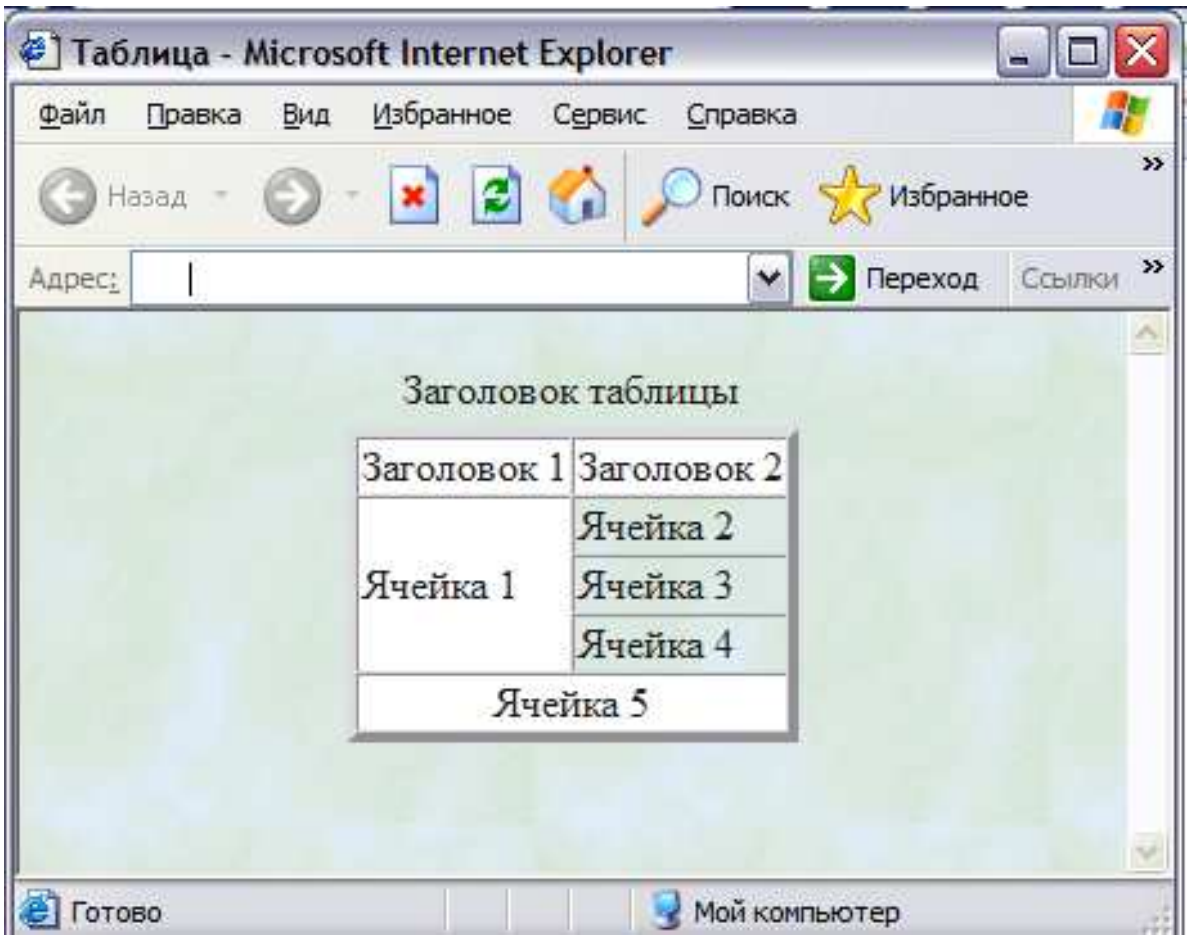

Рис. 8. Пример таблицы

Примечание. Границы ячейки будут отображаться только при наличии в ячейке какого-либо содержания. Для получения пустой ячейки с границами, следует разместить в ней специальный символ **` `** или небольшое gif-изображение с прозрачным фоном.

!! *Создайте в* ***Блокноте*** *новый документ и сохраните его в своей папке под именем* ***Primer4.html****. Создайте в документе таблицу по образцу в Листинге 8.*

1.8. Вставка в документ объектов

Объекты – это графические и мультимедийные вставки в HTML-документ, такие как изображения, Flash-анимация, видеоклипы, звуковые файлы и другие документы в формате HTML. Все мультимедиа-объекты в сети можно разделить на два основных типа: содержимое, обрабатываемое непосредственно браузером и содержимое, обрабатываемое дополнительными средствами. Браузер определяет способ обработки по типу содержимого файла и либо сам обрабатывает данные, либо передаёт их на обработку дополнительным приложениям. Тип содержимого файла определяется либо по его расширению, либо по специальному коду, вставляемому в исходный HTML-код документа.

Если браузер не в состоянии обработать мультимедиа-данные сам или при помощи встроенных модулей, то он запускает внешние приложения и передает им эти данные. Внешние приложения в своем окне обрабатывают содержимое для пользователя.

Ниже рассмотрены основные элементы для внедрения объектов в HTML-документ.

Элемент <img> (image)

Применяется для внедрения в HTML-документ изображений. Элемент не имеет конечного тега.

Атрибуты:

src – обязательный атрибут для указания адреса (URL) файла с изображением.

height и *width* – определяют ширину и высоту изображения соответственно. Если указанные значения не совпадают с реальным размером изображения, изображение масштабируется.

hspace и *vspace* – определяют отступ изображения (в пикселях) по горизонтали и вертикали от других объектов документа. Обычно используется при обтекании изображения текстом.

align – обязательный атрибут, определяет выравнивание изображения в документе. Возможные значения: *left* – выравнивает изображение по левому краю документа, прилегающий текст обтекает изображение справа; *right* – выравнивает изображение по правому краю документа, прилегающий текст обтекает изображение слева; *top* – строка, в которую вставлена картинка, будет выровнена по верхнему краю картинки и высота строки будет равна высоте самого рисунка.

middle – выравнивает базовую линию текущей текстовой строки с центром изображения.

bottom – выравнивает нижнюю кромку изображения с базовой линией текущей текстовой строки.

name – определяет имя изображения, уникальное для данного документа. Можно указать любое имя без пробелов с использованием латинских символов и цифр. Имя необходимо, если планируется осуществление доступа к изображению, например, из JavaScript-сценариев.

alt – определяет текст (альтернативный текст), отображаемый браузером на месте изображения, если браузер не может найти файл с изображением или включен в текстовый режим, или при наведении курсора на изображение. В качестве значения задается текст с описанием изображения.

border – определяет ширину рамки вокруг изображения в пикселях. Рамка возникает, только если изображение является гипертекстовой ссылкой. В таких случаях значение *border* обычно указывают равным нулю.

lowsrc – задает адрес (URL) файла с низкокачественной копией основного изображения, которое может быть визуализировано до изображения, заданного атрибутом *src*.

ismap – определяет изображение как навигационную карту (image map), обрабатываемую сервером. Используется совместно с элементом <a>. Данный атрибут не требует присвоения значения.

usemap – применяет к изображению навигационную карту (image map), заданную элементом <map>. В качестве значения задается имя карты с предшествующим ему знаком #.

Примечание: прописные и строчные буквы в данном атрибуте трактуются браузером как разные.

Листинг 9. Внедрение изображения.

```
    ...
    <p>Общая схема исходного кода HTML-документа выглядит следующим образом:
    <img src="htmlbody.gif" align="left" vspace="8" hspace="8" height="150"  width="75" alt="Структура HTML-документа">
    ...
```

Результат листинга на Рис. 9:

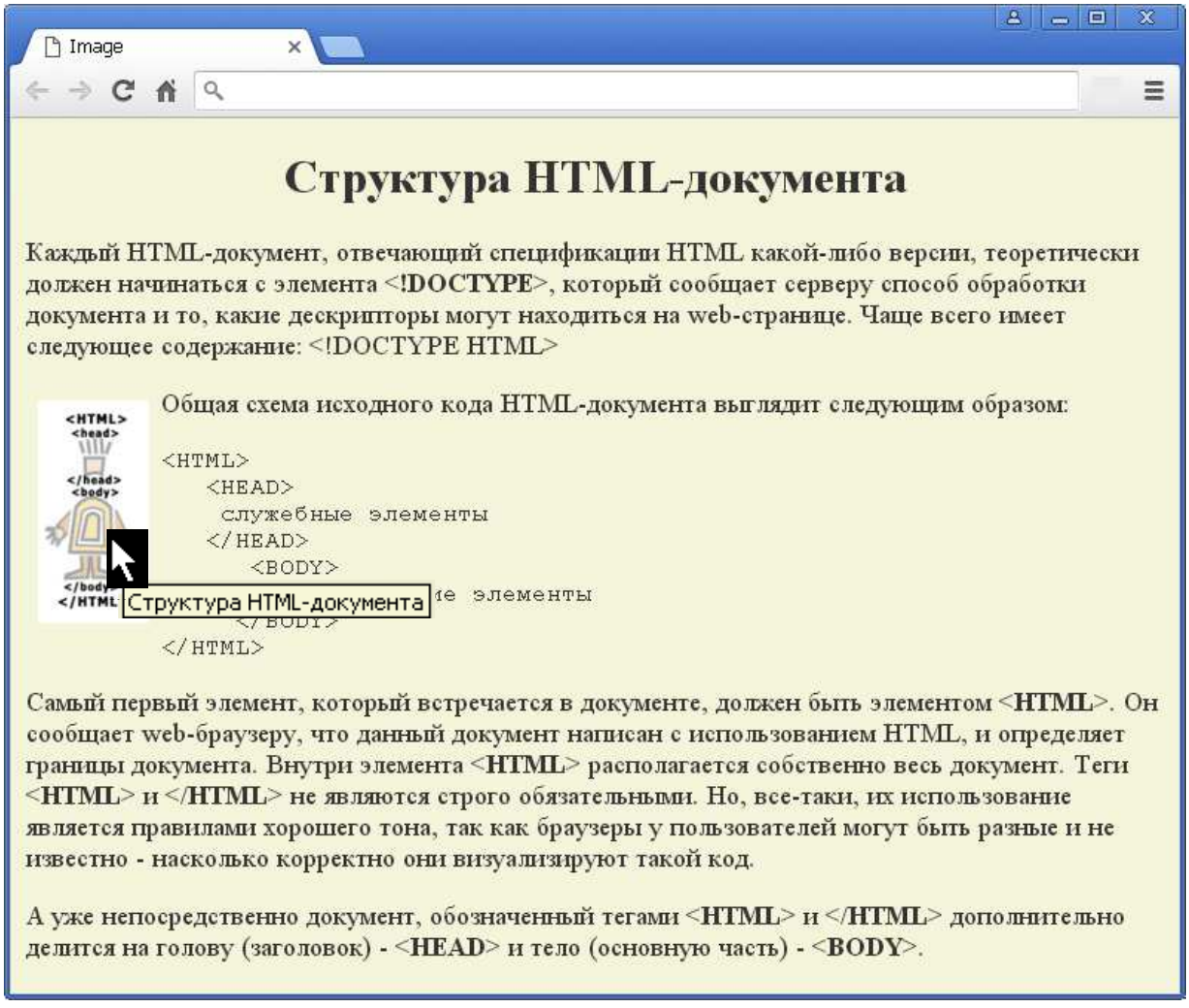

Рис. 9. Пример внедрения изображения.

Примечания:

☑ Обязательно после вставки изображения (сразу после элемента <img>) делать разрыв строки (элемент
).

☑ Обязательно указание значений атрибутов *hspace* и *vspace* (даже нулевые), так как некоторые браузеры по умолчанию присваивают им небольшое значение, не равное нулю.

☑ Необходимо всегда явно задавать размеры изображения в атрибутах *height* и *width*, резервируя тем самым место в окне браузера еще до загрузки изображения. В противном случае документ при загрузке каждой картинки будет заново перерисовываться.

☑ Желательно всегда описывать содержимое изображения, используя атрибут *alt*.

!! *Откройте в блокноте файл* ***Primer1.html****. Внедрите в документ любое изображение. Просмотрите результат в браузере.*

1.9. Элементы HTML5

В HTML5 появилось много новых элементов для смысловой разметки документа и добавления мультимедийных объектов. Рассмотрим некоторые из них.

Элементы <details> и <summary>

Данные элементы позволяют представить данные, которые можно скрыть или отобразить по щелчку мыши при работе пользователя с документом.

Содержимое элемента <details> по умолчанию не отображается в браузере. Данный элементы имеет один атрибут *open*, принимающий значение "*open*" для отображения содержимого элемента.

Синтаксис: `<details open="open">Текст</details>`.

С помощью элемента <summary> для элемента <details> можно указать заголовок скрываемого текста. В этом случае элемент <summary> располагается первым внутри элемента <details>. Не имеет атрибутов.

Листинг 10. Скрываемый текст.

```
...
<details>
<summary>Информация об авторе</summary>
<p>Петров И.В.<br>
e-mail: pochta@mail.ru</p>
</details>
...
```

Результат листинга на рис. 10.

!! *Добавьте в любой созданный ранее документ скрываемый текст по образцу из листинга 10 (введите свои фамилию и имея и адрес электронной почты). Просмотрите результат в браузере.*

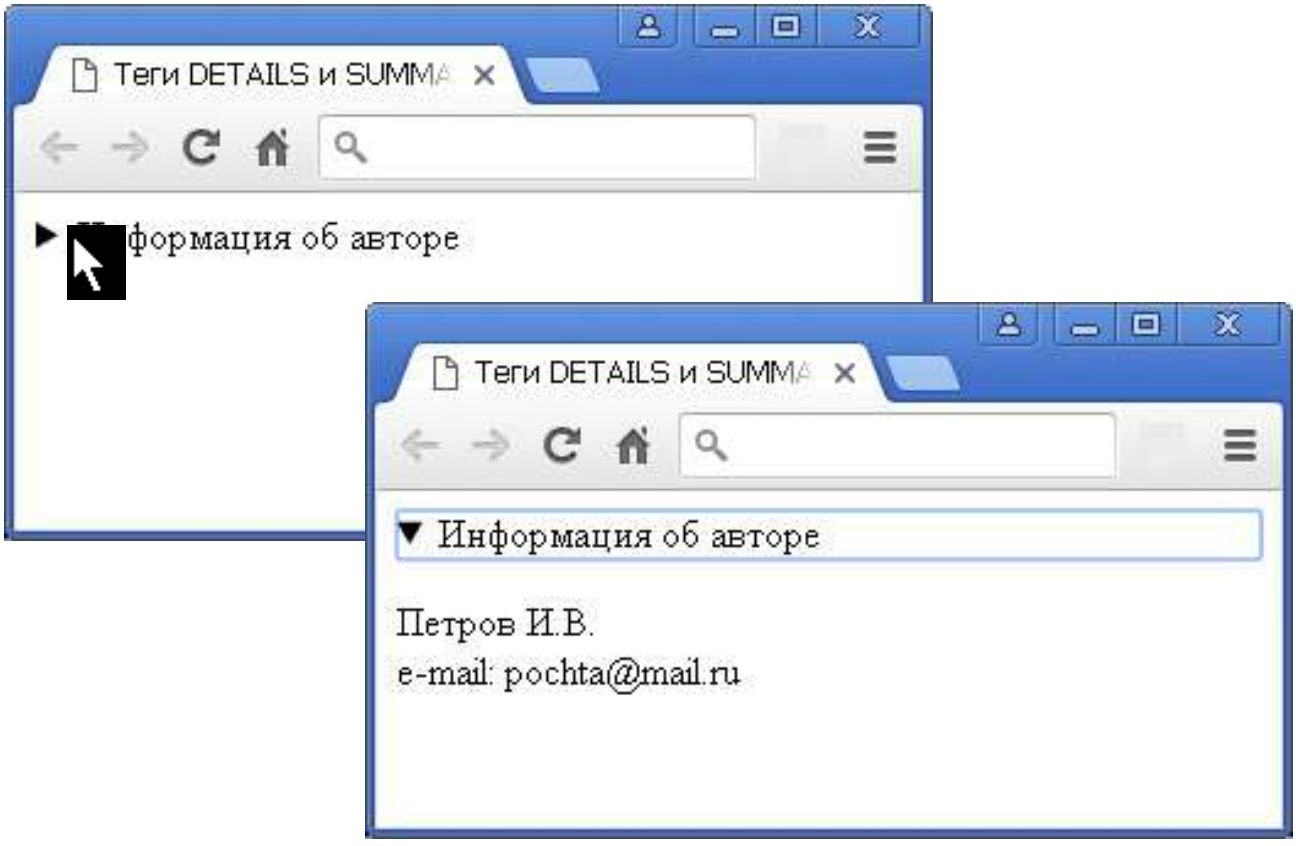

Рис. 10.Пример скрываемого текста.

Элементы <figure> и <figcaption>

В дополнение к элементу <img> могут использоваться теги <figcaption> для организации подписи к изображению (или другому

элементу) и <figure> для группирования изображения и подписи к нему. Не имеют атрибутов.

Листинг 11. Группировка изображения и подписи.

```
   ...
   <figure>
   <img src="htmlbody.gif" align="left" vspace="8"
hspace="8" height="150"  width="75" alt="">
   <figcaption>       Структура       HTML-документа
</figcaption>
   </figure>
   ...
```

Семантические элементы

Семантические теги позволяют создавать структуру документа, выделяя основные его части и определяя значимость каждой части web-документа, для более эффективной автоматизации их обработки. К таким элементам относятся теги **<article>, <aside>, <footer>, <header>, <main>, <nav> и <section>.** Все эти элементы не имеют атрибутов. Назначение элементов следующее (Рис. 11):

<article> - определят часть документа как содержание типа статьи, новости, записи блога и т.п.;

<aside> - определяет часть документа по бокам от основного контента (такой блок называют sidebar или боковой панелью);

<footer> - определяет часть документа как "подвал" web-страницы или раздела;

<header> - определяет часть документа как заголовок, "шапку" web-страницы или раздела;

<main> - определяет основную часть документа, не должен включать в себя остальные семантические элементы;

<nav> - определяет часть документа как элемент (панель) навигации, можно использовать несколько элементов <nav> в одном документе;

<section> - определяет часть документа как определенный раздел, обычно содержит заголовок, возможно расположение одного тега <section> внутри другого.

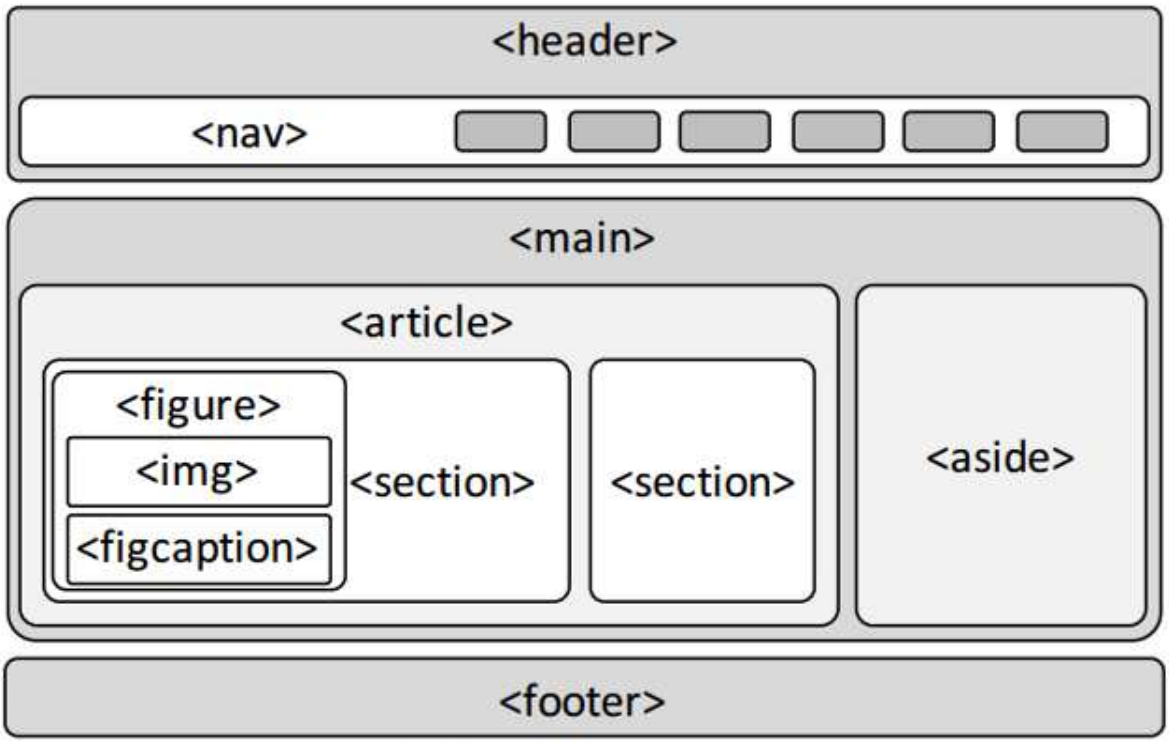

Рис. 11. Семантические теги в HTML-документе.

Элементы <audio> и <video>

Данные элементы применяются для добавления в документ аудио и видео файлов и позволяют воспроизводить их без использования дополнительных приложений. Путь к файлу задается либо атрибутом *src* либо с помощью вложенного тега **<source>.** Внутри элементов <audio> и <video> можно разместить текст, сообщающий пользователю о невозможности воспроизведения медиафайла.

Атрибуты:

src - необходим для указания пути к воспроизводимому файлу;

autoplay – указывает на автоматическое воспроизведение файла после загрузки web-страницы;

controls – для добавления панели управления к медиафайлу;

loop – включает автоматическое повторение воспроизведения файла после его завершения;

preload – указывает на загрузку медиафайла вместе с загрузкой web-страницы;

poster (только для <video>) – необходим для указания адреса изображения, которое отображается, если видео недоступно или не воспроизводится;

height (только для <video>) - указывает высоту области для воспроизведения видео;

width (только для <video>) - указывает ширину области для воспроизведения видео.

Вложенный тег **<source>** также имеет свои ***атрибуты***:

src - используется для указания пути к воспроизводимому файлу;

type – указывает MIME-тип мультмедийного источника;

media – для определения устройства, воспроизводимого файл.

Разные браузеры поддерживают различные аудио и видеокодеки. для возможности воспроизведения аудио или видео в разных браузерах медиафайлы кодируют различными кодеками и размещают в документе сразу несколько файлов.

Например:

```
...
<video width = "320" height = "240" controls au-
toplay>
<source src = "movie.mp4" type = "video/mp4">
<source src = "movie.ogg" type = "video/ogg">
Ваш браузер не поддерживает тег VIDEO
</video>
...
```

Элемент <embed>

Применяется при внедрении в страницы мультимедиа содержимого и других файлов. Данный элемент не является новшеством

HTML5, однако в предыдущих версиях HTML тег <embed> применялся редко, вместо него рекомендовалось применять элемент <object>.

Атрибуты:

Внедрение объекта происходит аналогично внедрению изображения с помощью элемента <img>. Поэтому элемент <embed> имеет ряд одинаковых с элементом <img> атрибутов, а именно: *align, width, height, hspace и vspace*.

src – обязательный атрибут для определения имени файла внедряемого объекта.

pluginspace – указывает на адрес (URL), по которому можно найти встраиваемый модуль (plug-in), необходимый для просмотра типа файлов, аналогичных указанному в атрибуте *src*.

hidden – определяет возможность отображения объекта. Может принимать значения: *true* – не отображать объект; *false* – отображать объект (используется по умолчанию).

type – определяет тип загружаемого объекта, для выбора браузером необходимого средства просмотра (plug-in). В качестве значения указывается зарегистрированный MIME-тип файла.

Синтаксис: `<embed src="video.mpg" width="300" height ="300" hspace="20" vspace="20">`

1.10. Создание форм

Формы являются наиболее важными интерактивными элементами HTML, которые позволяют разработчикам web-страниц интерактивно взаимодействовать с пользователями. С помощью форм пользователь может отправлять комментарии, запросы или проходить регистрацию. Содержимое формы может передаваться сценарию CGI, либо отправляться по электронной почте получателю.

Процесс создания формы состоит из двух этапов. Первый этап состоит в создании самой формы, а второй - в создании на сервере сценария CGI. Форма создается с использованием различных элемен-

тов и атрибутов, располагаемых внутри основного элемента **<form>**. Все элементы, используемые для создания форм называются *управляющими элементами*.

Элемент <form>

Является необходимым элементом для любой формы.

Атрибуты:

name – определяет имя формы, уникальное для данного документа. Используется, если в документе присутствует несколько форм.

action – обязательный атрибут, определяющий URL, по которому будет отправлено содержимое формы – путь к сценарию CGI сервера, обслуживающему данную форму, или адрес электронной почты.

method – определяет способ пересылки данных сценарию CGI. По умолчанию выбран протокол GET, но в большинстве случаев разработчики пользуются протоколом POST, который позволяет передавать большие объемы данных.

enctype - определяет способ кодирования содержимого формы, то есть сообщает браузеру о способе кодирования информации перед отсылкой серверу. По умолчанию используется значение "x-www-form-encoded".

target – определяет имя окна (фрейма), в которое возвращается результат обработки отправленной формы. Может принимать значения: *_self*, *_parent*, *_top*, *_blank* или явно указанное имя окна.

Элемент <textarea>

С помощью данного элемента создается область для ввода и просмотра текста.

Атрибуты:

name – обязательный атрибут, определяющий ключевое слово для обращения сценария к содержимому элемента;

rows - определяет высоту текстового поля в строках;

cols - определяет ширину текстового поля – в печатных символах.

wrap – определяет способ переноса слов в данной заполняемой форме. Может принимать значения: *off* – без переноса (значение по умолчанию); *virtual* – перенос слов только отображается, на сервер же

поступает неделимая строка; *physical* – перенос слов будет происходить во всех точках переноса.

Листинг 12. Создание текстового поля формы.

```
...<form>
Область для ввода и просмотра текста:
<br>
<textarea type="text" name="Область для ввода и
просмотра текста" rows=5 cols=50></textarea>
</form>...
```

Результат листинга на Рис. 12:

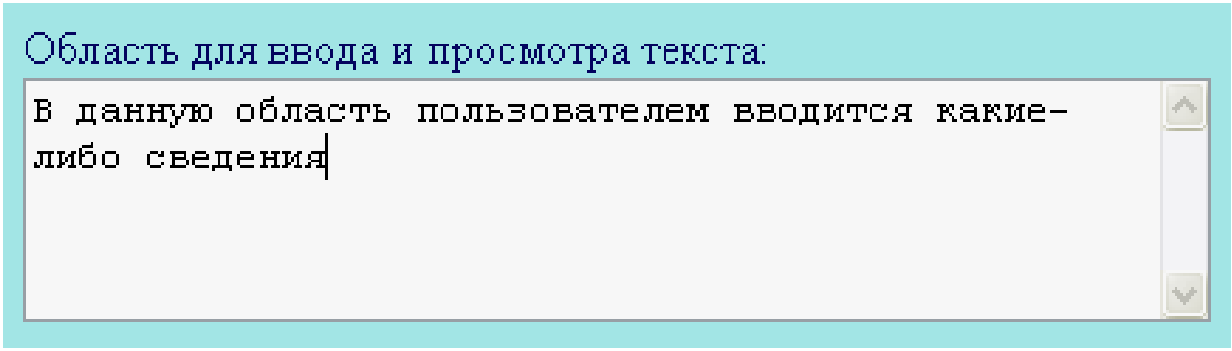

Рис. 12. Область для ввода текста.

Элемент <select>

Используется для создания раскрывающегося списка или меню элементов. Имеет вложенный элемент **<option>**.

Атрибуты:

name – определяет имя списка. Каждый выбранный пункт меню или списка при передаче на сервер будет иметь вид: *name/value*. Значение (*value*) формируется элементом <option>.

size – определяет количество видимых пунктов списка. Если значение этого атрибута больше единицы, то результатом будет список пунктов.

multiple – дает возможность выбора нескольких пунктов списка при удержании клавиши Ctrl. По умолчанию можно выбрать только один пункт меню.

Элемент <option>

Позволяет указать возможные варианты значения элементов списка <select>. Не имеет конечного тега.

Атрибуты:

selected – определяет пункт списка, который будет выбран изначально при загрузке документа. Если список имеет тип "один из многих", то атрибутом *selected* может быть помечен лишь один пунктов.

value – задает данному пункту значение, которое будет использовано наряду с другими сведениями о содержимом заполненной формы. При предоставлении информации на сервер это значение будет объединено со значением атрибута *name* в элементе SELECT.

Синтаксис: `<option value="n" selected>`

Листинг 13. Раскрывающийся список.

```
...
<form>
Раскрывающийся список или меню элементов:
<select name="Раскрывающийся список" size="1">
<option value=1>Элемент 1</option>
<option value=2>Элемент 2</option>
<option value=3>Элемент 3</option>
<option value=4>Элемент 4</option>
<option value=5>Элемент 5</option>
<option value=6>Элемент 6</option>
<option value=7>Элемент 7</option>
</select>
</form>
...
```

Результат листинга на Рис. 13:

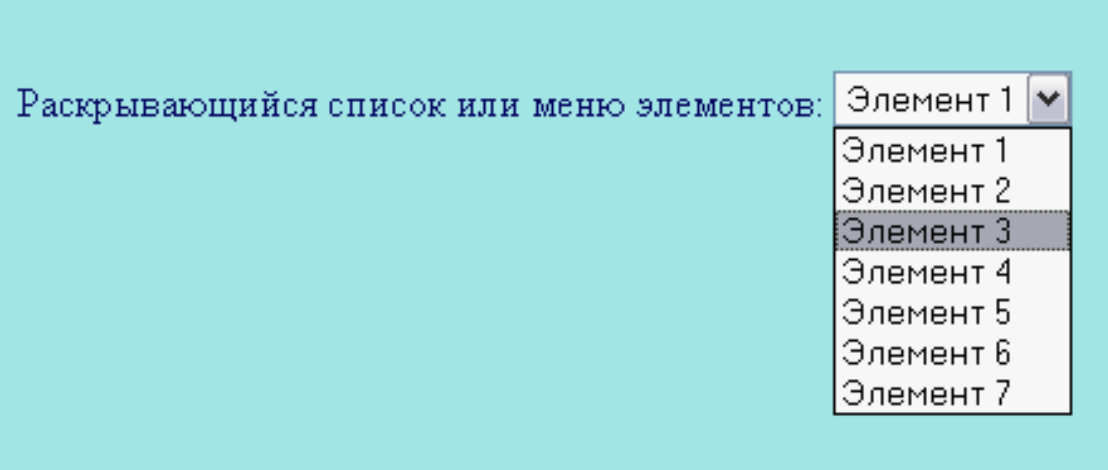

Рис. 13. Раскрывающийся список

Элемент <input>

Данный элемент является базовым для всех элементов формы. Применяется для добавления в форму кнопок, графических изображений, флажков, переключателей, паролей и текстовых полей. Не имеет конечного тега. Элемент <input> должен располагаться внутри элемента <form>.

Атрибуты:

name – определяет имя, используемое при передаче содержания данной формы на сервер. Этот атрибут необходим для большинства типов элемента <input> и обычно используется для идентификации поля или для группы полей, связанных логически.

type – определяет тип поля для ввода данных. По умолчанию имеет значение *text*. Может принимать значения:

text – создает поле ввода под одну строку текста. Как правило, используется совместно с атрибутами **size** и **maxlength**.

file – дает возможность пользователю приобщить файл к текущей форме. Возможно использование совместно с атрибутом **accept**.

password – создает однострочное поле ввода, в котором вводимые символы отображаются в виде знаков "*". Используется совместно с атрибутами **size** и **maxlength**.

checkbox – создает флажок, используемый для предоставления возможности односложного ответа типа "да/нет", "истина/ложь",

"больше/меньше" и т.д. или для ответа, который может одновременно принимать несколько значений. Обязательные совместно используемые атрибуты **name** и **value**.

radio – создает переключатель, в группе переключателей может быть выбран только один. Все переключатели в группе должны иметь одинаковые имена, но только выбранный в группе создает пару "name/value", которая будет послана на сервер. Для каждого переключателя указывается отдельный элемент INPUT.

submit – создает кнопку, при нажатии которой заполненная форма посылается на сервер. Атрибут **value** в данном случае изменяет надпись на кнопке, содержание которой, заданное по умолчанию, зависит от браузера.

image – создает графическую кнопку отправки данных на сервер. Местонахождение графического изображения можно задать с помощью атрибута **src**. При передаче данных серверу сообщаются координаты x и y той точки на изображении, где был произведен щелчок клавишей мыши. Координаты измеряются из верхнего левого угла изображения. При этом информация о поле типа *image* записывается в виде двух пар значений "*name/value"*. Значение *name* получается посредством добавления к названию соответствующего поля суффиксов ".x" (абсциссы), и ".y" (ординаты).

reset – создает кнопку, сбрасывающую значения полей формы (при отсутствии первоначальных значений) или возвращающую к их первоначальным значениям. При нажатии кнопки данные на сервер не отправляются. Надпись на кнопке может быть изменена с помощью атрибута **value**.

hidden – поля этого типа не отображаются на экране монитора, что позволяет разместить "секретную" информацию в рамках формы. Содержание этого поля посылается на сервер в виде name/value вместе с остальной информацией формы.

button – позволяет создать пользовательскую кнопку в HTML документе, что, при использовании JavaScript, добавляет форме функциональность. Атрибут *name* позволяет задать имя данной кнопке, которое

может быть использовано в сценарии. Атрибут **value** позволяет задать текст, который будет отображен на кнопке в документе.

value – задает текстовый заголовок для полей любого типа, в том числе и кнопок. Для таких полей как *checkbox* или *radio*, будет возвращено значение, заданное в атрибуте **value**.

checked – логический атрибут, указывает, что поля типов *checkbox* и/или *radio* атрибута **type** активизированы. Браузеры должны игнорировать этот атрибут для других типов управляющих элементов.

size – определяет размер поля в символах.

maxlength – определяет максимальное количество символов, которые можно ввести в текстовом поле. Оно может быть больше, чем количество символов, указанных в атрибуте **size**. По умолчанию количество символов не ограничено.

src – задает URL-адрес изображения, используемого при создании графической кнопки. Используется совместно с атрибутом type="image".

align – определяет способ вертикального выравнивания для изображений. Используется совместно с атрибутом type="image". Полностью аналогичен атрибуту *align* элемента img. По умолчанию имеет значение *bottom.*

accept – конкретизирует тип файла. Используется только совместно с параметром type="file". Значение задается в виде MIME-типа.

Листинг 14. Применение элемента <input>.

```
    …<form>
    <h3>Элемент &lt;INPUT&gt;:</h3>
    Текстовое поле:
    <input type="text" name="текст" size="50"><br>
    Переключатели:
    <input type="radio" name="переключатель"
value="01">Значение 1
    <input type="radio" name="переключатель"
value="02">Значение 2
    <input type="radio" name="переключатель"
value="03">Значение 3
```

```
    <br>
    Флажки:<br>
    <input type="checkbox" name="флажок"
value="01">Значение 1
    <br>
    <input type="checkbox" name="флажок"
value="02">Значение 2
    <br>
    <input type="checkbox" name="флажок"
value="03">Значение 3
    <br>
    <input type="checkbox" name="флажок"
value="04">Значение 4
    <br>
    Прикрепление файла:
    <input type="file" name="файл" size="40"><br>
    Кнопки отправки и очистки формы:
    <input type=submit value=Отправить>
    <input type="reset" value=Очистить><br>
    </form>…
```

Результат листинга на Рис. 14:

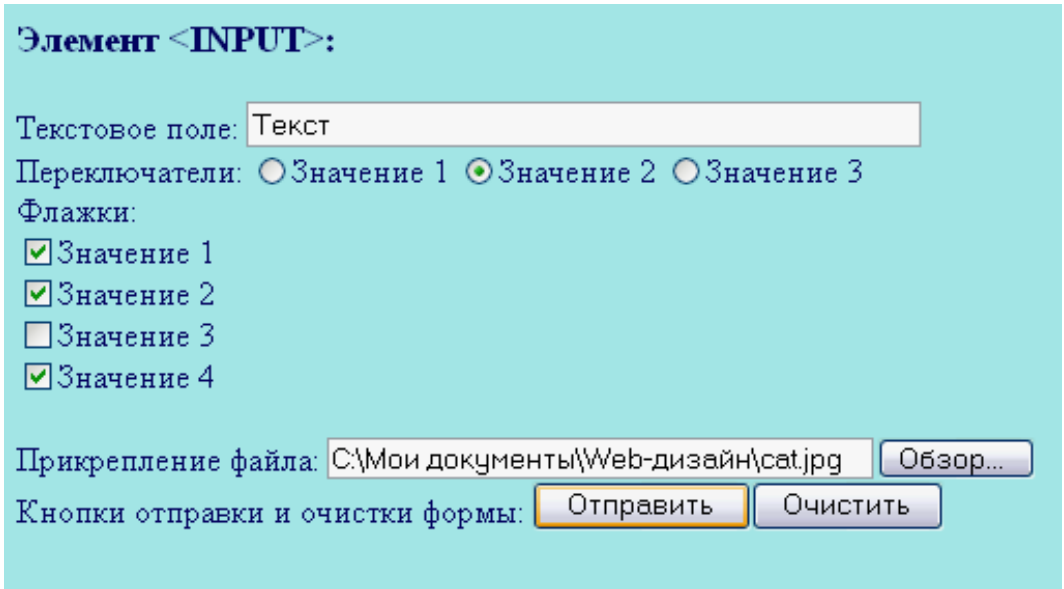

Рис. 14. Использование элемента <input>.

!! *Создайте в* ***Блокноте*** *новый документ и сохраните его в своей папке под именем* ***Form.html****. Используя вышеописанные элементы форм, создайте форму для заполнения (анкету) по образцу (Рис. 15). Просмотрите результат в браузере. Проверьте работоспособность формы.*

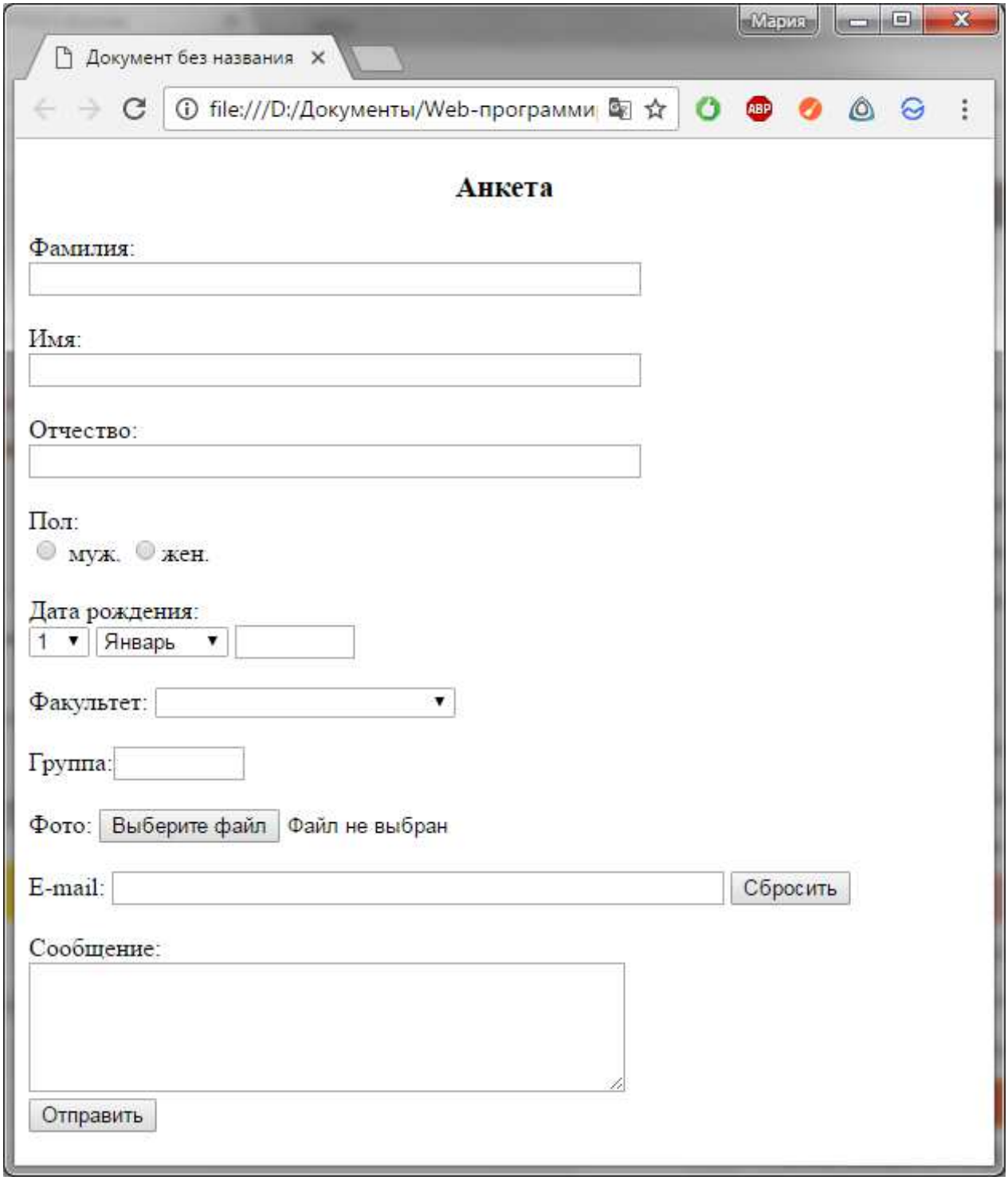

Рис. 15. Образец формы.

Контрольные вопросы

1. Что такое HTML?
2. Из чего состоят элементы языка разметки HTML? Поясните структуру HTML-документа.
3. Перечислите основные элементы заголовка HTML-документа?
4. Какие существуют виды списков в HTML-документе?
5. Какие элементы используются для создания списков?
6. Какой атрибут элемента <a> является обязательным при создании гиперссылок на внешний HTML-документ или ресурс Internet?
7. Как создаются ссылки на точки внутри HTML-документа?
8. Каким образом в HTML формируются таблицы?
9. Что относится к объектам в HTML?
10. С помощью каких элементов в HTML-документ можно вставить изображение?
11. Для чего применяются формы?
12. Какие элементы используются при создании форм?

2. Каскадные таблицы стилей (CSS)

2.1. Основные понятия CSS

Технология каскадных таблиц стилей (Cascade Style Sheets, CSS) была разработана в качестве дополнения к HTML с целью определения внешнего вида документов и сохранения за HTML только функции структурной разметки документов. Система CSS независима от HTML, имеет иной синтаксис и позволяет задавать параметры графического (также как и текстового, и звукового) представления элементов HTML.

В настоящее время официальной утвержденной спецификацией является версия CSS2.1 (уровень 2, ревизия 1, с внесенными правками от 12.04.2016). Рекомендация принята 7 июня 2011 года, основана на CSS2. Рекомендация CSS3 является разрабатываемой версией. Она сильно расширена по сравнению с предыдущими версиями. Главной особенностью CSS3 является возможность создавать анимированные

элементы без использования JavaScript, поддержка линейных и радиальных градиентов, теней, сглаживания и многое другое. В отличие от предыдущих версий спецификация разбита на модули, разработка и развитие которых идёт независимо. С 29 сентября 2011 г. разрабатывается рекомендация CSS4. Модули CSS4 построены на основе CSS3 и дополняют их новыми свойствами и значениями. Все они существуют пока в виде черновиков.

Каскадные таблицы стилей (CSS) представляют собой простую технологию определения и присвоения стилей к документу HTML. *Стиль* - это всё то, что определяет внешний вид документа HTML при его отображение в окне браузера. Стиль задаётся по определённым правилам, а таблица стилей – это набор правил форматирования, которые применяются к различным частям документа и описывают способ их представления на экране.

Синтаксис правил следующий:

```
селектор {свойство: значение}
```

Любое правило каскадных таблиц стилей состоит из двух частей: ***селектора*** и ***определения***. Селектором может быть любой тег (элемент) HTML для которого определение задаёт, каким образом необходимо его форматировать. Само определение, в свою очередь, также состоит из двух частей ***свойства*** и его ***значения***, разделённых знаком двоеточия. Назначение свойства очевидно из его названия [13].

Синтаксис правил не чувствителен к регистру. Селекторы, свойства и их значения можно задавать как строчными, так и прописными буквами, или в смешанном порядке.

Рекомендации Консорциума W3 позволяют использовать несколько таблиц стилей для управления форматированием одного документа HTML, а браузер по определённым правилам выстраивает приоритетность применения этих таблиц. Они выстраиваются неким "каскадом", отсюда и слово "каскадные" в названии технологии CSS.

2.2. Группирование и наследование стилей

Группирование стилей предусматривает возможность группирования нескольких селекторов при одном объявлении стиля или же группирование свойств и значений в объявлении стиля.

Селекторы группируются в виде списка элементов, разделённых запятыми:

```
h1,p,table {font-family: Arial}
```

Аналогично группируются определения, только в списке они разделяются точками с запятой (**;**):

```
h1 {font-family: Garamond;
    font-size: 14pt;
    color: #660066}
```

Некоторые свойства имеют собственный синтаксис группирования, связанный с заданием значений нескольких свойств в одном [13]. Например, свойства и значения, относящиеся к шрифтовому оформлению:

```
body {font: italic bold 15pt/18pt Verdana}
```

При таком группировании должен поддерживаться определенный порядок следования значений свойств шрифта: наклон (начертание) шрифта, насыщенность шрифта, размер символов/межстрочный интервал, семейство шрифтов.

При задании таблицы стилей можно свободно комбинировать все три правила группирования для уменьшения её размеров.

В HTML некоторые элементы могут содержать другие. В этом случае вложенный элемент является *потомком*, а элемент в который он вкладывается называется *предком*. В подобных случаях вложенный элемент **наследует** правила форматирования элемента-родителя.

Некоторые свойства не наследуются вложенными элементами от своих предков, например свойство *background*, но по умолчанию вложенные элементы будут отображаться с фоном родительского элемента [13]. Наследование удобно для задания свойств элемента, порождающего остальные элементы страницы HTML.

2.3. Связывание таблиц стилей с документами

Чтобы таблица стилей могла воздействовать на внешнее представление документа, браузер должен знать о её существовании. Для этого её необходимо связать с HTML-документом. Существует четыре способа связывания документа и таблицы стилей:

1. **Внедрение -** позволяет задавать все правила таблицы стилей непосредственно в самом HTML-документе.

2. **Связывание** - позволяет использовать одну таблицу стилей для форматирования многих страниц HTML.

3. **Импортирование** - позволяет встраивать в документ таблицу стилей, расположенную на сервере.

4. **Встраивание в элементы документа** - позволяет изменять форматирование конкретных элементов страницы.

Все способы добавления таблиц стилей свободно сочетаются в одном документе [13].

Внедрение таблиц стилей

Для внедрения таблицы стилей в HTML-документ используется элемент **<style>** в пределах раздела **<head>**. Сами правила заключаются в теги комментариев (**<!--…-->**):

```
<head>
   <style type="text/css">
     <!--
    b {text-transform: uppercase}
    p {background-color: lightgrey}
     -->
   </style>
</head>
```

В приведённом выше примере внедренная таблица стилей определяет отображение всех абзацев в документе (элемент **<p>**) на сером фоне, полужирный текст (элемент **<b>**) документа, будет отображаться прописными буквами, даже если в документе он задан строчными [13].

!! *В программе* ***Блокнот*** *откройте файл* ***Primer1.html****, созданный ранее.*

!! *Создайте в разделе <head> таблицу стилей с правилами, определяющими следующее оформление всех абзацев (элемент <p>) документа:*

параметры шрифта (свойство font):

гарнитура шрифта – Arial;

размер шрифта – 14 pt;

цвет шрифта – темно-красный (darkred);

параметры абзаца:

выравнивание (text-align) – по ширине (justify);

отступ первой строки (text-indent) – 20 px.

Сохраните документ. Просмотрите результат в браузере.

Связывание и импортирование таблиц стилей

При связывании стиля все стилевые описания производятся по тем же правилам, что и при внедрении, но хранятся они в отдельном файле, имеющем расширение **.css**. Сам файл должен находиться в корневом каталоге сайта, в противном случае нужно корректно указать связь с ним.

Например, файл с именем *mystyle.css* содержит описания стилей:

```
table {font-family: arial}
a {color: #ff0000; text-decoration: none}
p {margin-left: 20pt}
```

Документ HTML, который ссылается на этот файл, должен содержать в разделе **<head>** следующую ссылку:

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

Преимуществом связывания является то, что стилевые описания применяются ко всем страницам сайта и при необходимости изменения оформления достаточно внести изменения в файл с таблицей стилей, вместо того, чтобы исправлять все страницы сайта. Этот ме-

тод так же дает возможность ссылаться на таблицы стилей, которые располагаются не на том же сервере, что и сама страница.

!! *Создайте в редакторе* ***Блокнот*** *новый текстовый документ и сохраните его в своей папке под именем* ***mystyle.css***.

!! *Создайте в этом файле таблицу стилей с правилами, определяющими следующее оформление тела документа (<body>):*
параметры шрифта (свойство font):

цвет текста – темно-синий (Darkblue);

гарнитура шрифта – Courier;

стиль шрифта – курсив (italic);

параметры фона:

фон страницы (background-color) – lightblue.

Сохраните документ.

!! *Свяжите созданную таблицу стилей с файлом* ***Primer1.html.*** *Просмотрите результат в браузере. Обратите внимание, что параметры абзацев, к которым применена внедренная в документ таблица стилей, не изменились.*

!! *Свяжите созданную таблицу стилей с файлами* ***Primer2.html*** *и* ***Primer3.html.*** *Просмотрите результат в браузере.*

!! *Отредактируйте файл таблицы стилей, внеся следующие изменения:*

гарнитура шрифта – Times;

стиль шрифта – обычный (normal).

Просмотрите изменения в связанных с таблицей стилей документах.

При импортировании таблиц стилей в документ, также как и при связывании встраивается внешняя таблица стилей, но с помощью свойства @import таблицы стилей. Его следует задавать в начале стилевого блока <style> или связываемой таблицы стилей перед заданием остальных правил. Значением свойства @import является URL-адрес файла таблицы стилей [13]:

```
<head>
<style type="text/css">
@import url(http://www.mypage.ru/style.css);
    p {color: black; font-size:12pt}
    table {font-size: 10pt}
    a {color: #000080}
</style>
</head>
```

Преимуществом данного способа является возможность добавить на страницу сразу несколько таблиц стилей.

Встраивание стилей в элементы HTML

Описание стиля можно встроить в различные элементы HTML, для которых стиль имеет смысл, например, абзацев, заголовков, гиперссылок или ячеек таблицы.

При встраивании в элемент добавляется атрибут style, значением которого является свойство и его значение, разделенные двоеточием.

Например:

```
<p style="font-family:Verdana"> Текст абзаца шрифтом Verdana </p>
```

Если внедренные, связанные и импортируемые таблицы стилей влияют на форматирование всех элементов документа, для которых определенны в таблице правила, то встраивание определений стилей в конкретный тег влияет на отображение только элемента, определяемого данным тегом.

!! *Отредактируйте HTML-код документа* ***Primer2.html*** *с помощью встраивания стиля в теги, чтобы в списке определений все термины (элемент <dt>) отображались следующим образом:*

размер шрифта – 14 pt;

цвет шрифта – красный (red);

начертание шрифта – полужирный (bold).

Просмотрите результат в браузере.

2.4. Селекторы CSS

Правила каскадных таблиц стилей, в которых в качестве селектора используются теги HTML, влияют на отображение всех элементов заданного типа в документе. Но бывает необходимость отображать не все элементы данного типа одинаково. Можно задать для них правило форматирования непосредственно в теге, а можно применить специальные селекторы.

Селектор class. ***Класс*** позволяет задать разные правила форматирования для одного элемента определённого типа или всех элементов документа. Имя класса указывается в селекторе правила после имени тега и отделяется от него точкой. Можно определить несколько правил форматирования для одного элемента и с помощью атрибута class соответствующего тега применять разные правила форматирования в документе [13].

Например, можно оформить заголовки первого уровня с различными видами выравнивания – по левому и правому краям:

```
<style  type="text/css">
<!--
  h1.left {color:#ff0000;
        text-align: left}
  h1.right {color:#0000ff;
         text-align: right}
-->
</style>
```

В тексте документа ссылка на соответствующий класс задаётся в атрибуте class:

```
<h1 class="left">Текст заголовка с левосторонним выравниванием</h1>
<h1 class="right">Текст заголовка с правосторонним выравниванием</h1>
```

!! *Создайте во внешней таблице стилей (**mystyle.css**) два класса для элемента списка (<li>). Имена классам задайте по своему усмотрению.*

Первый класс с параметрами:

цвет текста – Forestgreen;

вид текста (text-transform) – все прописные буквы (uppercase);

Второй класс с параметрами:

цвет текста – Maroon;

оформление текста (text-decoration)– с подчеркиванием (underline);

расстояние между символами (letter-spacing)– 1 pt.

!! *Примените созданные классы к разным элементам нумерованного списка в документе* ***Primer2.html****. Просмотрите результат в браузере.*

В приведённом выше примере классы использовались для различного отображения элементов одного типа. Для использования класса к любому элементу документа, необходимо в селекторе указать имя класса с лидирующей точкой без указания конкретного элемента:

```
<style  type="text/css">
<!--
  .left {color:#ff0000;
      text-align: left}
  .right {color:#0000ff;
       text-align: right}
-->
</style>
```

В данном случае два класса `left` и `right` можно применять к любым элементам документа.

!! *Создайте во внешней таблице стилей (****mystyle.css****) класс под именем* ***.border*** *с параметрами:*

стиль границы элемента (border-style) – двойная линия (double);

цвет границы элемента (border-color) – Magenta;

отступ от границы элемента до его содержимого (padding) – 10px.

!! *Примените созданный стиль к различным элементам (абзаца, заголовкам, элементам списков) связанных документов. Просмотрите результат в браузере.*

Селектор id (идентификатор). Кроме селектора класса в правиле стиля можно использовать ***селектор id - идентификатор***. Идентификаторы позволяют назначать свойства отдельным компонентам HTML без использования прочих стандартных методов, чаще всего используются для позиционирования элементов. Идентификатор имеет собственное имя, состоящее из цифр и символов латинского алфавита. В таблице стилей идентификатор начинается с символа **#**. В документе HTML идентификатор задается атрибутом id. Атрибут id можно применить к любому элементу документа.

Например:

```
<style type="text/css">
<!--
#pict {position:absolute;
    top:100;
    left:50}
-->
</style>
```

В тексте документа ссылка на идентификатор задаётся в атрибуте id:

```
<div id="pict">
<img src="cats.jpg" alt="cats">
</div>
```

В данном примере идентификатор `#pict` используется для позиционирования графического изображения, которое помещено в элемент .

Контекстные селекторы. При разработке страниц HTML, как уже говорилось, часто приходиться одни элементы вкладывать в другие. Некоторые свойства элементов-предков наследуются вложенными элементами. Если необходимо, чтобы элементы-потомки, вложен-

ные в элементы-предки, отображались одним образом, а в других частях документа аналогичные элементы отображались по-другому, применяют ***контекстные селекторы***.

Контекстный селектор состоит из нескольких простых, разделённых пробелами. Интерпретатор браузера просматривает все элементы-предки, находит элементы-потомки и применяет к ним указанное правило форматирования.

Например:

```
...
<style  type="text/css">
 <!--
P A {color: green; text-decoration: none}
 -->
</style>
...
<body>
<p> Для регистрации перейдите по ссылке
<a href="registration.html">Регистрация</a>
</p>
<a href="index.html">На главную страницу</A>
</body>
...
```

В данном примере ссылка для регистрации будет отображаться зеленым цветом без подчеркивания, а ссылка на главную страницу сайта будет оформлена по умолчанию (синим цветом с подчеркиванием).

!! *Оформите элементы маркированных списков в документе* ***Primer2.html*** *следующим образом с использованием контекстных селекторов:*

цвет текста – Black;

стиль шрифта – курсив (italic);

вид маркера (list-style-type) – квадратный (*square).*

Просмотрите результат в браузере.

Псевдоклассы. Псевдоклассы определяют динамическое состояние элементов, которое изменяется со временем или с помощью действий пользователя, а также положение в дереве документа. Примером такого состояния служит текстовая ссылка, которая меняет свой цвет при наведении на нее курсора мыши. При использовании псевдоклассов браузер не перегружает текущий документ, поэтому с помощью псевдоклассов можно получить разные динамические эффекты на странице [12].

Синтаксис применения псевдоклассов:

```
Селектор:Псевдокласс {правило стиля}
```

Псевдоклассы можно применять к именам идентификаторов или классов (`A.menu:hover {color: green}`), а также к контекстным селекторам (`.menu A:hover {background: #fc0}`). При отсутствии селектора перед псевдоклассом (`:hover`), он будет применяться ко всем элементам документа.

Существующие псевдоклассы:

`:focus` - применяется к элементу при получении им фокуса. Например, для текстового поля формы получение фокуса означает, что курсор установлен в поле, и с помощью клавиатуры можно вводить в него текст.

`:first-child` - применяется к первому дочернему элементу селектора, который расположен в дереве элементов документа.

`:lang` - определяет язык, который используется в документе или его фрагменте. В коде HTML язык устанавливается через атрибут charset тега <meta>. С помощью псевдокласса :lang можно задавать определенные настройки, характерные для разных языков [12].

Псевдоклассы ссылок. Для элемента **<a>**, описывающего гиперссылки, можно использовать несколько псевдоклассов, определяющих оформление непосещённых, активных, посещённых ссылок, а также ссылок, над которыми находится курсор мыши (т.е. свойства ссылок "при наведении" курсора мыши).

`a:link` - непосещённая ссылка;

`a:visited` - посещённая ссылка;

`a:active` - активная ссылка;

`a:hover` - ссылка, выбранная курсором мыши.

Любую ссылку в документе можно отнести к какому-либо псевдоклассу. Так как псевдоклассы ссылок могут использоваться только для элемента <a>, в описании правила в селекторе этот элемент допускается не указывать. Например, данные правила можно считать равносильными:

```
a:link {color: red}
 :link {color: red}
```

!! *Создайте во внешней таблице стилей (**mystyle.css**) псевдоклассы ссылок для различного оформления всех видов ссылок. Задайте следующие параметры:*

для непосещенной ссылки – без подчеркивания (text-decoration:none), полужирный шрифт (bold);

для посещенной ссылки – без подчеркивания, полужирный шрифт;

для активной ссылки – с подчеркиванием, обычный шрифт, цвет Red;

для ссылки, выбранной курсором – с подчеркиванием, обычный шрифт, цвет Black.

Просмотрите результат в браузере для всех связанных документов.

Дочерние селекторы. *Дочерний селектор* - который расположен прямо внутри родительского элемента. Если в свою очередь, родительский элемент тоже вложен в другой, то более высокий по иерархии элемент, первому (дочернему) родительским не является. Синтаксис применения таких селекторов следующий:

```
Селектор 1 > Селектор 2 {правило стиля}
```

Стиль применяется к Селектору 2, но только в том случае, если он является дочерним для Селектора 1.

По своей логике дочерние селекторы похожи на селекторы контекстные. Разница между ними следующая. Стиль к дочернему селектору применяется только в том случае, когда он является прямым потомком, иными словами, непосредственно располагается внутри родительского элемента. Для контекстного селектора же допустим любой уровень вложенности [5].

Например:

```
<style type="text/css">
 div > b {color: red}
 </style>
...
 <body>
 <div>
 <b>Дочерний элемент</b>, выделенный жирным шрифтом, отображается красным цветом
<p><b>Данный элемент</b> не является дочерним для элемента DIV</p>
 </div>
</body>
```

В данном примере жирный текст, находящийся в элементе div, который будет для элемента b родительским, будет отображаться красным цветом. Во втором же случае элемент b является дочерним для элемента p, поэтому отображается по умолчанию.

Соседние селекторы. *Соседними* называются элементы web-страницы, когда они следуют непосредственно друг за другом в коде документа. Для управления стилем соседних элементов используется символ плюса (+), который устанавливается между двумя селекторами. Общий синтаксис следующий:

```
Селектор 1 + Селектор 2 {правило стиля}
```

Пробелы вокруг плюса не обязательны, стиль при такой записи применяется к Селектору 2, но только в том случае, если он является соседним для Селектора 1 и следует сразу после него.

Например:

```
   <style type="text/css">
    b + i {color: red}
    </style>
   ...
    <body>
    <p><b>Изменение цвета с применением спосо-
ба</b><i> соседнего селектора</i></p>
    <p><b>Изменение цвета с применением способа</b>
соседнего селектора не происходит.</p>
   </body>
```

В первом случае селекторы b и i являются соседними, т.к. идут следом друг за другом, поэтому применяется стиль соседних селекторов. Во втором случае – стиль соседних селекторов не применяется, т.к. между i и b присутствует элемент p.

Универсальный селектор. Иногда требуется установить одновременно один стиль для всех элементов web-страницы, например, задать шрифт или начертание текста. В этом случае поможет универсальный селектор, который соответствует любому элементу web-страницы. Для обозначения универсального селектора применяется символ звездочки (*) и в общем случае синтаксис будет следующий [12]:

```
* {правило стиля}
```

Комментарии. Таблица стилей может содержать введенные пользователем комментарии, поясняющие назначение тех или иных участков кода. Комментарии в стандарте CSS заключаются в последовательность символов /* и */:

```
 :visited {color: green} /*оформление посещенной ссылки*/
```

2.5. Каскады таблиц

Одной из возможностей CSS является возможность создавать каскады таблиц стилей. Это значит, что в браузерах предполагается воз-

можность одновременного использования различных таблиц стилей для одного документа.

У каждого браузера свой стиль, определённый по умолчанию, для представления страниц. Когда браузер загружает web-страницу, он показывает её своим стилем, определённым по умолчанию. Если отображается страница, которая ссылается на CSS, на экране будут отображены особенности, заданные в каскадной таблице стилей. Обычная страница отображается так, как это установлено в браузере, а страница с использованием CSS - так, как того захочет её разработчик.

Какой системе стилей будет отдано предпочтение, задаётся некоторой системой правил. Браузер, выбирая из предложенных стилей, сначала определяет, нет ли противоречий (ошибок) в задании параметров какого-нибудь элемента. Если они присутствуют, то используются параметры по умолчанию. Если их нет, используются параметры, заданные автором страницы. Если конфликтуют два стиля и один применяется в конкретной ситуации, а другой - во всех остальных случаях, предпочтение отдаётся первому.

Чтобы в подобной ситуации браузер понимал, какое в итоге правило применять к элементу, и не возникало конфликтов в поведении разных браузеров, введены определенные приоритеты. Чем ниже в списке находится пункт, тем ниже его приоритет, и наоборот [12].

1. Стиль пользователя с добавлением !important.
2. Стиль автора с добавлением !important.
3. Стиль автора.
4. Стиль пользователя.
5. Стиль браузера.

Наиболее низкий приоритет имеет стиль браузера - оформление, применяемое к элементам web-страницы браузером по умолчанию. Такое оформление наблюдается без использования в HTML-документе стилевых правил.

Ключевое слово **!important** играет роль в том случае, когда пользователи подключают свою собственную таблицу стилей. Если возникает противоречие, когда стиль автора страницы и пользователя для

одного и того же элемента не совпадает, то !important позволяет повысить приоритет стиля [12].

Синтаксис: `Свойство:значение !important`

Повышение важности требуется не только для регулирования приоритета между авторской и пользовательской таблицей стилей, но и для повышения специфичности определенного селектора.

Если к одному элементу одновременно применяются противоречивые стилевые правила, то более высокий приоритет имеет правило, у которого значение **специфичности** селектора больше. Идентификаторы имеют большее значение специфичности, чем классы и псевдоклассы, которые в свою очередь имеют большую специфичность по отношению к селекторам тегов и псевдоэлементам. Встроенный стиль, добавляемый к тегу через атрибут *style*, имеет наибольшую специфичность, поэтому всегда перекрывает связанные и глобальные стили. Однако добавление !important перекрывает в том числе и встроенные стили. Если два селектора имеют одинаковую специфичность, то применяться будет тот стиль, что определен в коде ниже [12].

2.6. Модель форматирования CSS

Для грамотного использования правил форматирования необходимо представлять, каким образом в таблицах стилей форматируются элементы, а именно что возможно изменить в отображении элементов.

Модель форматирования каскадных таблиц стилей предполагает представление любого элемента HTML в окружении вложенными прямоугольными блоками (Рис. 16).

Блок содержимого элемента (самый внутренний блок) отделён от *границы* отступами. Самым внешним блоком является *поле*. Свойства таблиц стилей позволяют устанавливать размеры и цвета всех блоков, составляющих в сумме отображаемый элемент. Поле всегда является прозрачным прямоугольником, поэтому его цвет наследует цвет элемента-родителя (для абзаца это элемент <body>). *Отступ* всегда имеет цвет фона самого элемента. Цвет и ширину границы

можно указать отдельно, при её помощи можно создать рамку вокруг элемента [13].

Рис. 16. Модель форматирования элементов HTML.

Все перечисленные блоки в совокупности составляют блок форматирования элемента или блок отображения элемента, т.е. видимое в окне браузера изображение элемента. Размеры блока форматирования/отображения элемента складываются из размеров самого элемента и размеров отступов, границы и полей. С точки зрения процесса форматирования документа существуют два типа элементов: блоковые и встроенные [13].

Блоковые элементы

Каждый элемент HTML-документа согласно модели форматирования имеет свойство *display*, значение которого определяет отображение элемента (*none* – элемент не отображается) и его тип: блок (*block*), список (*list-item*) или встроеннй элемент (*inline*).

Элементы, имеющие значение свойства *display*, отличное от *none*, являются блоковыми элементами. Процесс форматирования таких элементов подразумевает установку значений параметров вложенных блоков, составляющих элемент в целом. На

Рис. 17 представлены все возможные параметры блоковых элементов согласно модели форматирования каскадных таблиц стилей.

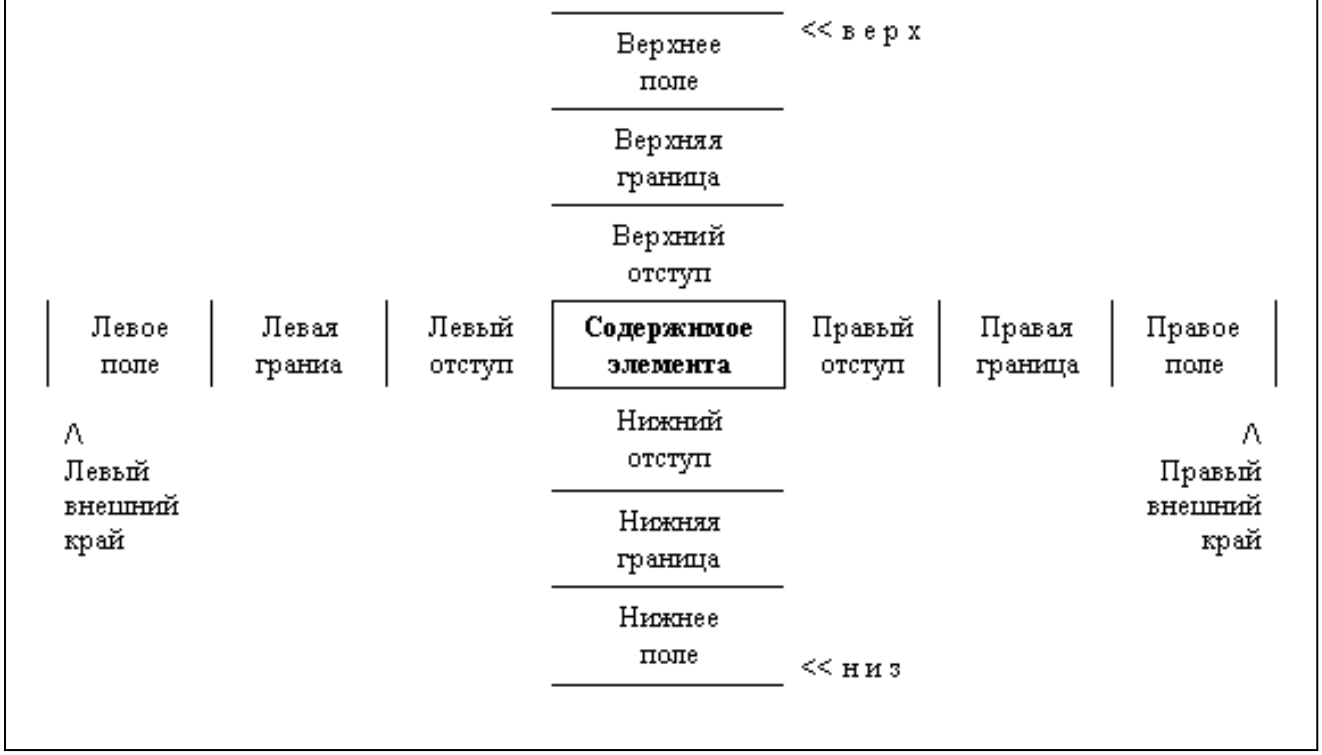

Рис. 17. Параметры блоковых элементов

Свойства полей, отступов и границы определяются следующим облазом:

```
{Область-Сторона-Свойство: значение}
```

Свойство *float* может переводить любой элемент в разряд "плавающих". Это приводит к тому, что указанный элемент выводится из нормального потока отображения и форматируется как блоковый элемент. Например, установка свойства *float* элемента <img> равным *left* позволяет создать буквицу при выводе абзаца текста. При значении свойства *float* равным *left*, элемент сдвигается влево до поля, отступа или границы другого бокового элемента, а нормальный поток отображения будет обтекать его с правой стороны [13].

Встроенные элементы

Остальные элементы, не форматируемые в виде блока, являются встроенными элементами (*inline*). Вместе с другими элементами они используют область строки.

Для задания значений свойств, определяющих некоторые размеры, в каскадных таблицах стилей применяются относительные и абсолютные единицы измерения. **Относительные единицы** задают размер относительно значения другого свойства, определяющего размер. Документы, в которых таблицы стилей используют относительные единицы измерения, более приспособлены для отображения на разных устройствах (например, дисплей или лазерный принтер). **Абсолютные единицы** измерения полезны только тогда, когда известны физические характеристики устройства отображения [13].

Относительные	***Абсолютные***
em - высота шрифта элемента	in - дюйм (1 in = 2.54 cm)
ex - высота буквы x	cm - сантиметр
px - пиксель	mm - миллиметр
% - процент	pt - пункт (1 pt = 1/72 in)
	pc - пика (1 pc = 12 pt)

Примечание. Относительные единицы измерения em и ex во всех свойствах вычисляются относительно высоты шрифта элемента. Единственное исключение - свойство font-size, в котором эти единицы относятся к высоте шрифта элемента-родителя.

2.7. Работа со слоями в CSS

Стили CSS предоставляют возможность работы со слоями: фрагментами HTML, которые можно размещать на web-странице путем наложения их друг на друга с точностью до пикселя.

Для создания слоев используются элементы ** ** или ****. Эти элементы взаимозаменяемы и различаются лишь внешним видом в браузере. Для создания элемента web-страницы с отступами до и после текста, используется элемент . Для размещения текста внутри абзаца используется элемент .

Позиционирование элемента определяется указанием значений свойства *position,* положение элемента указанием координат с помощью свойств *top* и *left*, положение слоя свойством *z-index*, возможность отображения слоя с помощью свойства *visibility*.

Свойство *position* определяет тип системы координат. Возможные значения: *static* (статическое), *absolute* (абсолютное) и *relative* (относительное). Параметр *static* по умолчанию не оказывает никакого влияния на расположение слоев. При абсолютном позиционировании слой размещается относительно левого верхнего угла окна документа. В случае размещения слоя внутри другого, абсолютные координаты считаются от левого верхнего угла родительского слоя. Относительное позиционирование применяется для смещения слоя относительно родительского элемента. Установка этого значения не изменяет размещение элемента, но если установлены значения свойств *top* или *left*, то слой смещается от своего нормального положения в документе.

Синтаксис: `<div style="position:relative">`

С помощью свойств *top* и *left* определяется точное размещение слоя. Значения этих свойств могут принимать значение в пикселях или в процентах, могут быть положительными и отрицательными. Это дает возможность размещать содержимое выше или ниже на странице независимо от физической позиции кода HTML.

Синтаксис: `<div style="position:relative; top:-55; left:5;>`

!! *Создайте в программе* ***Блокнот*** *новый документ и сохраните его в своей папке под именем* ***Layers.html.*** *Создайте стилевое правило во внедренной в документ таблице стилей для класса* ***.lay1****:*

```
{
position:absolute;
left:350px;
top:50px;
width:400px;
height:48px;
```

```
color:red;
text-align:center;
}
```

!! *Создайте в документе новый слой с заголовком первого уровня с текстом "Заголовок страницы". Примените к нему созданный класс .lay1.*

!! *Создайте в таблице стилей стилевое для класса* ***.lay2****:*

```
{
position:absolute;
left:352px;
top:50px;
width:400px;
height:48px;
color:gray;
text-align:center;
}
```

!! *Создайте в документе еще один слой с заголовком первого уровня с текстом "Заголовок страницы". Примените к нему созданный класс .lay2.*

Свойство *z-index* определяет порядок слоев, или их расположение по отношению к другим слоям. По умолчанию все слои позиционированы со значением *z-index* равным нулю. Другие слои могут размещаться ниже путем установки отрицательного значения *z-index*. Для слоев, у которых *z-index* не установлен, это значение назначается неявно в соответствии с их положением в документе, то есть слой, который помещен в документ позже, размещается выше остальных элементов, позиционированных ранее.

Синтаксис: `<div style="position:relative; z-index:2; color: navy">`

!! *Установите порядок расположение слоев, изменив стилевые правила классов, так, чтобы серый текст располагался под красным. В результате получается текст с тенью. Проверьте результат в браузере.*

Для отображения или скрытия слоя используется параметр **visibility**. Он может принимать значения ***visible***, установленное по умолчанию, для показа слоя, и ***hidden*** для скрытия.

Синтаксис: `<div style="visibility: hidden">`

!! *Скройте один из слоев в документе. Проверьте результат в браузере. Верните его отображение.*

Контрольные вопросы

1. Что такое стили? Для чего они используются при разработке web-страниц?
2. Поясните общий синтаксис записи стилей.
3. Что такое таблица стилей?
4. Какие способы присоединения таблиц стилей к документу вы знаете?
5. Какие специальные селекторы CSS используются при создании таблиц стилей?
6. Что такое классы и псевдоклассы ссылок? Поясните их синтаксис и применение.
7. В чем различие дочернего и соседнего селекторов?
8. Что такое каскады таблиц стилей?
9. Какие приоритеты между стилями существуют?
10. Для чего используется ключевое слово !important ?

3. Основы JavaScript

3.1. Назначение и особенности языка JavaScript

JavaScript – это упрощенный объектно-ориентированный[1] язык программирования, позволяющий добавлять web-страницам интерактивность. Язык JavaScript позволяет разрабатывать клиентские web-приложения. Программы, написанные на языке JavaScript называются сценариями или скриптами (script), включаются в состав HTML-

[1] **Объект** - совокупность *свойств* (структур данных, характерных для данного объекта), *методов* их обработки (подпрограмм изменения их свойств) и *событий,* на которые данный объект может реагировать и, которые приводят, как правило, к изменению свойств объекта.

документов и распространяются вместе с ними. Браузеры распознают встроенные в текст документа скрипты и выполняют их. JavaScript - интерпретируемый язык программирования, что обеспечивает безопасность написанных программ, так как в нем исключены все операции работы с файлами (чтение, запись и т.д.).

Примерами программ на JavaScript могут служить программы, проверяющие введенные пользователем данные или выполняющие какие-то действия при открытии или закрытии документа. Такие программы могут реагировать на действия пользователя - нажатие кнопок "мыши", ввод данных в экранной форме или перемещение "мыши" по странице. Более того, JavaScript-программы могут управлять самим браузером и атрибутами документа.

Для создания программ на JavaScript не требуется никаких дополнительных средств - необходим лишь браузер, поддерживающий язык JavaScript соответствующей версии и текстовый редактор, позволяющий создавать HTML-документы. Так как программа на JavaScript встраивается непосредственно в текст HTML-документа, можно немедленно увидеть результаты ее работы во время просмотра документа браузером и при необходимости внести изменения.

JavaScript – интерпретируемый язык. Текст программы интерпретируется, то есть анализируется и сразу же исполняется.

JavaScript предназначен для:

- обработки событий, происходящих на странице (придания странице интерактивности);
- изменения содержимого web-страниц во время их просмотра, анализ данных web-страниц (динамическое создание содержимого);
- формирование запросов к серверу и получение ответов без перегрузки web-страницы;
- работа с cookies (данные web-страницы, хранящиеся в браузере на компьютере пользователя).

Реализация этих функций не возможна средствами HTML и CSS, поэтому JavaScript дополняет эти инструменты web-программирования, а не заменяет или дублирует их.

3.2. Синтаксис языка JavaScript

Синтаксис языка JavaScript очень близок языку C++. Текст программы на JavaScript состоит из последовательности операторов. Один оператор в JavaScript может быть разбит на несколько строк, или, наоборот, в одной строке может быть несколько операторов. В программе отдельные операторы записываются через точку с запятой (допускается не ставить **;** если оператор является в строке последним).

JavaScript не имеет жестких требований к форматированию текста программы, таким образом, возможно использование символов перевода строки и отступов для придания тексту программы лучшей читабельности. Однострочные комментарии в JavaScript записываются после символов //, многострочные - между символами /* */.

Допустимые синтаксисом языка JavaScript операции представлены в табл. 1.

Типы данных

Переменные JavaScript могут иметь один из следующих типов:

- числовой (целые числа или числа с плавающей точкой);
- логический;
- строковый;
- нулевой тип (определяется ключевым словом null).

Тип данных при объявлении переменной не указывается. Тип присваивается переменной только тогда, когда ей присваивается какое-либо значение. Переменные в программе объявляются при помощи ключевого слова **var** (допускается опускать), например:

```
var x=1; y="string";
```

Основные арифметические и логические операции и применяемые операторы описаны в **Ошибка! Источник ссылки не найден.**2.

Таблица 2

Арифметические и логические операции

Арифметические операторы	
+,-,*,/,%	сложение, вычитание, умножение, деления, остаток от деления
++, --	приращение на единицу, уменьшение на единицу
-	изменение знака
Операторы присваивания	
=	присваивание значения
+=, -+, *=, /=, %=	увеличение, уменьшение, умножение, деление на заданную величину, взятие модуля заданной величины
Операторы сравнения	
= =	эквивалентность сравниваемых объектов
! =	не равно
>, >=, <, <=	больше, больше или равно, меньше, меньше или равно
Логические операторы	
&&, || , !	логическое И, ИЛИ, НЕ

Тип массив введен в JavaScript для возможности манипулирования разными объектами: это список всех гипертекстовых ссылок страницы, список всех картинок на странице, список, список всех элементов формы и т.п. Пользователь может создать и свой собственный массив, используя конструктор Array(). Например:

```
new array = new Array(); new array5 = new Array(5);
colors = new Array ("red","white","blue")
```

Размерность массива может динамически изменяться. Можно сначала определить массив, а потом присвоить одному из его элементов значение. Как только это значение будет присвоено, изменится и размерность массива:

```
colors = new Array(); colors[5] = "red"
```

В данном случае массив будет состоять из 6 элементов, т.к. первым элементом массива считается элемент с индексом 0.

Основные свойства и методы массивов отражены в табл.3.

Таблица 3

Свойства и методы массива

Объект	Значение
length	число элементов массива
join	объединяет элементы массива в строку символов, в качестве аргумента в этом методе задается разделитель, например: `string = acolors.join("+")`
reverse	изменяет порядок элементов массива на обратный
sort	сортирует элементов массива в порядке возрастания

Управление потоком вычислений

Для управления потоком вычислений используются операторы подобные операторам языка C++. Синтаксис основных операторов представлен ниже.

Условный оператор

```
if(i>0)
  {выражение1}
else
  {выражение2}
```

Оператор выбора

```
switch (значение)
{ case label1:
  выражение1; break;
 case label2 :
  выражение2; break;
...
case labelN :
  выражениеN; break;
 default: выражение;}
```

Операторы цикла

Цикл с предусловием:

```
while(условие)
   {выражение}
```

```
for(i=0;i<n;i++)
   { выражение}
```

При работе с объектами:

```
for(i in obj)
    {выражение}
```

Цикл с постусловием:

```
do {выражение}
while (условие)
```

Операторы прерывания потока вычислений

Оператор **break** используется для прерывания выполнения цикла, либо оператора switch, например:

```
while(i==0)
 {if(j==3) break;}
```

Оператор **continue** используется для рестарта операторов цикла.

```
while(i==0)
 {if(j==3) continue;}
```

3.3. Размещение JavaScript на web-странице

Код сценария JavaScript может объявляться в HTML-коде следующими способами.

1. Непосредственное размещение в любой части HTML-кода при помощи тега **<script>**, например:

```
<script language="JavaScript">
  document.write("Генерируемый текст")
</script>
```

Примечание. В примере **document** – обращение к объекту, содержащему текущий html-документ, **write ()** – метод, позволяющий добавить текст в документ.

!! *Создайте в **Блокноте** новый HTML-документ и сохраните его в своей папке под именем **Primer5.html**. Введите вышеуказанный код в тело документа. Проверьте результат в браузере.*

2. Размещение в отдельном файле с объявлением в заголовке HTML-документа при помощи тега **<script>**, например:

```
<script language="JavaScript" src="script.js">
</script>
```

Содержимое файла script.js:

```
window.alert("Привет!")
```

Примечание. В примере **window** – объект, обращающийся к текущему активному окну браузера, **alert ()** – метод, позволяющий вызвать системное окно с сообщением.

В случае подключения внешнего сценария содержимое тега <body> не будет показано до тех пор, пока не будет выполнен сценарий из подключаемого файла.

!! *Создайте в **Блокноте** новый документ и сохраните его в своей папке под именем **script.js**, введите в него вышеуказанный код.*

!! *Добавьте в код страницы **Primer5.html** ссылку на созданный файл со сценарием (см. пример выше). Просмотрите результат в браузере (пример на Рис. 18).*

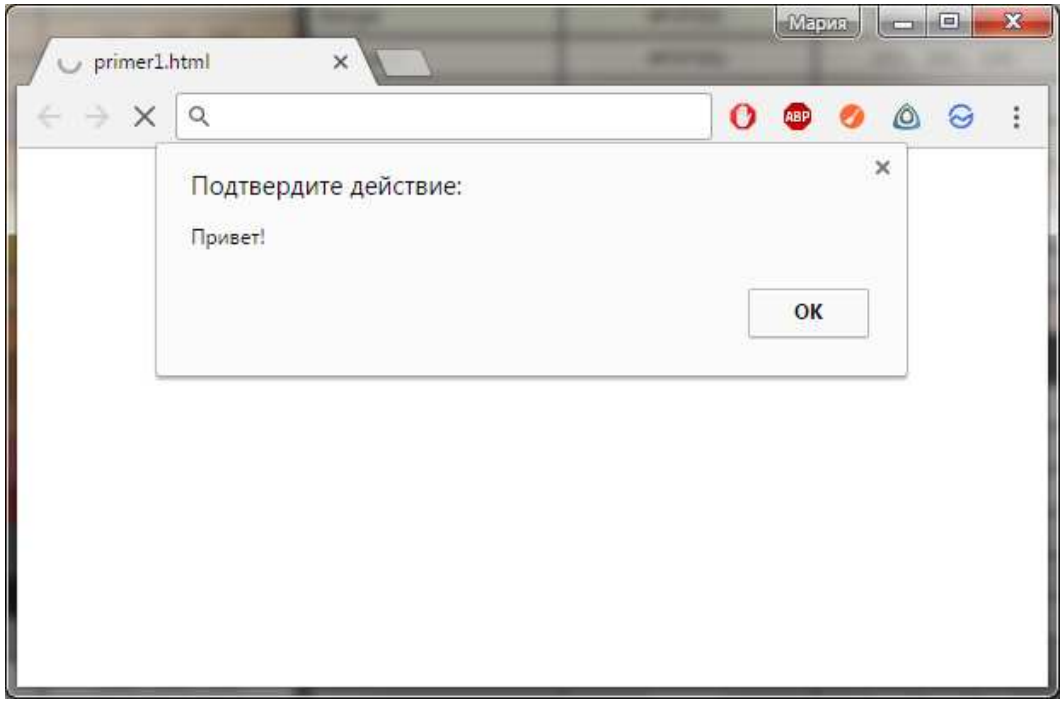

Рис. 18. Вызов окна с сообщением в объекте window.

3. Контекстное размещение в обработчике событий. Например:

```
<input type="button" value="Нажми меня" onClick="alert('Кнопка нажата!')">
```

Примечание. В примере **onClick** – событие при щелчке кнопки мыши на объекте, **alert ()** – метод, позволяющий вызвать системное окно с сообщением.

!! *Добавьте тело документа вышеуказанный код. Просмотрите результат в браузере.*

JavaScript - это объектно-ориентированный язык программирования (ООП), основанный не на обработке команд кода, а на присвоении отдельным элементам программы конкретных событий и выполнении их, если данное событие имело место. Основное понятие ООП - **объект** - совокупность ***свойств***, ***методов*** их обработки и ***событий***, на которые данный объект может реагировать.

Объект (object) - это то, с чем производится действие, событие. Это может быть документ, открываемый в окне браузера или само окно браузера, или какая-то часть документа, теги. Объект должен иметь уникальное имя (ID), чтобы к нему можно было обратиться. Каждый объект обладает своими методами.

Метод объекта (method) - это действия, которые можно выполнять над объектом такого типа, или которые сам объект может выполнять. Между именем объекта и методом обязательно ставят разделительный оператор точка, после метода в скобках параметры метода. Параметры метода относятся к типу данных - строки символов. Строки символов нужно обязательно взять в кавычки либо в одинарные, либо в двойные.

Синтаксис кода:

```
Объект.Метод("параметры метода")
```

!! *Добавьте в файл* ***script.js*** *метод* **confirm()** *для объекта window с параметром* "Продолжить работу?". *Метод* **alert()** *можно удалить или скрыть, превратив в комментарий. Просмотрите результат в браузере.*

Каждый объект обладает своими **свойствами** (properties). Один и тот же объект может обладать многими свойствами. Часто эти свойства

необходимо изменить, при возникновении некоторого события. Синтаксис кода:

```
Объект.свойство объекта="новое значение свойства"
```

!! *Добавьте в сценарий в теле документа изменение свойства* **BgColor** *(фон страницы) объекта document на* 'LightGreen' . *Просмотрите результат в браузере.*

Изменение свойства может происходить при возникновении какого-либо события. **Событие** (event) - это все, что случилось: открытие окна, загрузка в него документа, клик клавишей мышки или просто перемещение курсора по экрану, нажатие клавиши на клавиатуре - это все события, и они могут инициировать запуск больших и маленьких программ (табл. 4).

Синтаксис кода:

```
<тег событие="объект.свойство='значение свойства'">
<тег событие="объект.метод('параметры метода')">
```

Таблица 4

Стандартные события в HTML

Событие	Происходит
onClick	при щелчке кнопки мыши на элементе
onDblClick	при двойном щелчке кнопки мыши на элементе
onMouseDown	при нажатии кнопки мыши на элементе
onMouseUp	при отпускании кнопки мыши на элементе
onMouseOver	при попадании курсора мыши на элемент
onMouseMove	при движении курсора мыши по элементу
onMouseOut	при попадании курсора мыши за пределы элемента
onKeyPress	при нажатии и отпускании клавиши на элементе
onKeyDown	при нажатии клавиши на элементе
onKeyUp	при отпускании клавиши на элементе

!! *Добавьте созданной в документе кнопке событие* **onMouseOver** *с методом* alert('Кнопка в фокусе!'). *Просмотрите результат в браузере.*

3.4. Объектная модель браузера и документа

Совокупность приемов, позволяющих динамически изменять оформление и содержание web-страницы в ответ на действия пользо-

вателя называется **динамический HTML (DHTML).** DHTML является продуктом взаимодействия трех технологий: языка HTML, каскадных таблиц стилей (CSS) и языка сценариев (JavaScript).

Язык JavaScript позволяет работать с загруженным html-документом, отображаемым в окне браузера посредством DOM (Document Object Model - объектная модель документа), с браузером - посредством BOM (Browser Object Model - объектная модель браузера). Кроме того, есть объекты, поддерживаемые непосредственно JavaScript.

Объектная модель браузера представляет собой строгую иерархическую структуру, позволяющую обращаться к любой части браузера или загруженных страниц с помощью языка JavaScript. Обобщенная объектная модель представлена на Рис. 19. Объектная модель браузера. Рис. 19.

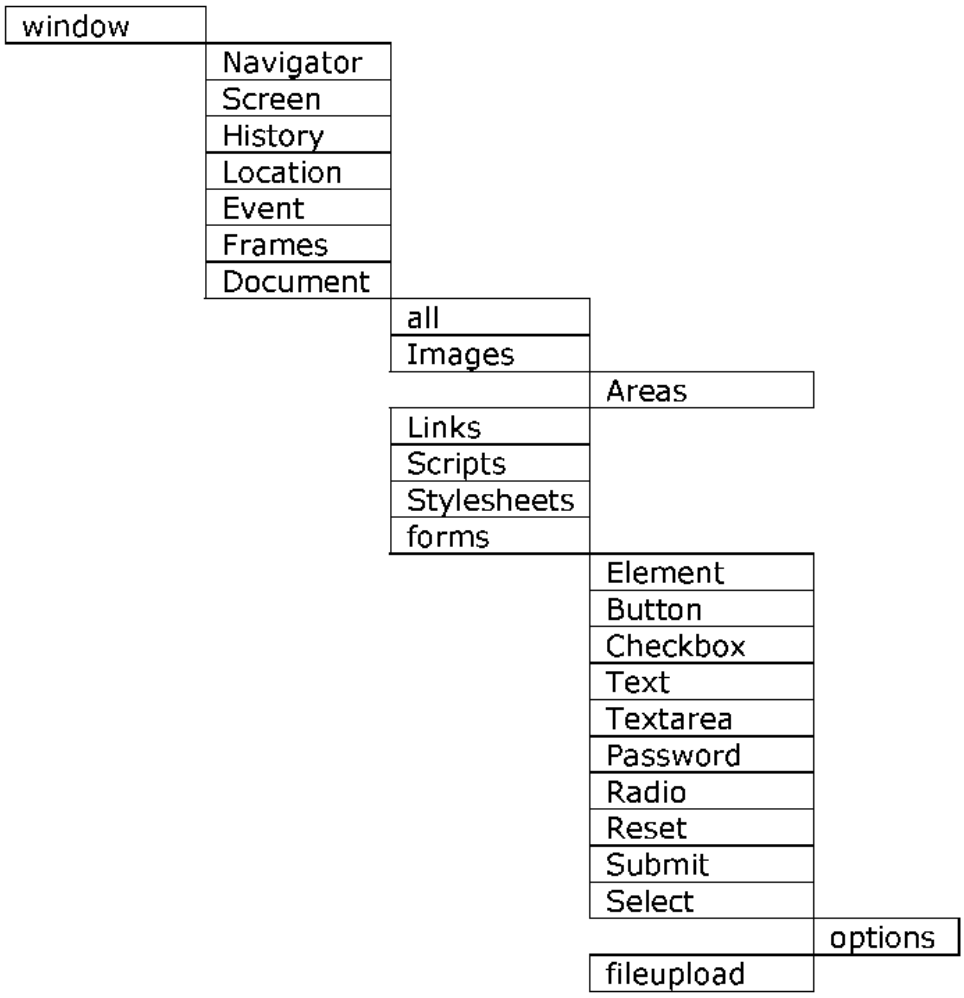

Рис. 19. Объектная модель браузера.

В качестве примера на Рис. 20 представлена HTML-страница с указанием соответствия каждого ее элемента объектной модели.

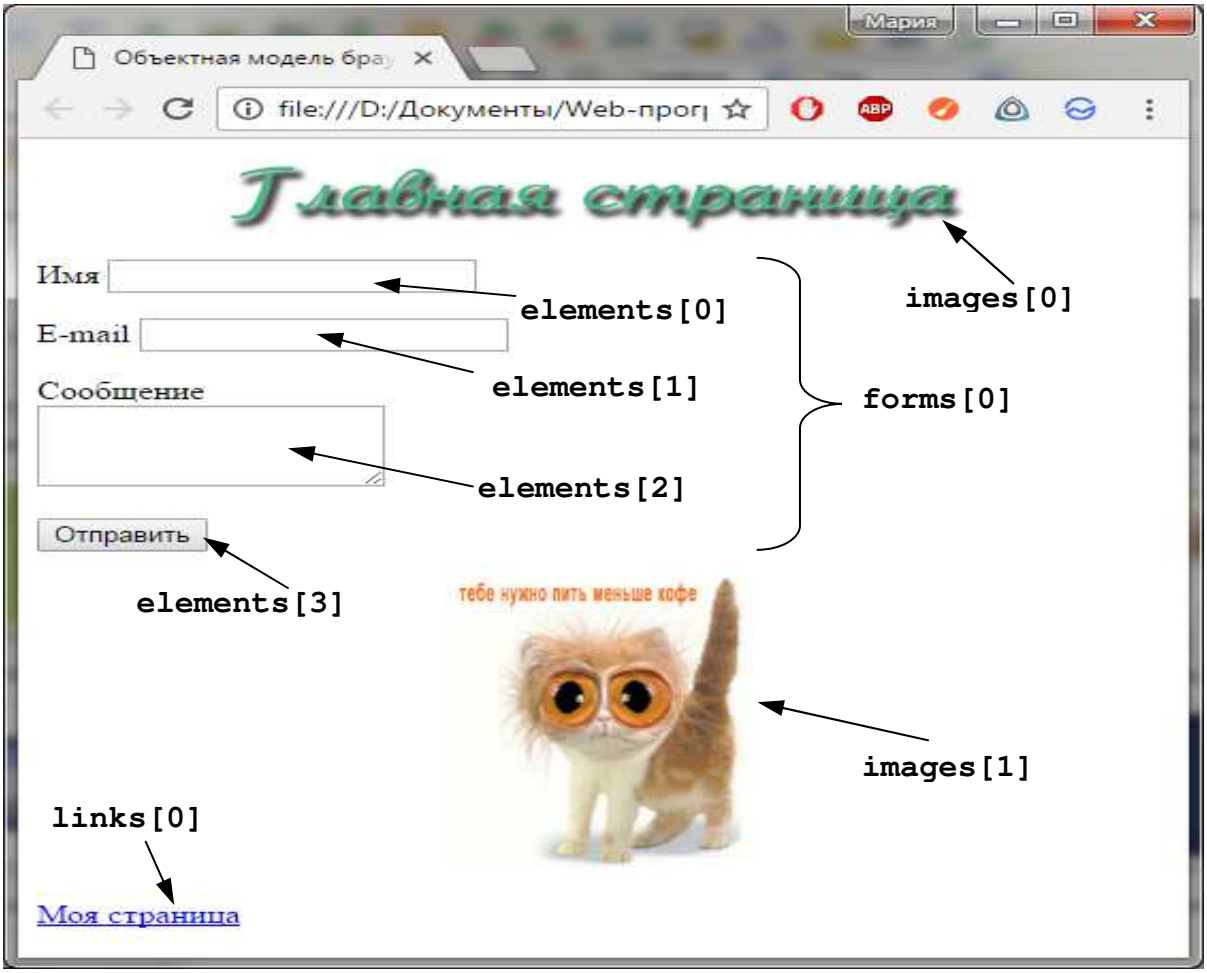

Рис. 20. Соответствие HTML-элементов объектной модели браузера.

Используя синтаксис JavaScript можно обратиться к любому элементу HTML-документа и изменить его поведение при помощи свойств и методов этого элемента, например:

```
document.forms[0].elements[0].value = "Имя";
```

Если web-страница достаточно объемная по содержанию и структуре, то процедура адресации к различным объектам по номеру может стать весьма неудобной. Во избежание подобной проблемы можно присваивать различным тегам уникальные имена, например:

```
<form name="myForm">
Имя: <input type="text" name="name" value=""><br>
```

```
   E-mail: <input type="text" name="email" val-
ue=""><br>
   Сообщение: <textarea name="msg" rows="3" val-
ue=""><br>
   <input type="submit" name="submit"
value="Отправить">
   </form>
```

В этом случае обращение к элементам HTML-документа выглядит следующим образом:

```
document.myForm.name.value = "Имя";
```

3.5. Объект window

Объект **window** - главный объект в иерархии объектной модели браузера. Объект window обращается к активному окну.

Перечень основных свойств, методов и событий объекта window представлены в таблицах 5, 6 и 7.

Таблица 5

Свойства объекта window

Свойство	Значение
parent	Возвращает родительское окно
name	Название окна
status	Текст, отображаемый в строке состояния
document	Ссылка «только для чтения» на объект окна document
event	Ссылка «только для чтения» на объект окна event
history	Ссылка «только для чтения» на объект окна history
location	Ссылка «только для чтения» на объект окна location
navigator	Ссылка «только для чтения» на объект окна location
screen	Ссылка «только для чтения» на объект окна screen

Таблица 6

События объекта window

Событие	Значение
onBlur	Потеря фокуса
onFocus	Окно в фокусе
onHelp	Нажатие F1 или Help
onResize	Изменение размеров окна
onScroll	Прокрутка окна
onLoad	Страница полностью загружена
onUnload	Перед выгрузкой страницы

Таблица 7

Методы объекта window

Метод	Значение
alert	Вызов системного окна с простым сообщением. Например, `alert("Заполните пустые поля формы")`
Confirm(text)	Вызов системного окна с кнопками OK/CANCEL. Например, `confirm("Продолжить поиск?")`
Prompt(text)	Вызов системного окна с полем для ввода текста с клавиатуры. Например, `prompt ("Введите Ваше имя")`
Open(url, name, settings)	Открывает новое окно. Например, `window.open("help.htm", "helpWindow", "width=400,height=300,status=no,toolbar=no, menubar=no")`
Close()	

Дочерние объекты window

Объект window имеет несколько дочерних объектов, которые доступны с его помощью:

- history содержит информацию о посещенных адресах.
- navigator содержит информацию о браузере.
- location содержит информацию о URL текущей страницы.
- event содержит информацию о событиях, происходящих в браузере.
- screen содержит информацию о возможностях экрана клиента.
- document содержит отображаемый документ.

Перечень основных свойств, методов и событий дочерних объектов window представлены в табл. 8.

Таблица 8

Свойства, методы и события дочерних объектов window

Объект	Значение
History	
length	Длина списка посещенных адресов
back()	Загружает предыдущий адрес
forward()	Загружает следующий адрес
go(n)	Загружает адрес с номером n
Navigator	
appName	Название браузера
appVersion	Версия браузера
cookieEnabled	Возможность работы с cookie
mimeTypes	Список поддерживаемых браузером типов файлов
plugins	Список поддерживаемых браузером внедряемых объектов
javaEnabled()	Возможность запуска JavaScript
Location	
href	Указывет URL страницы
reload()	Обновляет страницу
Event	
X	Горизонтальная позиция курсора мыши
Y	Вертикальная позиция курсора мыши
button	Кнопка мыши, если она нажата
altkey	Состояние клавиши Alt
Ctrlkey	Состояние клавиши Ctrl
Shiftkey	Состояние клавиши Shift
Keycode	ASCII код нажатой клавиши
Screen	
Width	Разрешение по ширине экрана
Height	Разрешение по высоте экрана
colorDepth	Глубина цвета палитры экрана

Объектная модель документа (DOM) строится в полном соответствии со структурой html-документа (Рис. 21). В соответствии с объектной моделью документа (далее - DOM), html- документ представляется в виде дерева, состоящего, как правило, из узлов двух ти-

пов: узел-элемент и узел-текст. Каждому тегу html-документа в дереве DOM соответствует узел-элемент, вложенные теги становятся дочерними узлами (на рисунке узлы-элементы представлены прямоугольниками). Текстовое содержимое тегов соответствует узлам-текстам (на рисунке узлы-тексты представлены овалами).

```
<html>
 <head>
  <title>
 Название html-документа
  </title>
 Заголовок html-документа
 </head>
     <body>
     Тело html-документа
     </body>
</html>
```

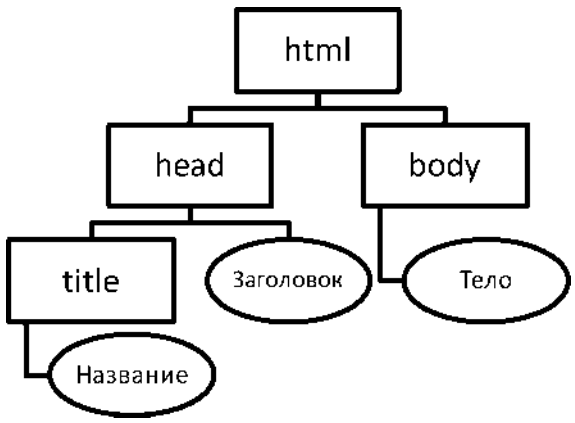

Рис. 21. Структура и DOM html-документа.

Всего стандартом определено 12 типов узлов, но узел-элемент и узел-текст используются чаще других.

Перечень свойств, методов и событий дочерних объектов document представлены в табл. 9,10 и 11.

Таблица 9

Свойства объекта document

Объект	Значение
Title	Название документа, определяемое в теге <title>
Bgcolor	Цвет фона
Fgcolor	Цвет текста
linkColor	Цвет ссылок
alinkColor	Цвет активных ссылок
vlinkColor	Цвет посещенных ссылок
activeElement	Элемент в фокусе

Таблица 10

Методы объекта document

Объект	Значение
open()	Открывает объект document для редактирования
write(text)	Запись текста в объект document
close()	Закрывает объект document для редактирования
clear	Очищает содержимое объекта document
alinkColor	Цвет активных ссылок
vlinkColor	Цвет посещенных ссылок
activeElement	Элемент в фокусе

Таблица 11

События объекта document

Объект	Значение
onClick	Щелчок мыши
onDblClick	Двойной щелчок мыши
onMouseOut	Мышь уходит с элемента
onMouseOver	Мышь появляется над элементом
onKeyDown	Нажатие клавиши
onHelp	Нажатие F1 или Help
onLoad	Полная загрузка элемента

!! *Измените значение свойства* **title** *(название страницы) дочернего объекта window.document на* 'Примеры ***JavaScript*** *' в файле* ***script.js****. Просмотрите результат в браузере. Обратите внимание на произошедшие изменения.*

3.6. Работа с функциями

Если сценарий состоит из большого количества строк кода, удобнее использовать для редактирования кода страницы собственные функции. **Функция** – это именованная совокупность сгруппированных инструкций. Функцию можно создать один раз, а вызывать многократно. Функции JavaScript похожи на функции C/C++ за тем исключением, что не нужно объявлять тип входных переменных и тип результата функции. Браузер, который выполняет сценарий, по присваиваемым значениям сам поймет, что имеется ввиду.

Функции должны быть определены перед вызовом, и размещение всех определений функций в разделе заголовка<head> документа гарантирует доступность этих функций при обработке документа.

Определение функции:

```
function имя([список параметров])
  {
   тело функции
   [return значение]
  }
```

С помощью ключевого слова **return** функция может вернуть значение. Функция может не иметь параметров, в этом случае она не возвращает никакого значения, а просто выполняет какое-либо действие.

Листинг 15. Пример функции вывода окна сообщения.

```
<script type="text/javascript">
function showMsg()
{
alert('Вызвана функция');
}
</script>
```

!! *Создайте новый HTML-документ и сохраните его в своей папке под именем* ***Primer6.html****. Создайте в заголовке документа функцию из Листинга 15.*

Вызов функции осуществляется из любого места документа, где имеется код JavaScript. Для вызова функции достаточно указать ее имя и параметры. Если функция не имеет параметров, при ее вызове необходимо оставить пустые скобки после имени функции, иначе браузер будет пытаться использовать ее как переменную.

Пример вызова функции:

```
<script type="text/javascript">
showMsg();
</script>
```

!! *В теле документа вызовите созданную ранее функцию. Просмотрите результат в браузере.*

!! *Отредактируйте текст созданной функции следующим образом:*

```
<script type="text/javascript">
```

```
    function showMsg(msg)
    {
    //Получение текущей даты
    var data=new Date();

    //Создание текстового сообщения
    var fullMsg = "Важное сообщение!\n\n";
    fullMsg+=msg + "\n\n";
fullMsg+="Сообщение сгенерировано:" + data.toDateString();

    //Вывод сообщения
    alert(fullMsg);
    }
    </script>
```

!! *При вызове функции добавьте параметр* "Эта страница содержит функцию JavaScript*". Просмотрите результат в браузере (пример на Рис. 22).*

Примечания. 1. Для вставки разрыва строки используется код **\n**.

2. Знак **+** при работе с текстовыми данными является оператором конкатенации – объединения нескольких текстовых строк в одну.

3. Функция `toDateString` объекта `data` преобразует информацию о дате, полученную от операционной системы, в текстовую строку.

3.7. Обращение к объектам и свойствам таблицы стилей

Для изменения содержимого web-страницы используются свойства и методы объекта документ и его дочерних объектов - элементов web-страницы в соответствии с объектной моделью документа (DOM). Для доступа к любому элементу документа используются различные функции. Например, функция (метод) **getElementById()** позволяет получить доступ к элементу по идентификатору ID. Получив доступ к элементу можно изменять его значение и значение свойств стилей.

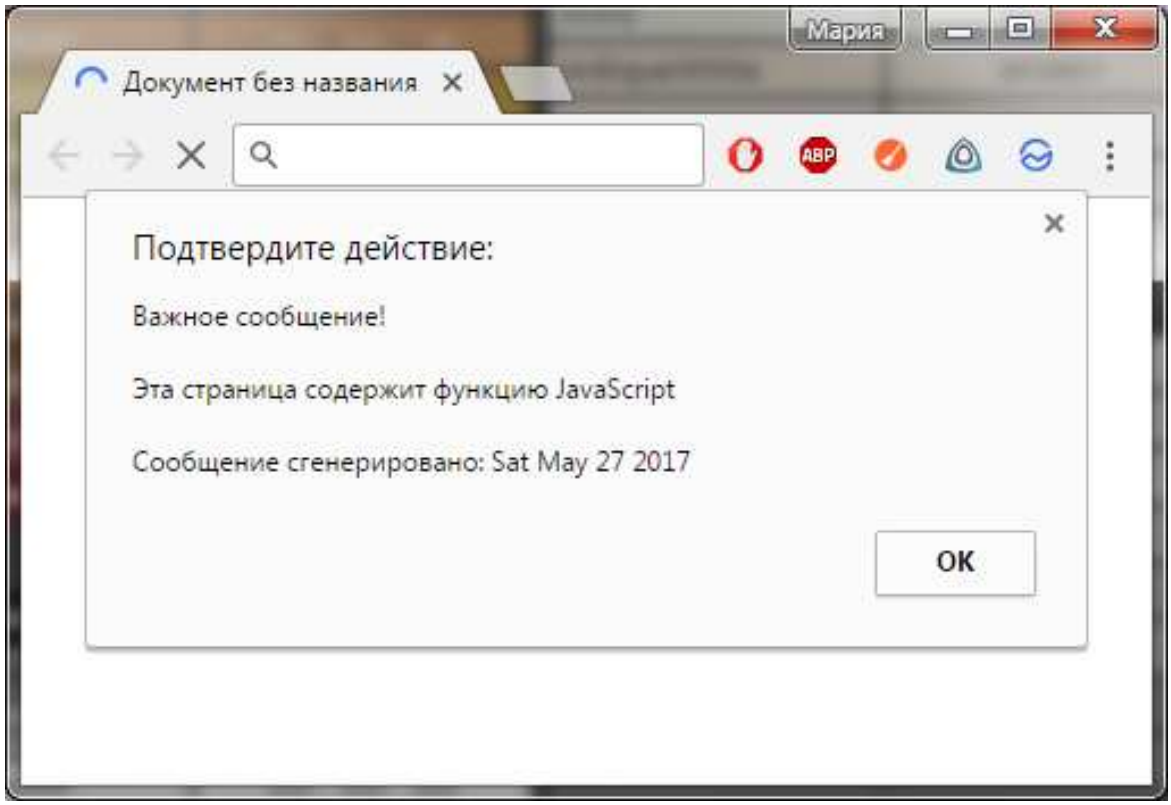

Рис. 22. Результат работы функции JavaScript.

Свойства управления содержимым объекта:

`innerText` – доступ к содержимому узла-текста (текст, содержащийся в элементе);

`outerText` – доступ ко всему узлу-тексту;

`innerHTML` – доступ к содержимому узла-элемента (текст и код элемента);

`outerHTML` - доступ ко всему узлу-элементу (весь текст и код элемента).

Например, имеется заголовок третьего уровня:

```
<h3 id="heading1">Старый заголовок<h3>
```

В сценарии JavaScript заменяется текст заголовка:

```
objHead1=document.getElementById("heading1");
objHead1.innerText="Новый заголовок"
```

!! *Создайте новый HTML-документ и сохраните его в своей папке под именем* ***Primer7.html****. Создайте в документе заголовок треть-*

его уровня и сценарий, изменяющий его содержимое (см. пример). Просмотрите результат в браузере.

Методы управления содержимым объекта:

`insertAdjacentText(положение, текст)` - вставка текста;

`insertAdjacentHTML(положение, текст)` - вставка текста и HTML.

Варианты положения: BeforeBegin, AfterBegin, BeforeEnd, AfterEnd.

Например:

```
objHead1.insertAdjacentText("AfterBegin","Привет")
```

!! *Добавьте в сценарий строку, добавляющую в заголовок слово* Привет *(см. пример). Просмотрите результат в браузере.*

!! *Добавьте в текст документа текст (абзац)* Пример изменения текста *и слой с помощью элемента , создав для него идентификатор (ID) lay1. Для идентификатора создайте стилевое правило во внедренной в документ таблице стилей:*

```
#lay1 {
        background-color:yellow;
        padding: 15px;
        }
```

В содержимое слоя добавьте текст Желтый прямоугольник.

!! *Создайте сценарий с функцией следующего содержания:*

```
function primer()
{
 layer=document.getElementById("lay1");
 layer.style.backgroundColor="red";
 layer.innerHTML="Красный прямоугольник";
 }
```

!! *Добавьте к элементу* ***lay1*** *вызов функции* **primer()** *при событии* OnMouseOver. *Просмотрите результат в браузере.*

Контрольные вопросы

1. Поясните назначение языка JavaScript?
2. Чем характеризуется объект JavaScript?
3. Что такое метод, свойство и событие объекта?
4. Опишите основные особенности синтаксиса языка JavaScript.
5. Как можно разместить сценарии JavaScript на web-странице?
6. Что такое объектная модель браузера? Какой объект является самым главным?
7. Приведите примеры основных методов объекта window.
8. Что такое объектная модель документа? Чем отличаются узел-текст и узел-элемент?
9. Как создается функция JavaScript?
10. Каким образом можно обратиться к объектам документа?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Будин В.И., Поднебесова М.И. Основы информатики и информационных технологий: учеб. пособие/В.И. Будин, М.И. Поднебесова. – Самара: Самар. гос. техн. ун-т, 2014. – 172 с.
2. Васильев В.В. Практикум по Web-технологиям :учеб. пособие для вузов/ В.В.Васильев, Н. В. Сороколетова, Л.В. Хливненко.- М.: Форум, 2009.- 416с.:ил.- (Высшее образование)
3. Дуванов, А.А. Web-конструирование: элективный курс . - СПб.:БХВ-Петербург, 2006. - 432с.:ил
4. Использование HTML 4: Пер. с англ. – 3-е изд./ Л. Паттерсон, С. Шарльворс, Дж. Корнелиус и др. – К.; М.; СПб.: Издат. дом "Вильямс", 1998. – 384 с.: ил.
5. Лыткина Е.А. Основы языка HTML: учеб. пособие /Е.А. Лыткина, А.Г. Глотова. - Архангельск: Сев.(Арктич.) федер. ун-т им. М.В. Ломоносова, 2014. – 112 с.
6. МакДональд, М. Создание Web-сайтов. Основное руководство/ Мэтью МакДональд; [пер. с англ. и ред. М.А. Райтмана]. – М.: Эксмо, 2010. -768 с.
7. Основы web-технологий/ П.Б. Храмцов, С.А. Брик, А.М. Русак, А.И. Сурин/ Под редакц.П.Б. Храмцова. – М.: ИНТУИТ.РУ "Интернет-Университет Информационных Технологий", 2003. – 512 с.
8. Смирнова И.Е. Начала Web-дизайна.- СПб.: БХВ-Петербург, 2003.- 256с.
9. Спецификация CSS 2.1. URL: http://specs.operafan.net/css2.1RU/CSS21/cover.html#minitoc.
10. Спецификация HTML 5.1. URL: https://www.w3.org/TR/2016/WD-html51-20160310.
11. Сычев А.В. Web-технологии [Электронный ресурс]/ Сычев А.В. Электрон. текстовые данные. М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016.- 184 с. URL: http://www.iprbookshop.ru/56344
12. Htmlbook.ru. Сайт Влада Мержевича. URL: http://htmlbook.ru/ .
13. HTML 4.0 Учебник. URL: http://mysite.e-stile.ru/html4/.

СОДЕРЖАНИЕ

Учебное пособие

ПОДНЕБЕСОВА Мария Игоревна

Web-программирование

Редакторы:
Е.С. Захарова
И. А. Назарова

Подписано в печать 25.05.2017г.
Формат 60x84 1/16. Бумага офсетная
Усл. п. л. 5,6 Уч.-изд. л. 4,2
Тираж 50 экз. Рег. № 8/17sf

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
"Самарский государственный технический университет"
443100, г. Самара, ул. Молодогвардейская, 244. Главный корпус

Отпечатано в типографии
Самарского государственного технического университета
Филиал в г. Сызрани, 446001, г. Сызрань, ул. Советская 45