МИНИСТЕРСТВО ВЫСШЕГО ОБРАЗОВАНИЯ, НАУКИ И ИННОВАЦИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

ТАШКЕНТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ ИСЛАМА КАРИМОВА

ОСНОВЫ ВЕБ-ПРОГРАММИРОВАНИЯ

МЕТОДИЧЕСКОЕ ПОСОБИЕ

Ташкент - 2023

УДК 621.314

Методическое пособие к выполнению лабораторных работ по предмету «Основы веб-программирования». Хамракулов У.Ш., Шоймардонов Т.Т., Ашуралиев А.А., Омонова М.Ш. – Ташкент.: ТашГТУ, 2023. -87 с.

В методическом пособии рассмотрены основные разработки веб-технологии, описаны общие понятия о языке разметки гипертекстовых документов HTML, основы языка JavaScript и технологии CSS (Cascading Style Sheets).

Данное методическое пособие предназначено для студентов высших учебных заведений, обучающихся по направлению образования 5330200 – Информационные системы и технологии (по отраслям и сферам).

Печатается по решению научно-методического совета Ташкентского государственного технического университета. Протокол №6 от 30 марта 2023г.

Рецензенты:
ст. преп. Эргашев О.М. (ТУИТ)
проф.,д.т.н. Сагатов М.В. (ТашГТУ)

ВВЕДЕНИЕ

В нашей стране принимаются комплексные меры по активному развитию цифровой экономики, широкому внедрению современных информационно-коммуникационных технологий во все отрасли, особенно в образование. В частности, реализация проекта стратегии «Цифровой Узбекистан – 2030», предусматривающей совершенствование электронного правительства, дальнейшее развитие местного рынка программных продуктов и информационных технологий, создание IT-парков во всех регионах страны, а также обеспечение отрасли квалифицированными кадрами.

В настоящее время все усилия направлены на обучение молодежи разработке программного обеспечения и информационных технологий, разработке и внедрению аппаратного и программного обеспечения, робототехнике, экспорту информационных услуг через Интернет, а также хранению и обработке данных. Одним из важнейших вопросов является обеспечение учебниками, способными в совершенстве научить практически применять теоретические знания по каждому предмету.

Веб-программирование стало новой и быстро развивающейся областью интернет-технологий. Целью подготовки веб-страниц и их размещения в сети Интернет являются предоставление различной информации в качестве справочной информации, реклама продукции, распространение среди широкой публики литературных произведений, музыки и изображений.

Целью методического пособия по основам веб-программирования является обучение теоретическим основам и принципам практического программирования, их функциональной и структурной организации, методам создания динамических веб-страниц с использованием специальных языков программирования на примере разработки веб-страниц.

Основной задачей обучения основам веб-программирования является обучение студентов таким темам, как принцип работы интернета, основы веб-технологий, основы веб-дизайна, язык гипертекстовой разметки HTML, управление веб-страницей с помощью языка программирования JavaScript. и использование графики на веб-страницах, а также для облегчения получения практических результатов.

ЛАБОРАТОРНАЯ РАБОТА №1

Классификация веб-технологий

Цель работы: Приобрести навыки при работе с программами веб-программирования, их настройки и редактирования.

Задание: Установить и настроить инструменты веб программирования. Создать свою первую веб-страницу.

Методические указания:

HTML (HyperText Markup Language) представляет язык разметки гипертекста, используемый преимущественно для создания документов в сети интернет. HTML начал свой путь в начале 90-х годов как примитивный язык для создания веб-страниц, и в настоящий момент уже трудно представить себе интернет без HTML. Подавляющее большинство сайтов так или иначе используют HTML.

В 2014 году официально была завершена работа над новым стандартом - HTML5, который фактически произвел революцию, привнеся в HTML много нового.

Что именно привнес HTML5?

- HTML5 определяет новый алгоритм парсинга для создания структуры DOM
- добавление новых элементов и тегов, как например, элементы video, audio и ряд других
- переопределение правил и семантики уже существовавших элементов HTML

Фактически с добавлением новых функций HTML5 стал не просто новой версией языка разметки для создания веб-страниц, но и фактически платформой для создания приложений, а область его использования вышла далеко за пределы веб-среды интернет: HTML5 применяется также для создания мобильных приложений под Android, iOS, Windows Mobile и даже для создания десктопных приложений для обычных компьютеров (в частности, в ОС Windows 8/8.1/10).

В итоге, как правило, HTML 5 применяется преимущественно в двух значениях:

HTML 5 как обновленный язык разметки гипертекста, некоторое развитие предыдущей версии HTML 4

HTML 5 как мощная платформа для создания веб-приложений, которая включает не только непосредственно язык разметки

гипертекста, обновленный HTML, но и язык программирования JavaScript и каскадные таблицы стилей CSS 3.

Кто отвечает за развитие HTML5? Этим занимается World Wide Web Consortium (сокращенно W3C - Консорциум Всемирной Паутины) - независимая международная организация, которая определяет стандарт HTML5 в виде спецификаций. Текущую полную спецификацию на английском языке можно посмотреть по адресу https://www.w3.org/TR/html5/. И надо отметить, что организация продолжает работать над HTML5, выпуская обновления к спецификации.

Поддержка браузерами. Надо отметить, что между спецификацией HTML5 и использованием этой технологии в веб-браузерах всегда был разрыв. Большинство браузеров стало внедрять стандарты HTML5 еще до их официальной публикации. И к текущему моменту большинство последних версий браузеров поддерживают большинство функциональностей HTML5 (Google Chrome, Firefox, Opera, Internet Explorer 11, Microsoft Edge). В то же время многие старые браузеры, как например, Internet Explorer 8 и более младшие версии, не поддерживают стандарты, а IE 9, 10 поддерживает лишь частично.

При этом даже те браузеры, которые в целом поддерживают стандарты, могут не поддерживать какие-то отдельные функции. И это тоже надо учитывать в работе. Но в целом с поддержкой данной технологии довольно хорошая ситуация.

Для проверки поддержки HTML5 конкретным браузером можно использовать специальный сервис http://html5test.com.

Что потребуется для работы с HTML5? В первую очередь, текстовый редактор, чтобы набирать текст веб-страниц на html. На данный момент одним из самых простых и наиболее популярных текстовых редакторов является Notepad++, который можно найти по адресу http://notepad-plus-plus.org/. К его преимуществам можно отнести бесплатность, подсветка тегов html. В дальнейшем будем ориентироваться именно на этот текстовый редактор.

И также потребуется веб-браузер для запуска и проверки написанных веб-страничек. В качестве веб-браузера можно взять последнюю версию любого из распространенных браузеров - Google Chrome, Mozilla Firefox, Microsoft Edge, Opera.

Элементы и атрибуты HTML5.

Прежде чем переходить непосредственно к созданию своих веб-страниц на HTML5, рассмотрим основные строительные блоки, кирпичики, из которых состоит веб-страница.

Документ HTML5, как и любой документ HTML, состоит из элементов, а элементы состоят из тегов. Как правило, элементы имеют открывающий и закрывающий тег, которые заключаются в угловые скобки. Например:

Текст элемента div

Здесь определен элемент div, который имеет открывающий тег и закрывающий тег . Между этими тегами находится содержимое элемента div. В данном случае в качестве содержимого выступает простой текст "Текст элемента div".

Элементы также могут состоять из одного тега, например, элемент
, функция которого - перенос строки.

Текст
 элемента div

Такие элементы еще называют пустыми элементами (void elements). Хотя был использован закрывающий слеш, но его наличие согласно спецификации необязательно, и равнозначно использованию тега без слеша:

Каждый элемент внутри открывающего тега может иметь атрибуты. Например:

Кнопка

<input type="button" value="Нажать">

Здесь определено два элемента: div и input. Элемент div имеет атрибут style. После знака равно в кавычках пишется значение атрибута: style="color:red;". В данном случае значение "color:red;" указывает, что цвет текста будет красным.

Второй элемент - элемент input, состоящий из одного тега, имеет два атрибута: type (указывает на тип элемента - кнопка) и value (определяет текст кнопки).

Существуют глобальные или общие для всех элементов атрибуты, как, например, style, а есть специфические, применяемые к определенным элементам, как, например, type.

Кроме обычных атрибутов существуют еще булевые или логические атрибуты (boolean attributes). Подобные атрибуты могут не иметь значения. Например, у кнопки можно задать атрибут disabled:

<input type="button" value="Нажать" disabled>

Атрибут disabled указывает, что данный элемент отключен.

Глобальные атрибуты.

В HTML5 есть набор глобальных атрибутов, которые применимы к любому элементу HTML5:

accesskey: определяет клавишу для быстрого доступа к элементу

class: задает класс CSS, который будет применяться к элементу

contenteditable: определяет, можно ли редактировать содержимое элемента

contextmenu: определяет контекстное меню для элемента, которое отображается при нажатии на элемент правой кнопкой мыши

dir: устанавливает направление текста в элементе

draggable: определяет, можно ли перетаскивать элемент

dropzone: определяет, можно ли копировать переносимые данные при переносе на элемент

hidden: скрывает элемент

id: уникальный идентификатор элемента. На веб-странице элементы не должны иметь повторяющихся идентификаторов

lang: определяет язык элемента

spellcheck: указывает, будет ли для данного элемента использоваться проверка правописания

style: задает стиль элемента

tabindex: определяет порядок, в котором по элементам можно переключаться с помощью клавиши TAB

title: устанавливает дополнительное описание для элемента

translate: определяет, должно ли переводиться содержимое элемента

Но, как правило, из всего этого списка наиболее часто используются три: class, id и style.

Пользовательские атрибуты.

В отличие от предыдущей версии языка разметки в HTML5 были добавлены пользовательские атрибуты (custom attributes). Теперь разработчик или создатель веб-страницы сам может определить любой атрибут, предваряя его префиксом data-. Например:

<input type="button" value="Нажать" data-color="red" >

Здесь определен атрибут data-color, который имеет значение "red". Хотя для этого элемента, ни в целом в html не существует подобного атрибута. Мы его определяем сами и устанавливаем у него любое значение.

Одинарные или двойные кавычки.

Нередко можно встретить случаи, когда в html при определении значений атрибутов применяются как одинарные, так и двойные кавычки. Например:

<input type='button' value='Нажать'>

И одинарные, и двойные кавычки в данном случае допустимы, хотя чаще применяются именно двойные кавычки. Однако иногда само значение атрибута может содержать двойные кавычки, и в этом случае все значение лучше поместить в одинарные:

Создание документа HTML5

Элементы являются кирпичиками, из которых складывается документ html5. Для создания документа нам надо создать простой текстовый файл, а в качестве расширения файла указать *.html

Создадим текстовый файл, назовем его index и изменим его расширение на .html.

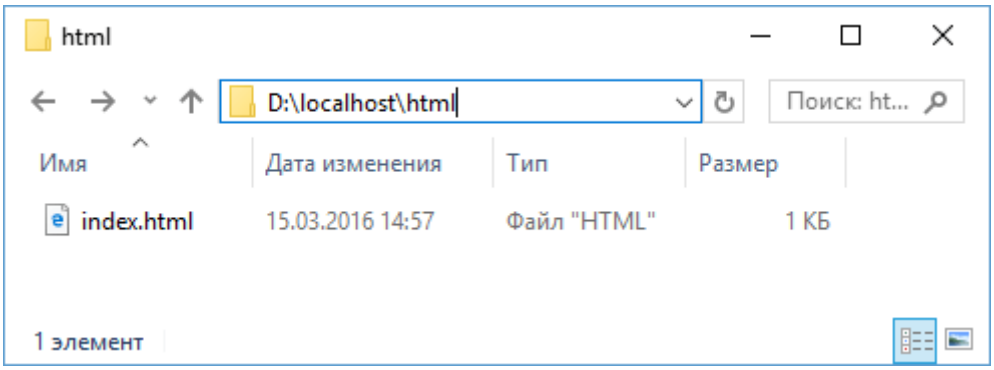

Рис. 1.1. Создание документа HTML.

Затем откроем этот файл в любом текстовом редакторе, например, в Notepad++. Добавим в файл следующий текст:

<!DOCTYPE html>
<html>
...
</html>

Для создания документа HTML5 нам нужны в первую очередь два элемента: DOCTYPE и html. Элемент doctype или Document Type Declaration сообщает веб-браузеру тип документа. <!DOCTYPE html> указывает, что данный документ является документом html и что используется html5, а не html4 или какая-то другая версия языка разметки.

А элемент html между своим открывающим и закрывающим тегами содержит все содержимое документа.

Внутри элемента html мы можем разместить два других элемента: head и body. Элемент head содержит метаданные веб-страницы - заголовок веб-страницы, тип кодировки и т.д., а также ссылки на внешние ресурсы - стили, скрипты, если они используются. Элемент body собственно определяет содержимое html-страницы.

Теперь изменим содержимое файла index.html следующим образом:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Документ HTML5</title>
    </head>
    <body>
        <div>Содержание документа HTML5</div>
    </body>
</html>
```

В элементе head определено два элемента:

элемент title представляет заголовок страницы

элемент meta определяет метаинформацию страницы. Для корректного отображения символов предпочтительно указывать кодировку. В данном случае с помощью атрибута charset="utf-8" указываем кодировку utf-8.

В пределах элемента элемента body используется только один элемент - div, который оформляет блок. Содержимым этого блока является простая строка.

Поскольку мы выбрали в качестве кодировки utf-8, то браузер будет отображать веб-страницу именно в этой кодировке. Однако необходимо, чтобы сам текст документа также соответствовал выбранной кодировке utf-8. Как правило, в различных текстовых редакторах есть соотвествующие настройки для установки кодировки. Например, в Notepad++ надо зайти в меню Кодировки и в открывшемся списке выбрать пункт Преобразовать в UTF-8 без BOM:

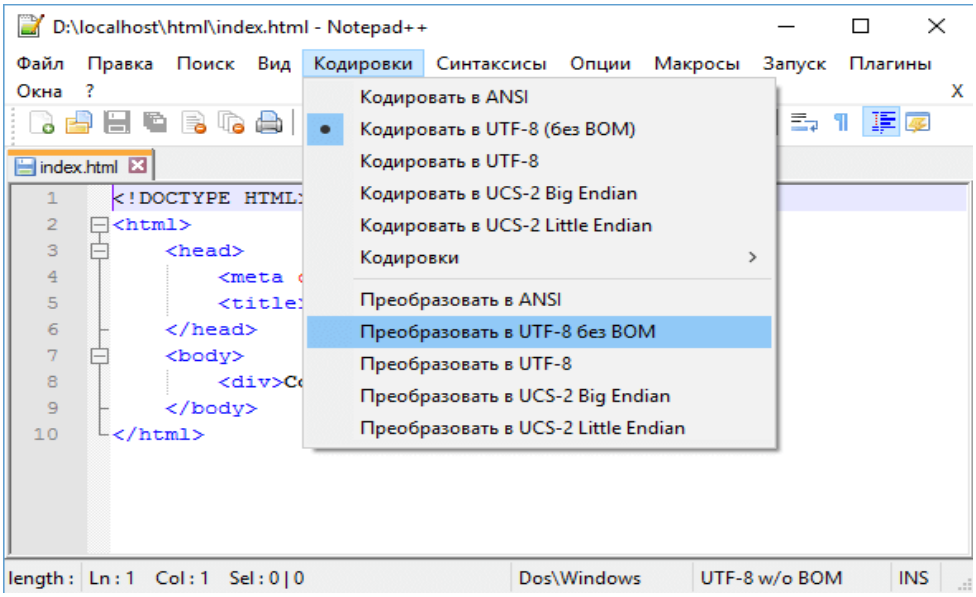

Рис. 1.2. Выбор пункта «Преобразовать в UTF-8».

После этого в статусной строке можно будет увидеть UTF-8 w/o BOM, что будет указывать, что нужная кодировка установлена.

Сохраним и откроем файл index.html в браузере:

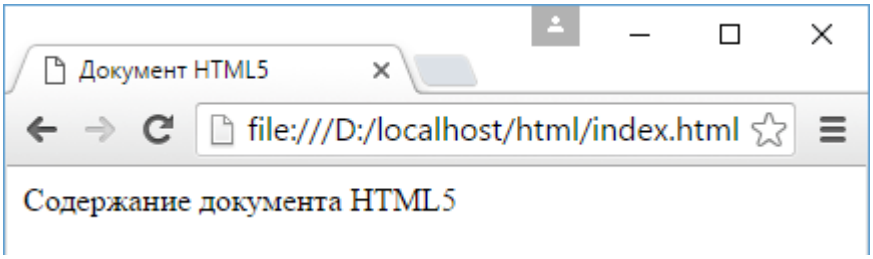

Рис. 1.3. Откройте файл index.html в браузере.

Таким образом, мы создали первый документ HTML5. Так как мы указали в элементе title заголовок "Документ HTML5", то именно такое название будет иметь вкладка браузера.

Так как указана кодировка utf-8, то веб-браузер будет корректно отображать кириллические символы.

А весь текст, определенный внутри элемента body, мы увидим в основном поле браузера.

Контрольные вопросы.

1. Что такое HTML?
2. Как создать html-документ?
3. Перечислите браузеры и их типы.

ЛАБОРАТОРНАЯ РАБОТА №2

Создание документа, шаблона в HTML

Цель работы: Приобрести навыки использования основных тегов и элементов HTML.

Задание: Создание веб-страницы с использованием основных HTML-тегов.

Методические указания:

Элементы в HTML5.

Основная часть документа html, фактически все, что мы увидим в своем браузере при загрузке веб-страницы, располагается между тегами <body> и </body>. Здесь размещаются большинство элементов html.

Хотя большинство элементов в HTML5 остаются теми же, что и в ранних версиях, но несколько изменился способ их использования. Рассмотрим базовые элементы HTML5, их предназначение и использование.

Элемент head и метаданные веб-страницы.

Как правило, одним из первых элементов html-документа является элемент head, задача которого состоит в установке метаданных страницы и ряда сопроводительной информации. Метаданные содержат информацию о html-документе.

Заголовок

Для установки заголовка документа, который отображается на вкладке браузера, используется элемент title

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Элемент title</title>
    </head>
    <body>
        <p>Содержание документа HTML5</p>
    </body>
</html>
```

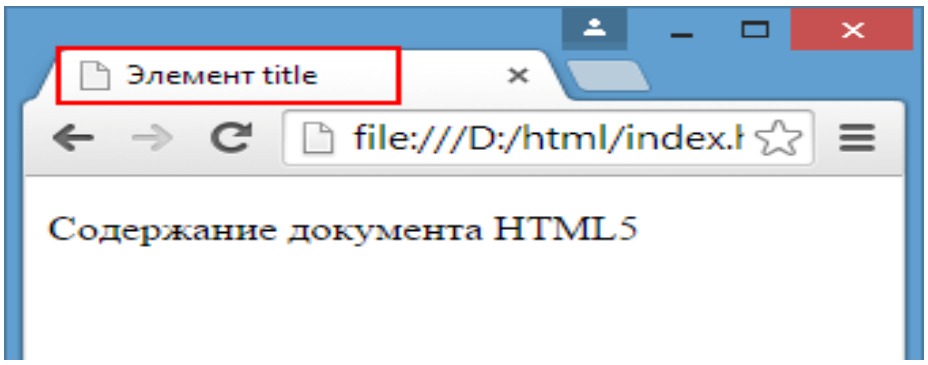

Рис. 2.1. Результат использования элемента <title>.

Элемент base позволяет указать базовый адрес, относительно которого устанавливаются другие адреса, используемые в документе:

```
<!DOCTYPE html>
<html>
  <head>
    <base href="content/">
    <meta charset="utf-8">
    <title>Элемент base</title>
  </head>
  <body>
    <a href="newpage.html">Перейти</a>
  </body>
</html>
```

Хотя для ссылки в качестве адреса указана страница newpage.html, но фактически ее адресом будет content/newpage.html. То есть в одной папке с текущей страницей должна быть подпапка content, в которой должен находится файл newpage.html

Можно также указывать полный адрес:

```
<base href="http://www.microsoft.com/">
```

В этом случае ссылка будет вести по адресу http://www.microsoft.com/newpage.html

Элемент meta.

Элемент meta определяет метаданные документа.

Чтобы документ корректно отображал текст, необходимо задать кодировку с помощью атрибута charset. Рекомендуемой кодировкой является utf-8:

```
<meta charset="utf-8">
```

При этом надо помнить, что указанная элементе meta кодировка должна совпадать с кодировкой самого документа. Как правило,

текстовый редактор позволяет указать кодировку документа. Если мы хотим ориентироваться на utf-8, то в настройках текстового редактора надо выбирать UTF-8 w/o BOM. Например, выбор кодировки в Notepad++:

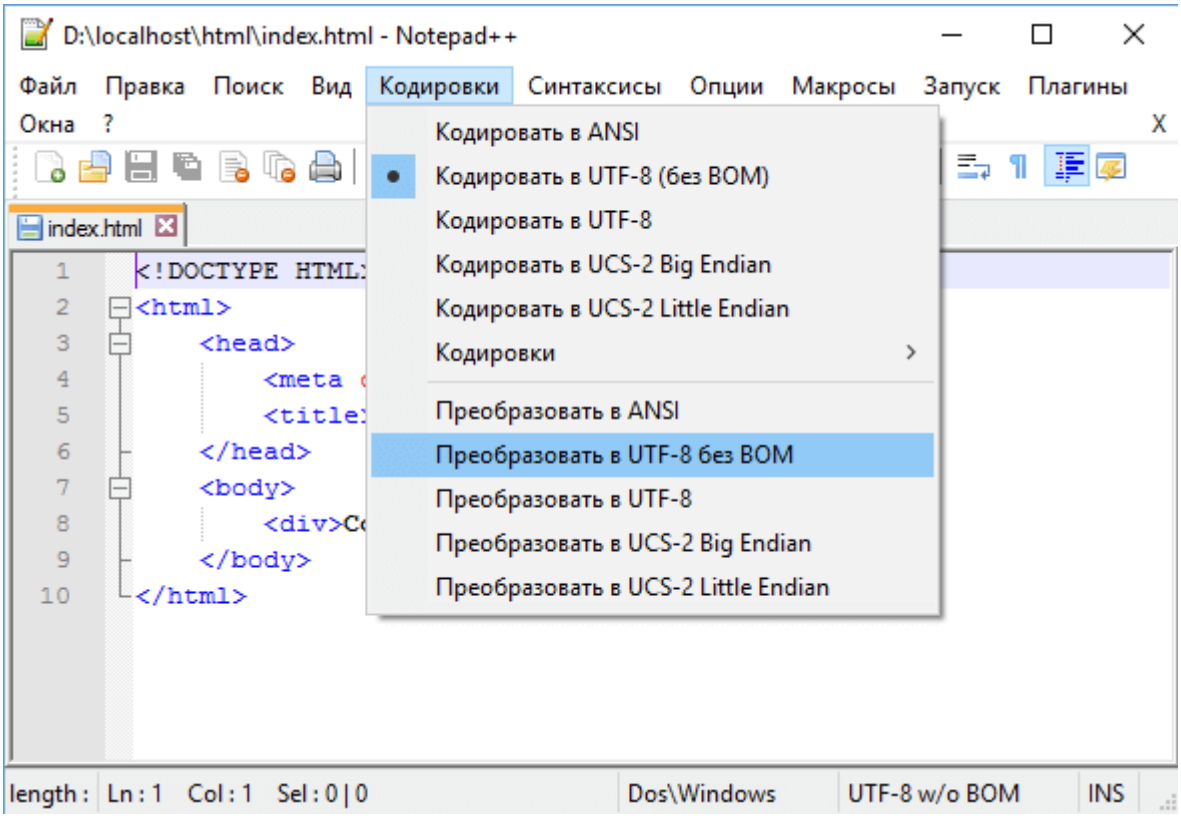

Рис. 2.2. Выбор пункта «Преобразовать в UTF-8».

Элемент meta также имеет два атрибута: name и content. Атрибут name содержит имя метаданных, а content - их значение.

По умолчанию в HTML определены пять типов метаданных:

application name: название веб-приложения, частью которого является данный документ

author: автор документа

description: краткое описание документа

generator: название программы, которая сгенерировала данный документ

keywords: ключевые слова документа

Надо отметить, что наиболее актуальным является тип description. Его значение поисковики часто используют в качестве аннотации к документу в поисковой выдаче.

Добавим в документ ряд элементов meta:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <base href="content/">
    <title>Элемент title</title>
    <meta name="description" content="Первый документ
HTML5">
    <meta name="author" content="Bill Gates">
  </head>
  <body>
    <a href="newpage.html">Содержание документа HTML5</a>
  </body>
</html>
```

Элементы группировки.

Элемент *div* служит для структуризации контента на веб-странице, для заключения содержимого в отдельные блоки. Div создает блок, который по умолчанию растягивается по всей ширине браузера, а следующий после div элемент переносится на новую строку. Например:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Документ HTML5</title>
  </head>
  <body>
    <div>Заголовок документа HTML5</div>
    <div>Текст документа HTML5</div>
  </body>
</html>
```

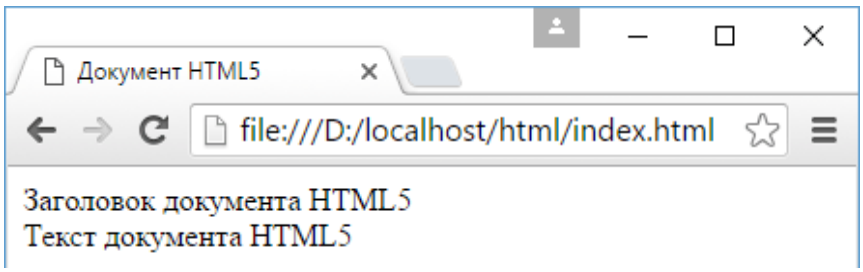

Рис. 2.3. Результат использования элемента .

Контрольные вопросы.

1. Каково назначение доменных имен в Интернете?
2. Какие атрибуты имеет элемент <a> - гиперссылка?
3. Какой атрибут можно использовать для создания редактируемого поля?
4. Что такое интернет-протоколы?
5. Какие атрибуты являются универсальными атрибутами элементов HTML?

ЛАБОРАТОРНАЯ РАБОТА №3

Работа с текстами в HTML

Цель работы: Приобрести навыки работы с заголовками и параграфами в Html.

Задание: Создание страниц с использованием заголовков и абзацев в HTML.

Методические указания:

Параграфы создаются с помощью тегов <*p*> и </*p*>, которые заключают некоторое содержимое. Каждый новый параграф располагается на новой строке. Применим параграфы:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Документ HTML5</title>
    </head>
    <body>
        <div>Заголовок документа HTML5</div>
        <div>
            <p>Первый параграф</p>
            <p>Второй параграф</p>
        </div>
    </body>
</html>
```

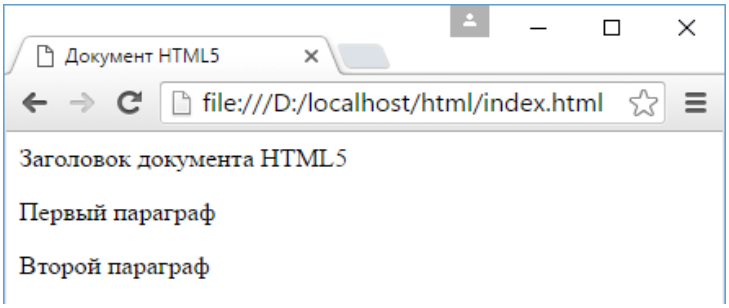

Рис. 3.1. Результат использования элемента <p>.

Если в рамках одного параграфа нам надо перенести текст на другую строку, то мы можем воспользоваться элементом <*br*>:

```
<p>Первая строка.<br/>Вторая строка.</p>
```

Элемент pre

Элемент *pre* выводит предварительно отформатированный текст так, как он определен:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Документ HTML5</title>
    </head>
    <body>
        <pre>
            Первая строка
            Вторая строка
            Третья строка
        </pre>
    </body>
</html>
```

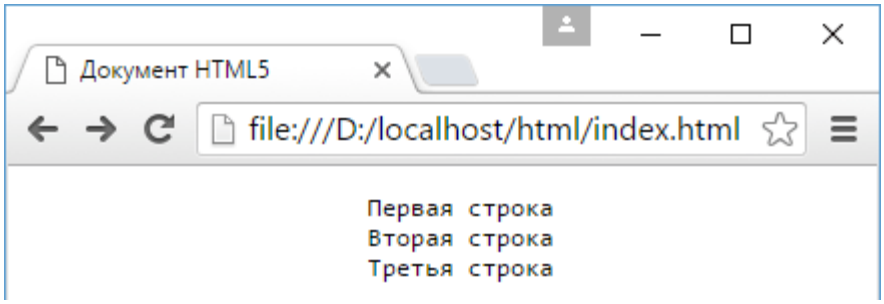

Рис. 3.2. Результат использования элемента <pre>.

Элемент *span* обтекает некоторый текст по всей его длине и служит преимущественно для стилизации заключенного в него текстового содержимого. В отличие от блоков div или параграфов span не переносит содержимое на следующую строку:

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8">
      <title>Документ HTML5</title>
   </head>
   <body>
      <div>Заголовок документа HTML5</div>
      <div>
         <p><span style="color:red;">Первый</span> параграф</p>
         <p><span>Второй</span> параграф</p>
      </div>
   </body>
</html>
```

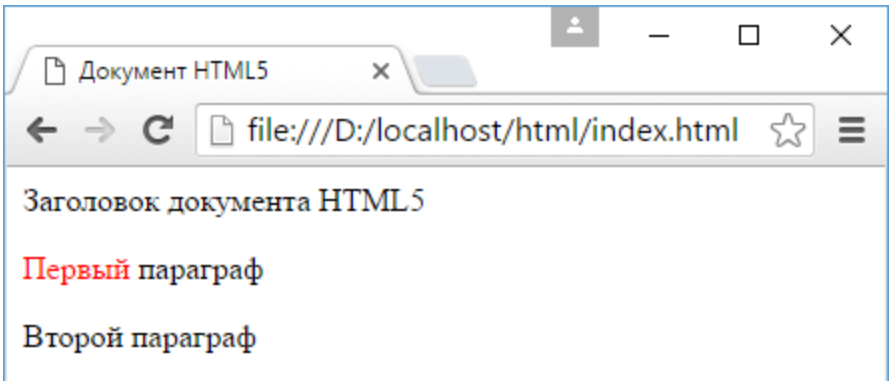

Рис. 3.3. Результат использования элемента .

При этом стоит отметить, что сам по себе span ничего не делает. Так, во втором параграфе span никак не повлиял на внутреннее текстовое содержимое. А в первом параграфе элемент span содержит атрибут стиля: style="color:red;", который устанавливает для вложенного текста красный цвет фона.

При этом стоит отметить, что элементы div и p являются блочными, элемент div может содержать любые другие элементы, а элемент p - только строчные элементы. В отличие от них элемент span является строчным, то есть как бы встраивает свое содержимое

во внешний контейнер - тот же div или параграф. Но при этом не следует помещать блочные элементы в строчный элемент span.

Элементы <h1>, <h2>, <h3>, <h4>, <h5> и <h6> служат для создания заголовков различного уровня:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Заголовки в HTML5</title>
  </head>
  <body>
    <h1>Заголовок первого уровня</h1>
    <h2>Заголовок второго уровня</h2>
    <h3>Заголовок третьего уровня</h3>
    <h4>Заголовок четвертого уровня</h4>
    <h5>Заголовок пятого уровня</h5>
    <h6>Заголовок шестого уровня</h6>
  </body>
</html>
```

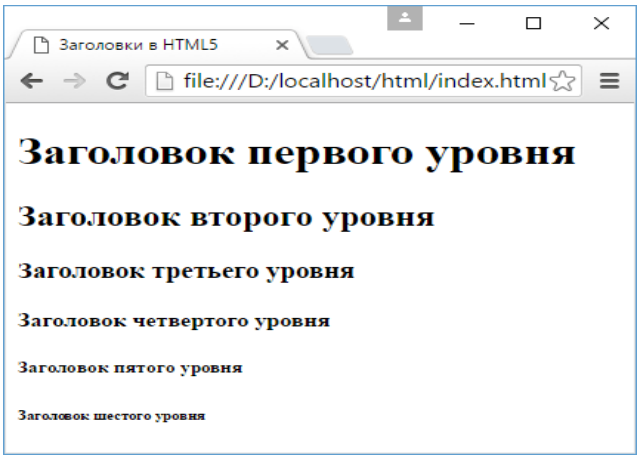

Рис. 3.4. Результат использования элемента заголовки в HTML.

Заголовки выделяют шрифт жирным и по умолчанию имеют некоторый размер: от самого крупного <h1> до самого мелкого <h6>.

При определении заголовков следует учитывать, что на странице должен быть только один заголовок первого уровня, то есть <h1>. Он выполняет роль основного заголовка веб-страницы.

Форматирование текста.

Ряд элементов html предназначены для форматирования текстового содержимого, например, для выделения жирным или курсивом и т.д. Рассмотрим эти элементы:

<*b*>: выделяет текст жирным

<*del*>: зачеркивает текст

<*i*>: выделяет текст курсивом

<*em*>: выделяет текст курсивом, в отличие от тега <i> носит логическое значение, придает выделяемому тексту оттенок важности

<*s*>: зачеркивает текст

<*small*>: делает текст чуть меньше размером, чем окружающий

<*strong*>: выделяет текст жирным. В отличие от тега <b> предназначен для логического выделения, чтобы показать важность текста. А <b> не носит характера логического выделения, выполняет функции только форматирования

<*sub*>: помещает текст под строкой

<*sup*>: помещает текст над строкой

<*u*>: подчеркивает текст

<ins>: определяет вставленный (или добавленный) текст

<*mark*>: выделяет текст цветом, придавая ему оттенок важности

Применим все эти элементы:

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8">
      <title>Форматирование текста в HTML5</title>
   </head>
   <body>
      <p>Форматирование в HTML5 <mark>HTML5</mark></p>
      <p>Это <b>выделенный</b> текст</p>
      <p>Это <strong>важный</strong> текст</p>
      <p>Это <del>зачеркнутый</del> текст</p>
      <p>Это <s>недействительный</s> текст</p>
      <p>Это <em>важный</em> текст</p>
      <p>Это текст <i>курсивом</i> </p>
      <p>Это <ins>добавленный</ins> текст</p>
      <p>Это <u>подчеркнутый</u> текст</p>
      <p>X<sub>i</sub>=Y<sup><small>2</small></sup>
       +Z<sup><small>2</small></sup></p>
   </body>
```

</html>

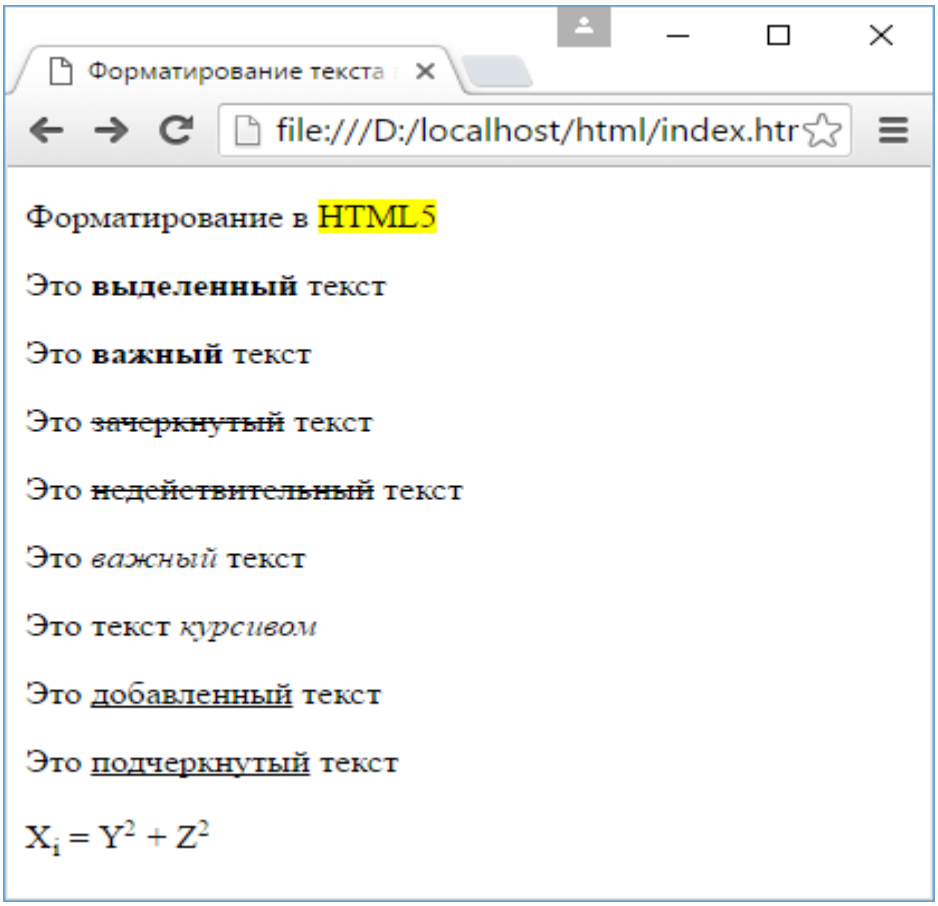

Рис. 3.5. Форматирование текста в HTML.

Контрольные вопросы.

1. Как использовать специальные символы в HTML-документе?
2. Как работать с цветами в HTML-документе?
3. Напишите элементы HTML, которые создают гиперссылки.
4. Какой тег можно использовать для создания горизонтального меню сайта?
5. Форматирование текста в HTML5.

ЛАБОРАТОРНАЯ РАБОТА №4

Работа с изображениями и списками в HTML

Цель работы: Приобрести навыки работы с изображениями и списками в HTML.

Задание: Создание страниц с использованием изображений и списков в HTML.

Методические указания:

Для вывода изображений в HTML используется элемент *img*. Этот элемент представляет нам два важных атрибута:

src: путь к изображению. Это может быть относительный или абсолютный путь в файловой системе или адрес в интернете

alt: текстовое описание изображения. Если браузер по каким-то причинам не может отобразить изображение (например, если у атрибута src некорректно задан путь), то браузер показывает вместо самой картинки данное текстовое описание.

Атрибут alt еще важен тем, что поисковые системы по текстовому описанию могут индексировать изображение.

Например, положим в ту же папку, где у нас лежит файл index.html, какой-нибудь файл изображения. И затем отобразим его на веб-странице:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Тег img в HTML5</title>
    </head>
    <body>
        <img src="dubi.png" alt="Зимняя равнина" />
    </body>
</html>
```

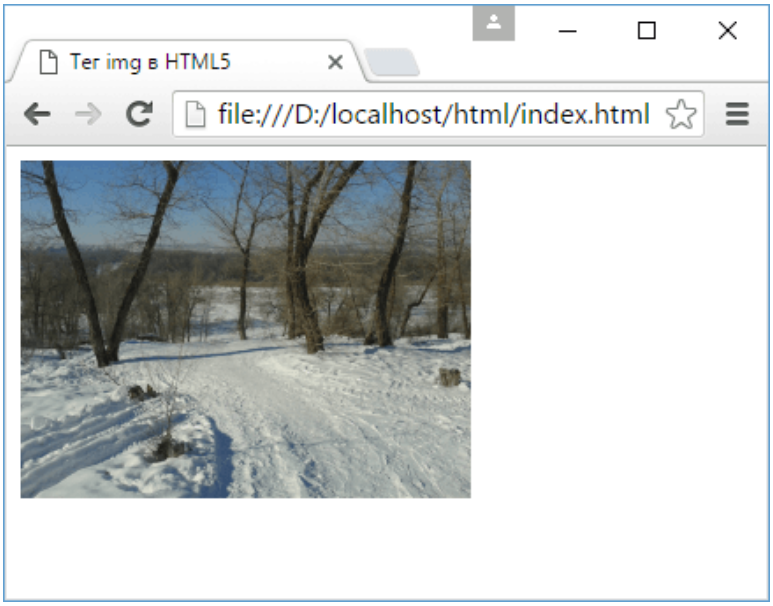

Рис. 4.1. Элемент <img> в HTML.

В нашем случае файл изображения называется dubi.png, и он находится в одной папке с веб-страницой index.html. При этом надо учитывать, что img является пустым элементом, то есть не содержит закрывающегося тега. Используя стилевые особенности, в частности, отступы и обтекание, можно комбинировать изображения с текстом. Например:

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8">
      <title>Тег img в HTML5</title>
   </head>
   <body>
      <div>
         <img src="dubi.png" alt="Зимняя равнина" style="float:left; margin-right:10px;" />
         <h1>Lorem Ipsum</h1>
         <b>Lorem Ipsum</b> is simply dummy text of the printing and typesetting industry.
            Lorem Ipsum has been the industry....
      </div>
   </body>
</html>
```

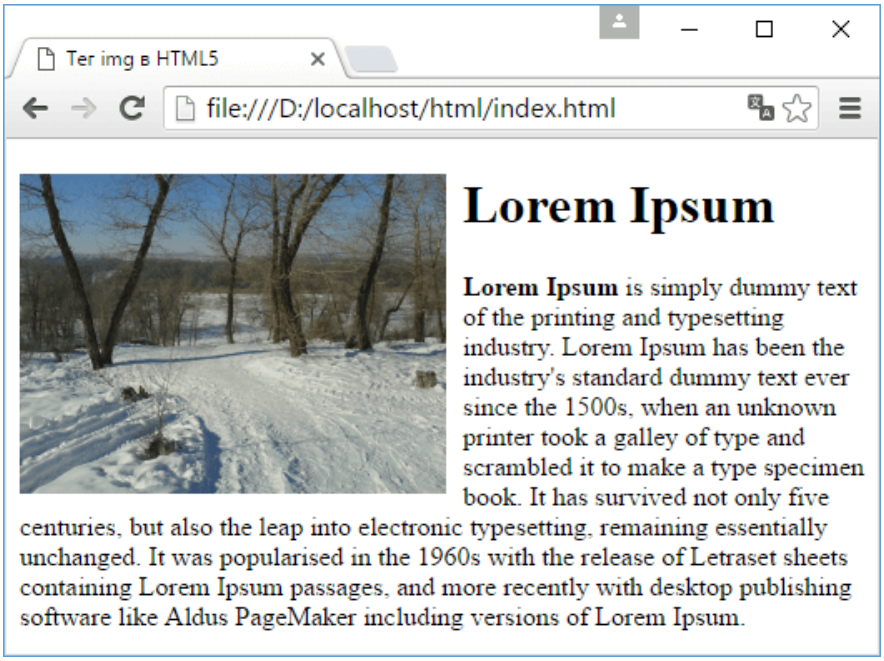

Рис. 4.2. Комбинировать изображения с текстом.

Для создания списков в HTML5 применяются элементы <ol> (нумерованный список) и <ul> (ненумерованный список):

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Списки в HTML5</title>
  </head>
  <body>
    <h2>Нумерованный список</h2>
    <ol>
      <li>iPhone 6S</li>
      <li>Galaxy S7</li>
      <li>Nexus 5X</li>
      <li>Lumia 950</li>
    </ol>
    <h2>Ненумерованный список</h2>
    <ul>
      <li>iPhone 6S</li>
      <li>Galaxy S7</li>
      <li>Nexus 5X</li>
      <li>Lumia 950</li>
    </ul>
  </body>
</html>
```

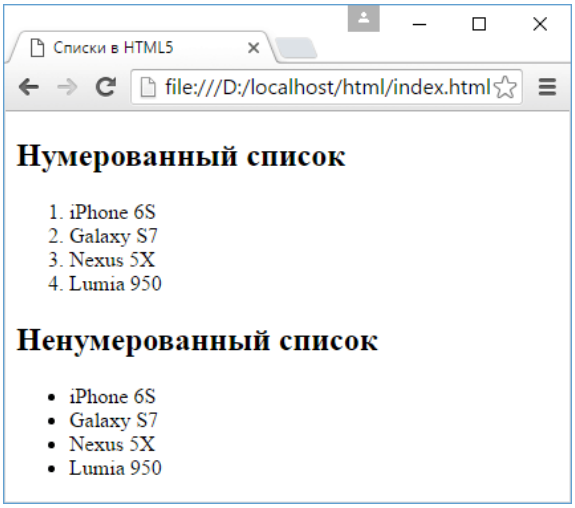

Рис. 4.3. Списки в HTML5.

В нумерованном списке для нумерации элементов по умолчанию используется стандартные цифры от 1. В ненумерованном списке каждый элемент предваряется черной точкой.

При необходимости мы можем настроить нумерацию или отражаемый рядом с элементом символ с помощью стиля list-style-type. Данный стиль может принимать множество различных значений. Отметим только основные и часто используемые. Для нумерованных списков стиль list-style-type может принимать следующие значения:

decimal: десятичные числа, отсчет идет от 1

decimal-leading-zero: десятичные числа, которые предваряются нулем, например, 01, 02, 03, … 98, 99

lower-roman: строчные римские цифры, например, i, ii, iii, iv, v

upper-roman: заглавные римские цифры, например, I, II, III, IV, V…

lower-alpha: строчные римские буквы, например, a, b, c..., z

upper-alpha: заглавные римские буквы, например, A, B, C, … Z

Для нумерованных список с помощью атрибута start можно дополнительно задать символ, с которого будет начинаться нумерация. Например:

```
<h2>list-style-type = decimal</h2>
<ol style="list-style-type:decimal;" start="3">
   <li>iPhone 6S</li>
   <li>Galaxy S7</li>
   <li>Nexus 5X</li>
   <li>Lumia 950</li>
</ol>
<h2>list-style-type = upper-roman</h2>
<ul style="list-style-type:upper-roman;">
   <li>iPhone 6S Plus</li>
   <li>Galaxy S7 Edge</li>
   <li>Nexus 6P</li>
   <li>Lumia 950 XL</li>
</ul>
<h2>list-style-type = lower-alpha</h2>
<ul style="list-style-type:lower-alpha;">
   <li>LG G 5</li>
   <li>Huawei P8</li>
   <li>Asus ZenFone 2</li>
```

```
</ul>
```

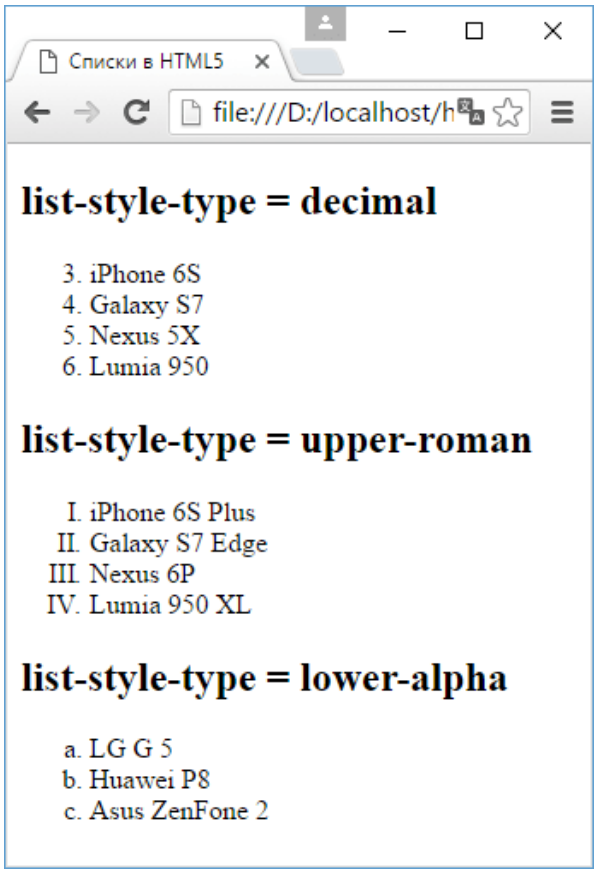

Рис.4.4. Типы списков в HTML5.

Для ненумерованного списка атрибут list-style-type может принимать следующие значения:

disc: черный диск

circle: пустой кружочек

square: черный квадратик

Например:

```
<h2>list-style-type = disc</h2>
<ul style="list-style-type:disc;">
   <li>iPhone 6S</li>
   <li>Galaxy S7</li>
   <li>Nexus 5X</li>
   <li>Lumia 950</li>
</ul>
<h2>list-style-type = circle</h2>
<ul style="list-style-type:circle;">
   <li>iPhone 6S Plus</li>
```

```
    <li>Galaxy S7 Edge</li>
    <li>Nexus 6P</li>
    <li>Lumia 950 XL</li>
</ul>
<h2>list-style-type = square</h2>
<ul style="list-style-type:square;">
    <li>LG G 5</li>
    <li>Huawei P8</li>
    <li>Asus ZenFone 2</li>
</ul>
```

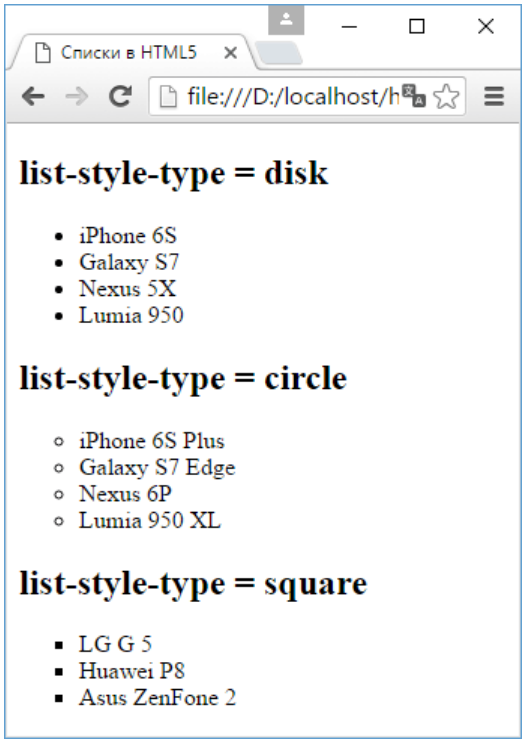

Рис. 4.5. Типы списков в HTML5.

Еще одну интересную возможность по настройке списков предоставляет стиль *list-style-image*. Он задает изображение, которое будет отображаться рядом с элементом списка:

```
<ul style="list-style-image:url(phone_touch.png);">
    <li>iPhone 6S</li>
    <li>Galaxy S7</li>
    <li>Nexus 5X</li>
    <li>Lumia 950</li>
</ul>
```

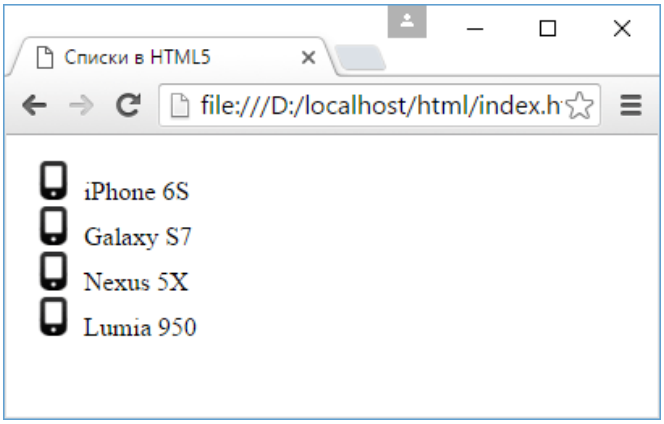

Рис. 4.6. Типы списков в HTML5.

Стиль list-style-image в качестве значения принимает url(phone_touch.png), где "phone_touch.png" - это название файла изображения. То есть в данном случае предполагается, что в одной папке с веб-страницей index.html у меня находится файл изображения phone_touch.png.

Горизонтальный список

Одним из распространенных способов стилизации списков представляет создание горизонтального списка. Для этого для всех элементов списка надо установить стиль display:inline:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Горизонтальный список в HTML5</title>
    <style>
      ul#menu li {
        display:inline;
      }
    </style>
  </head>
  <body>
    <ul id="menu">
      <li>Главная</li>
      <li>Блог</li>
      <li>Форум</li>
      <li>О сайте</li>
    </ul>
```

```
    </body>
</html>
```

Контрольные *вопросы.*

1. Какие существуют форматы графических файлов?
2. Какой элемент HTML встраивает изображение в документ?
3. Какие элементы HTML определяют структуру документа?
4. Как создать карту изображения, используя элементы IMG и AREA?
5. Что такое служба World Wide Web и как она работает?

ЛАБОРАТОРНАЯ РАБОТА №5

Работа с блоками в HTML. Элементы header, footer и address.

Цель работы: Работа с блоками в HTML. Приобрести навыки использования тегов заголовка, нижнего колонтитула и адреса.

Задание: Работа с блоками в HTML. Изучение и использование тегов заголовка, нижнего колонтитула и адреса.

Методические указания:

Элементы header, footer и address

Элемент header является как бы вводным элементом, предваряющим основное содержимое. Здесь могут быть заголовки, элементы навигации или какие-либо другие вспомогательные элементы, например, логотип, форма поиска и т.п. Например:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Семантическая разметка в HTML5</title>
  </head>
  <body>
    <header>
      <h1>Онлайн-магазин телефонов</h1>
        <nav>
          <ul>
            <li><a href="/apple">Apple</a>
            <li><a href="/microsoft">Microsoft</a>
            <li><a href="/samsung">Samsung</a>
          </ul>
```

```
            </nav>
        </header>
        <div>
            Информация о новинках мобильного мира....
        </div>
    </body>
</html>
```

Элемент header нельзя помещать в такие элементы как address, footer или другой header.

Элемент footer обычно содержит информацию о том, кто автор контента на веб-странице, копирайт, дата публикации, блок ссылок на похожие ресурсы и т.д. Как правило, подобная информация располагается в конце веб-страницы или основного содержимого, однако, footer не имеет четкой привязки к позиции и может использоваться в различных местах веб-страницы.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Семантическая разметка в HTML5</title>
    </head>
    <body>
        <h1>Xiaomi Mi 5</h1>
        <div>
            Xiaomi Mi 5 оснащен восьмиядерным процессором Qualcomm Snapdragon 820.
            Размер внутреннего хранилища - 32 и 64 МБ.
        </div>
        <footer>
            <p><a href="/license">Лицензионное соглашение</a><br/>
            Copyright © 2016. SomeSite.com</p>
        </footer>
    </body>
</html>
```

Здесь определен футер для всей веб-страницы. В него помещена ссылка на лицензионное соглашение использования сервисом и информация о копирайте.

Футер необязательно должен быть определен для всей страницы. Это может быть и отдельная секция контента:

```
<section>
        <h1>Последние статьи</h1>
        <article>
          <h2>Анонс Samsung Galaxy S7</h2>
          <p>Состоялся выход нового флагмана от компании Samsung Galaxy S7.....</p>
          <footer>
            Дата публикации: <time datetime="2016-03-16T15:16-00:00">16.03.2016 15:16</TIME>
          </footer>
        </article>
        <article>
          <h2>Скидки на Microsoft Lumia 950</h2>
          <p>С 1 марта смартфон Microsoft Lumia 950 стоит на 10 000 рублей дешевле</p>
          <footer>
            Дата публикации: <time datetime="2016-03-01T14:36-00:00">01.03.2016 14:36</TIME>
          </footer>
        </article>
      </section>
    </body>
</html>
```

Элемент footer не следует помещать в такие элементы как address, header или другой footer.

Address

Элемент address предназначен для отображения контактной информации, которая связана с ближайшим элементом article или body. Нередко данный элемент размещается в футере:

```
<footer>
    <address>
      Контакты для связи <a href="mailto:js@example.com">Том Смит</a>.
    </address>
    <p>© copyright 2016 Example Corp.</p>
</footer>
```

Контрольные вопросы.

1. Какой атрибут можно использовать для создания редактируемого поля?
2. Что такое интернет-протоколы?
3. Как реализуются универсальные атрибуты HTML-атрибутов?
4. Какие заголовки получаются в HTML?
5. Что такое служба World Wide Web и как она работает?

ЛАБОРАТОРНАЯ РАБОТА №6

Основы CSS3. Селекторы

Цель работы: Приобрести навыки работы с CSS.

Задание: Использование CSS. Обработка веб-страницы с помощью CSS.

Методические указания:

Основы CSS3. Селекторы

Любой html-документ, сколько бы он элементов не содержал, будет по сути "мертвым" без использования стилей. Стили или лучше сказать каскадные таблицы стилей (*Cascading Style Sheets*) или попросту CSS определяют представление документа, его внешний вид. Рассмотрим вкратце применение стилей в контексте HTML5.

Стиль в CSS представляет правило, которое указывает веб-браузеру, как надо форматировать элемент. Форматирование может включать установку цвета фона элемента, установку цвета и типа шрифта и так далее.

Определение стиля состоит из двух частей: *селектор*, который указывает на элемент, и *блок объявления стиля* - набор команд, которые устанавливают правила форматирования. Например:

```
div{
   background-color:red;
   width: 100px;
   height: 60px;
}
```

В данном случае селектором является div. Этот селектор указывает, что этот стиль будет применяться ко всем элементам div.

После селектора в фигурных скобках идет *блок объявления стиля*. Между открывающей и закрывающей фигурными скобками определяются команды, указывающие, как форматировать элемент.

Каждая команда состоит из *свойства* и *значения*. Так, в следующем выражении:

background-color:red;

background-color представляет свойство, а red - значение. Свойство определяет конкретный стиль. Свойств css существует множество. Например, background-color определяет цвет фона. После двоеточия идет значение для этого свойства. Например, выше-указанная команда определяет для свойства background-color значение red. Иными словами, для фона элемента устанавливается цвет "red", то есть красный.

После каждой команды ставится точка с запятой, которая отделяет данную команду от других.

Наборы таких стилей часто называют таблицами стилей или CSS (*Cascading Style Sheets* или каскадные таблицы стилей). Существуют различные способы определения стилей.

Атрибут style

Первый способ заключается во встраивании стилей непосредственно в элемент с помощью атрибута *style*:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Стили</title>
    </head>
    <body>
        <h2 style="color:blue;">Стили</h2>
        <div style="width: 100px; height: 100px; background-color: red;"></div>
    </body>
</html>
```

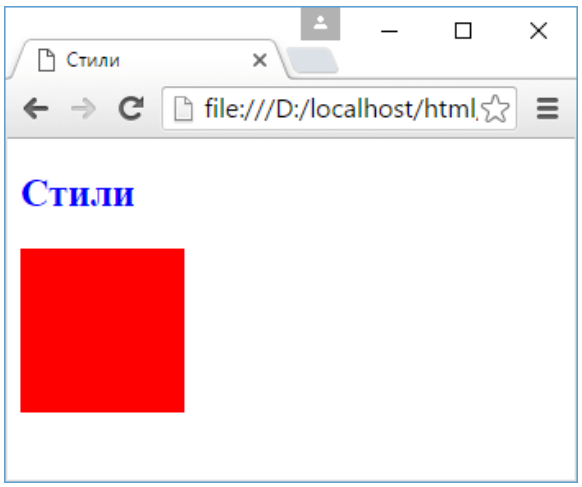

Рис.6.1. Использование атрибута стиля.

Здесь определены два элемента - заголовок h2 и блок div. У заголовка определен синий цвет текста с помощью свойства color. У блока div определены свойства ширины (width), высоты (height), а также цвета фона (background-color).

Второй способ состоит в использования элемента style в документе html. Этот элемент сообщает браузеру, что данные внутри являются кодом css, а не html:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Стили</title>
        <style>
        h2{
            color:blue;
        }
        div{
            width: 100px;
            height: 100px;
            background-color: red;
        }
        </style>
    </head>
    <body>
        <h2>Стили</h2>
        <div></div>
```

```
    </body>
</html>
```

Результат в данном случае будет абсолютно тем же, что и в предыдущем случае.

Часто элемент style определяется внутри элемента head, однако может также использоваться в других частях HTML-документа. Элемент style содержит наборы стилей. У каждого стиля указывается вначале селектор, после чего в фигурных скобках идет все те же определения свойств css и их значения, что были использованы в предыдущем примере.

Второй способ делает код html чище за счет вынесения стилей в элемент style. Но также есть и третий способ, который заключается в вынесении стилей во внешний файл.

Создадим в одной папке с html странице текстовый файл, который переименуем в styles.css и определим в нем следующее содержимое:

```
h2{
    color:blue;
}
div{
    width: 100px;
    height: 100px;
    background-color: red;
}
```

Это те же стили, что были внутри элемента style. И также изменим код html-страницы:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Стили</title>
        <link rel="stylesheet" type="text/css" href="styles.css"/>
    </head>
    <body>
        <h2>Стили</h2>
        <div></div>
    </body>
</html>
```

Здесь уже нет элемента style, зато есть элемент link, который подключает выше созданный файл styles.css: <link rel="stylesheet" type="text/css" href="styles.css"/>

Таким образом, определяя стили во внешнем файле, мы делаем код html чище, структура страницы отделяется от ее стилизации. При таком определении стили гораздо легче модифицировать, чем если бы они были определены внутри элементов или в элементе style, и такой способ является предпочтительным в HTML5.

Использование стилей во внешних файлах позволяет уменьшить нагрузку на веб-сервер с помощью механизма кэширования. Поскольку веб-браузер может кэшировать css-файл и при последующем обращении к веб-странице извлекать нужный css-файл из кэша.

Также возможна ситуация, когда все эти подходы сочетаются, а для одного элемента одни свойства css определены внутри самого элемента, другие свойства css определены внутри элемента style, а третьи находятся во внешнем подключенном файле. Например:

```
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" type="text/css" href="styles.css"/>
        <style>
            div{
                width:200px;
            }
        </style>
    </head>
    <body>
        <div style="width:120px;"></div>
    </body>
</html>
```

А в файле style.css определен следующий стиль:

```
div{
    width:50px;
    height:50px;
    background-color:red;
}
```

В данном случае в трех местах для элемента div определено свойство width, причем с разным значением. Какое значение будет

применяться к элементу в итоге? Здесь у нас действует следующая система приоритетов:

Если у элемента определены встроенные стили (inline-стили), то они имеют высший приоритет, то есть в примере выше итоговой шириной будет 120 пикселей

Далее в порядке приоритета идут стили, которые определены в элементе style

Наименее приоритетными стилями являются те, которые определены во внешнем файле.

Атрибуты html и стили css

Многие элементы html позволяют устанавливать стили отображения с помощью атрибутов. Например, у ряда элементов мы можем применять атрибуты width и height для установки ширины и высоты элемента соответственно. Однако подобного подхода следует избегать и вместо встроенных атрибутов следует применять стили CSS. Важно четко понимать, что разметка HTML должна предоставлять только структуру html-документа, а весь его внешний вид, стилизацию должны определять стили CSS.

Валидация кода CSS

В процессе написания стилей CSS могут возникать вопросы, а правильно ли так определять стили, корректны ли они. И в этом случае мы можем воспользоваться валидатором css, который доступен по адресу http://jigsaw.w3.org/css-validator/.

Контрольные вопросы.

1. Что такое CSS?
2. В чем разница между CSS и HTML?
3. Как привязать стили CSS к странице?
4. Сколько типов CSS существует?
5. Какова цель точки с запятой в правилах CSS?

ЛАБОРАТОРНАЯ РАБОТА №7

Цвета и типы шрифтов в CSS

Цель работы: Приобрести навыки использования цветов и типов шрифтов в CSS.

Задание: Использование CSS. Обработка веб-страницы с помощью CSS.

Методические указания:

Цвет в CSS

В CSS широкое распространение находит использование цветов. Чтобы установить цвет текста, фона или границы, нам надо указать цвет.

Например, определим красный цвет для фона элемента div:

```
div{
   background-color: red;
}
```

В CSS есть несколько различных свойств, которые в качестве значения требует определенный цвет. Например, за установку цвета текста отвечает свойство color, за установку фона элемента - свойство background-color, а за установку цвета границы - border-color.

Существует несколько различных способов определения цвета текста.

Шестнадцатеричное значение. Оно состоит из отдельных частей, которые кодируют в шестнадцатеричной системе значения для красного, зеленого и синего цветов.

Например, *#1C4463*. Здесь первые два символа 1C представляют значение красной компоненты цвета, далее 44 - значение зеленой компоненты цвета и 63 - значение уровня синего цвета. Финальный цвет, который мы видим на веб-странице, образуется с помощью смешивания этих значений.

Если каждое из трех двухзначных чисел содержит по два одинаковых символа, то их можно сократить до одного. Например, #5522AA можно сократить до #52A, или, к примеру, #eeeeee можно сократить до #eee. При этом не столь важно, в каком регистре будут символы.

Значение RGB. Значение RGB также представляет последовательно набор значений для красного, зеленого и синего цветов (Red — красный, Green — зеленый, Blue — синий). Значение каждого цвета кодируется тремя числами, которые могут представлять либо процентные соотношения (0–100%), либо число от 0 до 255.

Например

```
background-color: rgb(100%,100%,100%);
```

Здесь каждый цвет имеет значение 100%. И в итоге при смешивании этих значений будет создаваться белый цвет. А при значениях в 0% будет генерироваться черный цвет:

```
background-color: rgb(0%, 0%, 0%);
```

Между 0 и 100% будут находиться все остальные оттенки.

Но, как правило, чаще применяются значения из диапазона от 0 до 255. Например,

background-color: rgb(28, 68, 99);

Значение RGBA. Это то же самое значение RGB плюс компонент прозрачности (Alpha). Компонент прозрачности имеет значение от 0 (полностью прозрачный) до 1 (не прозрачный). Например:

background-color: rgba(28, 68, 99, .6);

Значение HSL. HSL представляет аббривиатуру: Hue — тон, Saturation — насыщенность и Lightness — осветленность. HSL задает три значения. Первое значение Hue угол в круге оттенков - значение в градусах от 0 до 360. Например, красный — 0 (или 360 при полном обороте круга). Каждый цвет занимает примерно 51°.

Второе значение - Saturation - представляет насыщенность, то указывает, насколько чистым является цвет. Насыщенность определяется в процентах от 0 (полное отсутствие насыщенности) до 100% (яркий, насыщенный цвет).

Третье значение - Lightness - определяет осветленность и указывается в процентах от 0 (полностью черный) до 100 (полностью белый). Для получения чистого цвет применяется значение 50 %.

Например:

background-color: hsl(206, 56%, 25%);

Данный цвет является эквивалентом значений #1C4463 и rgb(28, 68, 99)

Значение HSLA. Аналогично RGBA здесь к HSL добавляется компонента прозрачности в виде значения от 0 (полностью прозрачный) до 1 (не прозрачный). Например:

background-color: hsl(206, 56%, 25%, .6);

Строковые значения. Существует ряд константных строковых значений, например, red (для красного цвета) или green (для зеленого цвета). К примеру,

color: red;

является эквивалентом

color: #ff0000;

Полный перечень цветов можно найти на странице https://developer.mozilla.org/en-US/docs/Web/CSS/color_value

В заключение следует отметить, что существует множество бесплатных онлайн-генераторов цвета, где можно настроить и

посмотреть цвет в нужном формате. Например, генератор цвета на mdn.

Прозрачность

Ряд настроек цвета позволяют установить значение для альфа-компоненты, которая отвечает за прозрачность. Но также в CSS есть специальное свойство, которое позволяет установить прозрачность элементов - свойство opacity. В качестве значения оно принимает число от 0 (полностью прозрачный) до 1 (не прозрачный):

```
div{
    width: 100px;
    height: 100px;
    background-color: red;
    opacity: 0.4;
}
```

Стилизация шрифтов

Семейство шрифтов

Свойство *font-family* устанавливает семейство шрифтов, которое будет использоваться. Например:

```
body{
    font-family: Arial;
}
```

В данном случае устанавливается шрифт Arial.

Шрифт свойства font-family будет работать, только если у пользователя на локальном компьютере имеется такой же шрифт. По этой причине нередко выбираются стандартные шрифты, которые широко распространены, как Arial, Verdana и т.д.

Также нередко применяется практика нескольких шрифтов:

```
body{
    font-family: Arial, Verdana, Helvetica;
}
```

В данном случае основным шрифтом является первый - Arial. Если он на компьютере пользователя не поддерживается, то выбирается второй и т.д.

Если название шрифта состоит из нескольких слов, например, Times New Roman, то все название заключается в кавычки:

```
body{
    font-family: "Times New Roman";
}
```

Кроме конкретных стилей также могут использоваться общие универсальные шрифты, задаваемые с помощью значений *sans-serif* и *serif:*

```
body{
    font-family: Arial, Verdana, sans-serif;
}
```

Так, если ни Arial, ни Verdana не поддерживаются на компьютере пользователя, то используется sans-serif - универсальный шрифт без засечек.

Типы шрифтов. Шрифты с засечками

Шрифты с засечками названы так, потому что на на концах основных штрихов имеют небольшие засечки. Считается, что они подходят для больших кусков текста, так как визуально связывают одну букву с другой, делая текст более читабельным.

Распространенные шрифты с засечками: Times, Times New Roman, Georgia, Garamond. Универсальный обобщенный шрифт с засечками представляет значение serif.

Шрифты без засечек.

В отличие от шрифтов с засечками шрифты из этой группы не имеют засечек. Наиболее распространенные шрифты этой группы: Arial, Helvetica, Verdana.

Моноширинный шрифт преимущественно применяется для отображения программного кода и не предназначен для вывода стандартного текста статей. Свое название эти шрифты получили от того, что каждая буква в таком шрифте имеет одинаковую ширину. Примеры подобных шрифтов: Courier, Courier New, Consolas, Lucida Console.

Примеры шрифтов:

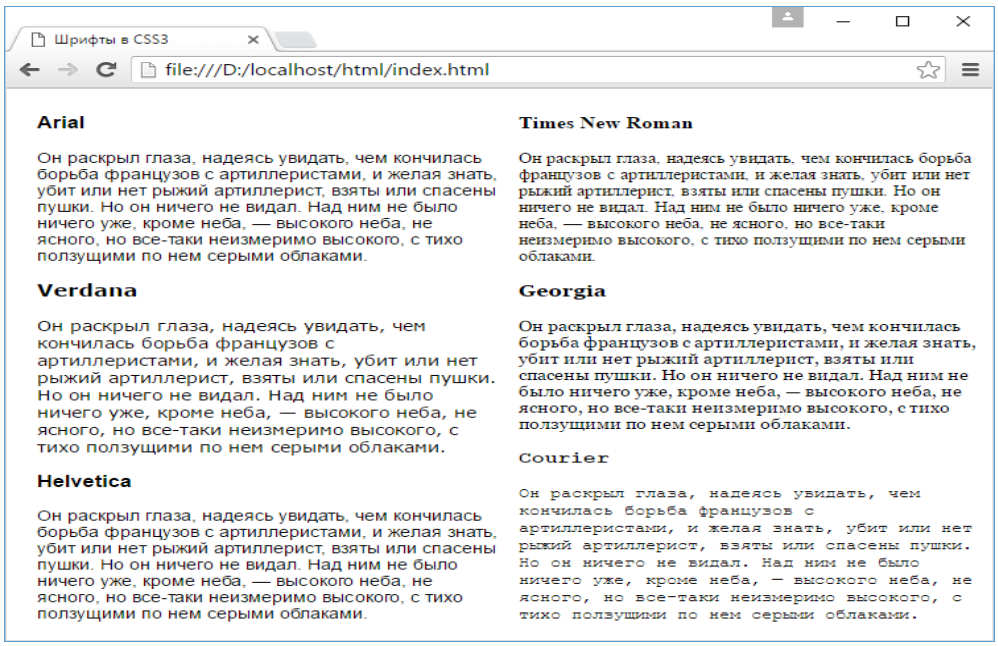

Рис.7.1. Моноширинные шрифты.

Свойство font-weight задает толщину шрифта. Оно может принимать 9 числовых значений: 100, 200, 300, 400,...900. 100 - очень тонкий шрифт, 900 - очень плотный шрифт.

В реальности чаще для этого свойства используют два значения: normal (нежирный обычный текст) и bold (полужирный шрифт):

font-weight: normal;

font-weight: bold;

Курсив

Свойство font-style позволяет выделить текст курсивом. Для этого используется значение italic:

p {font-style: italic;}

Если надо отменить курсив, то применяется значение normal:

p {font-style: normal;}

Свойство color устанавливает цвет шрифта:

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

Контрольные вопросы.

1. Как отобразить цвет в CSS?
2. Как разместить комментарии в CSS?
3. Сколько типов CSS существует?
4. Какова цель точки с запятой в правилах CSS?

ЛАБОРАТОРНАЯ РАБОТА №8

Работа с блочной моделью и границами в CSS

Цель работы: Приобрести навыки работы с моделями блогов и гранитов и CSS.

Задание: Использование CSS. Обработка веб-страницы с помощью CSS.

Методические указания:

Блочная модель

Для веб-браузера элементы страницы представляют небольшие контейнеры или блоки. Такие блоки могут иметь различное содержимое - текст, изображения, списки, таблицы и другие элементы. Внутренние элементы блоков сами выступают в качестве блоков.

Схематично блочную модель можно представить следующим образом:

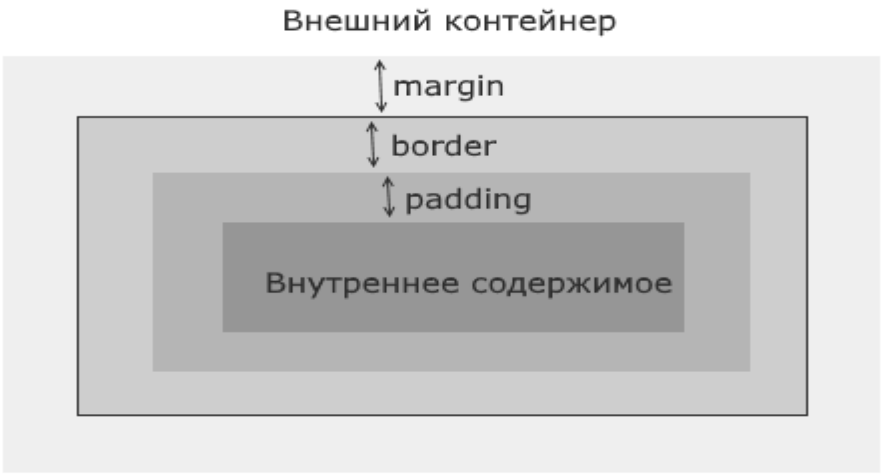

Рис.8.1. Схематическая модель блока.

Пусть элемент расположен в каком-нибудь внешнем контейнере. Это может быть элемент body, div и так далее. От других элементов он отделяется некоторым расстоянием - внешним отступом, которое описывается свойством CSS margin. То есть свойство margin определяет расстояние от границы текущего элемента до других соседних элементов или до границ внешнего контейнера.

Далее начинается сам элемент. И в начале идет его граница, которая в CSS описывается свойством border.

После границы идет внутренний отступ, который в CSS описывается свойством padding. Внутренний отступ определяет расстояние от границы элемента до внутреннего содержимого.

Далее идет внутреннее содержимое, которое также реализует ту же блочную модель и также может состоять из других элементов, которые имеют внешние и внутренние отступы и границу.

Например, определим следующую веб-страницу:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Блочная модель в CSS3</title>
        <style>
            div{
                margin: 15px; /* внешний отступ */
                padding: 11px; /* внутренний отступ */
                border: 3px solid red; /* границы шириной в 3 пикселя сплошной красной линией */
            }
        </style>
    </head>
    <body>
        <div>
            <p>Первый блок</p>
        </div>
        <div>
            <p>Второй блок</p>
        </div>
    </body>
</html>
```

После запуска веб-страницы в браузере мы можем посмотреть блочную модель конкретных элементов. Для этого надо нажать на нужный элемент правой кнопкой мыши и открывающемся контекстном меню выбрать пункт, который позволяет просмотреть исходный код элемента. Для разных браузеров этот пункт может называться по разному. К примеру в Google Chrome это Посмотреть код:

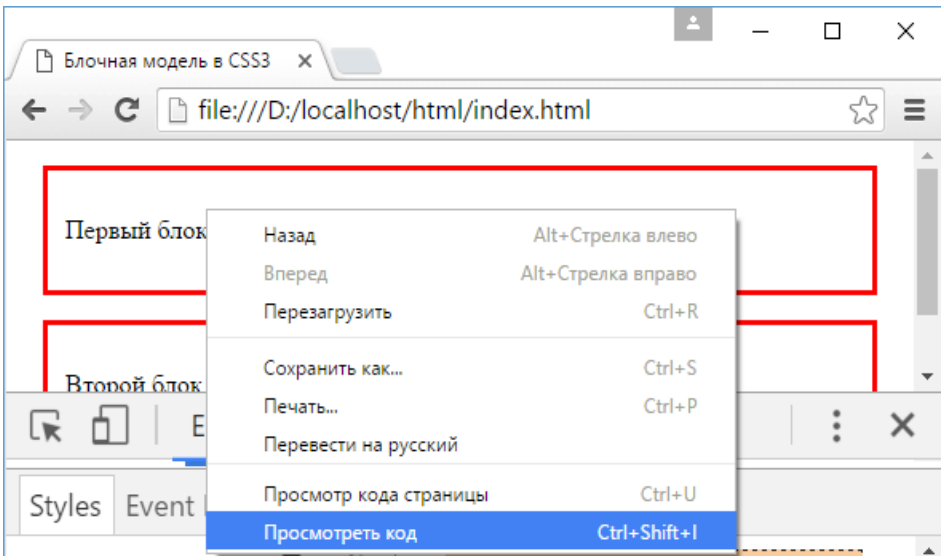

Рис.8.2. В Google Chrome это код просмотра.

И по выбору данного пункта браузер откроет панель, где будет показан код элемента его стили и блочная модель:

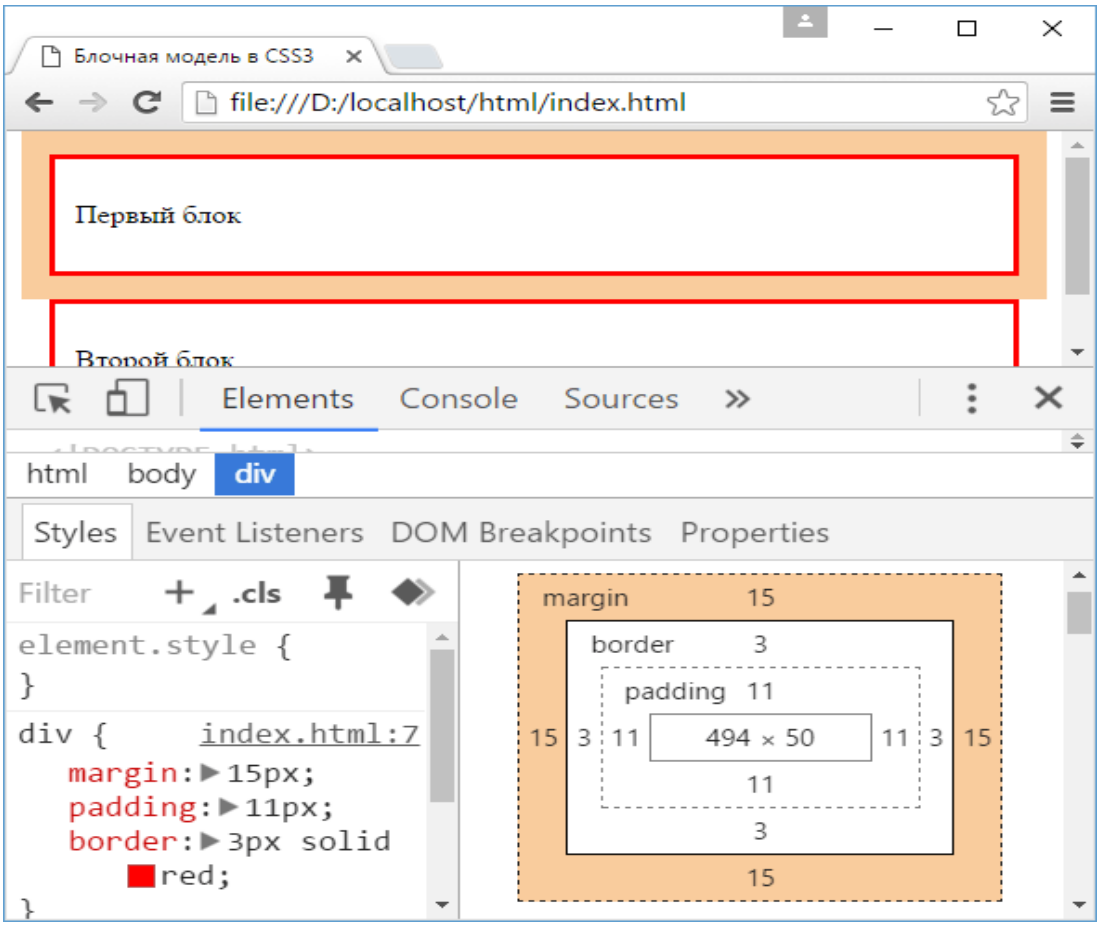

Рис. 8.3. Отображать код элемента, его стили и блочную модель в браузере.

В этой модели мы можем увидеть, как задаются отступы элемента, его граница, посмотреть отступы от других элементов и при необходимости динамически поменять значения их стилей.

Граница отделяети элемент от внешнего по отношению к нему содержимого. При этом граница является частью элемента.

Для настройки границы могут использоваться сразу несколько свойств:

border-width: устанавливает ширину границы

border-style: задает стиль линии границы

border-color: устанавливает цвет границы

Свойство border-width может принимать следующие типы значений:

Значения в единицах измерения, таких как em, px или cm

border-width: 2px;

Одно из константных значений: thin (тонкая граница - 1px), medium (средняя по ширине - 3px), thick (толстая - 5px)

border-width: medium;

Свойство border-color в качестве значения принимает цвет CSS:

border-color: red;

Свойство border-style оформляет тип линии границы и может принимать одно из следующих значений:

none: граница отсутствует

solid: граница в виде обычной линии

dashed: штриховая линия

dotted: линия в виде последовательности точек

double: граница в виде двух параллельных линий

groove: граница имеет трехмерный эффект

inset: граница как бы вдавливается во внутрь

outset: аналогично inset, только граница как бы выступает наружу

ridge: граница также реализует трехмерный эффект

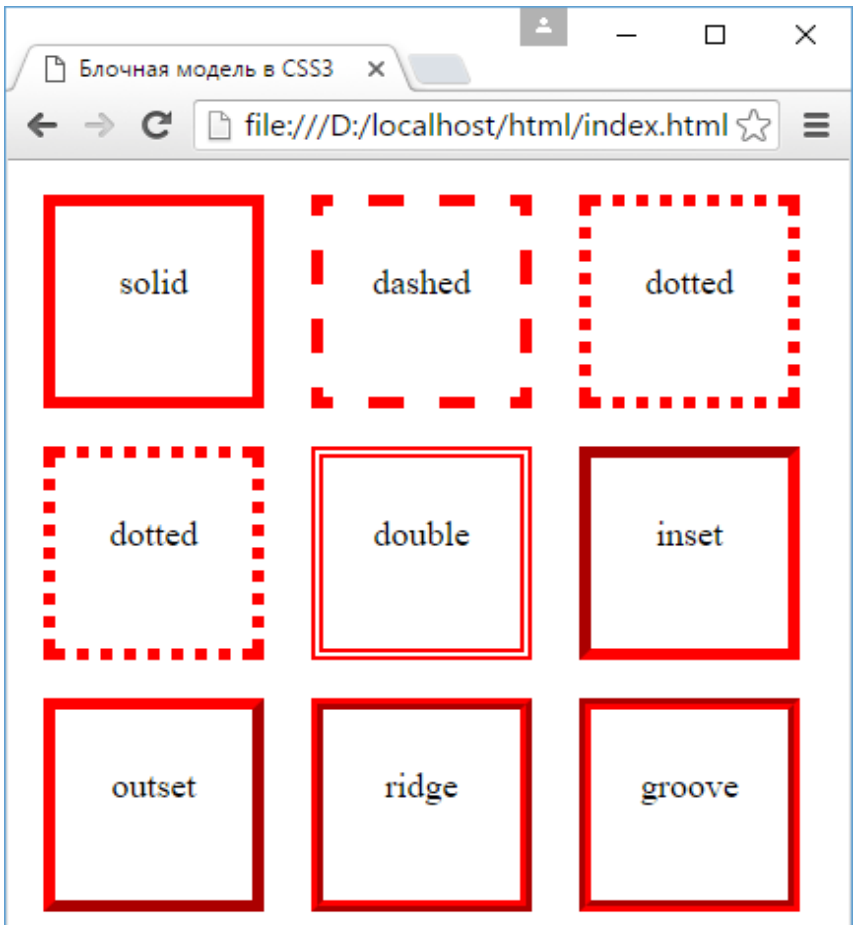

Рис. 8.4. Результаты использования граничных элементов.

Контрольные вопросы.

1. В чем разница между margin и padding?
2. Что означают пограничное положение и пограничное-все?
3. Как установить цвет фона страницы?
4. Как сделать текст абзаца жирным?

ЛАБОРАТОРНАЯ РАБОТА №9

Введение в JavaScript. Работа с операторами языка Java Script.

Цель работы: Приобрести навыки работы с операторами языка Java Script.

Задание: Введение в JavaScript. Работа с операторами языка Java Script.

Методические указания:

Введение в JavaScript.

Сегодняшний мир веб-сайтов трудно представить без языка JavaScript. JavaScript - это то, что делает живыми веб-страницы, которые мы каждый день просматриваем в своем веб-браузере.

JavaScript был создан в 1995 году в компании Netscape разработчиком Брендоном Айком (Brendon Eich) в качестве языка сценариев в браузере Netscape Navigator 2. Первоначально язык назывался LiveScript, но на волне популярности в тот момент другого языка Java LiveScript был переименован в JavaScript. Однако данный момент до сих пор иногда приводит к некоторой путанице: некоторые начинающие разработчики считают, что Java и JavaScript чуть ли не один и тот же язык. Нет, это абсолютно два разных языка, и они связаны только по названию.

Первоначально JavaScript обладал довольно небольшими возможностями. Его цель состояла лишь в том, чтобы добавить немного поведения на веб-страницу. Например, обработать нажатие кнопок на веб-странице, произвести какие-нибудь другие действия, связанные прежде всего с элементами управления.

Однако развитие веб-среды, появление HTML5 и технологии Node.js открыло перед JavaScript гораздо большие горизонты. Сейчас JavaScript продолжает использоваться для создания веб-сайтов, только теперь он предоставляет гораздо больше возможностей.

Также он применяется как язык серверной стороны. То есть, если раньше JavaScript применялся только на веб-странице, а на стороне сервера нам надо было использовать такие технологии, как PHP, ASP.NET, Ruby, Java, то сейчас благодаря Node.js мы можем обрабатывать все запросы к серверу также с помощью JavaScript.

В последнее время переживает бум сфера мобильной разработки. И JavaScript опять же не остается в стороне: увеличение мощности устройств и повсеместное распространение стандарта HTML5 привело к тому, что для создания приложений для смартфонов, планшетов и настольных компьютеров мы также можем использовать JavaScript. То есть JavaScript уже перешагнул границы веб-браузера, которые ему были очерчены при его создании.

И что вообще раньше казалось фантастикой, но сегодня стало реальностью - javascript может использоваться для набирающего популярность направления разработки для IoT(Internet of Things или Интернет вещей). То есть JavaScript можно использовать для программирования самых различных "умных" устройств, которые взаимодействуют с интернетом.

Таким образом, вы можете встретить применение JavaScript практически повсюду. Сегодня это действительно один из самых популярных языков программирования, и его популярность еще будет расти.

С самого начала существовало несколько веб-браузеров (Netscape, Internet Explorer), которые предоставляли различные реализации языка. И чтобы свести различные реализации к общему стержню и стандартизировать язык под руководством организации ECMA был разработан стандарт ECMAScript. В принципе сами термины JavaScript и ECMAScript являются во многом взаимозаменяемыми и относятся к одному и тому же языку.

К настоящему времени ECMA было разработано несколько стандартов языка, которые отражают его развитие. В последнее время почти каждый год выходит новый стандарт. На данный момент последним принятым стандартом является ECMAScript 2021, который был одобрен 22 июня 2021 года. Однако реализация стандартов в браузерах занимает довольно продолжительное время. Одни браузеры быстрее реализуют новые стандарты, другие медленнее. Кроме того, есть большой пласт старых версий браузеров, которыми простые пользователи продолжают пользоваться и которые естественно могут не поддерживать нововведения последних стандартов. И это надо учитывать при разработке программ на JavaScript. В данном же руководстве будут рассматриваться в основном те возможности JavaScript, которые поддерживаются всеми наиболее распространенными современными браузерами.

JavaScript является интерпретируемым языком. Это значит, что код на языке JavaScript выполняется с помощью интерпретатора. Интерпретатор получает инструкции языка JavaScript, которые определены на веб-странице, выполняет их (или интерпретирует).

Средства разработки

Для разработки на JavaScript нам потребуется текстовый редактор для написания кода и веб-браузер для его тестирования. В качестве текстового редактора следует использовать такую программу как Visual Studio Code. Он бесплатен, имеет много возможностей и может быть установлен как на Windows, так и на Linux и MacOS. Хотя этот может быть любой другой текстовый редактор.

Также существуют различные среды разработки, которые поддерживают JavaScript и облегчают разработку на этом языке,

например, Visual Studio, WebStorm, Netbeans и так далее. При желании можно использовать также эти среды разработки.

Итак, приступим к созданию первой программы.

Первая программа на JavaScript

Создадим первую программу на javascript. Для написания и тестирования программ на JavaScript нам потребуются две вещи: тестовый редактор и веб-браузер.

В качестве текстового редактора можно взять любой, который нравится - Atom, Sublime Text, Visual Studio Code, Notepad++ и другие. В данном руководстве будем ориентироваться на текстовый редактор Visual Studio Code, поскольку он является наиболее популярным.

В качестве браузера также можно взять последние версии любого предпочтительного веб-браузера. В настоящем руководстве будем преимущественно ориентироваться на Google Chrome.

Для начала определим для нашего приложения какой-нибудь каталог. Например, создадим на диске C папку app. В этой папке создадим файл под названием index.html. То есть данный файл будет представлять веб-страницу с кодом HTML.

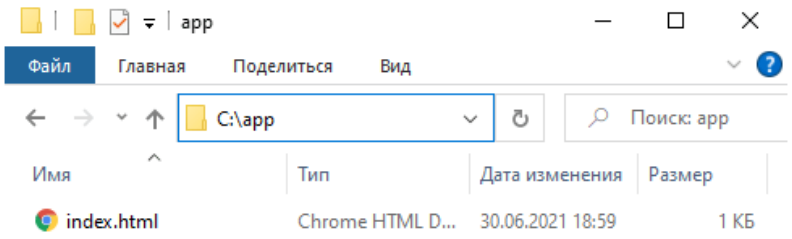

Рис.9.1. Создайте файл с именем index.html в папке.

Откроем этот файл в текстовом редакторе и определим в файле следующий код:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>METANIT.COM</title>
</head>
<body>
    <script>
```

```
        document.write("<h2>Первая программа на JavaScript</h2>");
    </script>
</body>
</html>
```

Здесь мы определяем стандартные элементы html. В элементе *head* определяется кодировка utf-8 и заголовок (элемент title). В элементе *body* определяется тело веб-страницы, которое в данном случае состоит только из одного элемента <*script*>

Подключение кода javascript на html-страницу осуществляется с помощью тега <*script*>. Данный тег следует размещать либо в заголовке (между тегами <head> и </head>), либо в теле веб-странице (между тегами <body> и </body>). Нередко подключение скриптов происходит перед закрывающим тегом </body> для оптимизации загрузки веб-страницы.

Раньше надо было в теге <script> указывать тип скрипта, так как данный тег может использоваться не только для подключения инструкций javascript, но и для других целей. Так, даже сейчас вы можете встретить на некоторых веб-страницах такое определение элемента script:

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

Но в настоящее время предпочтительнее опускать атрибут type, так как браузеры по умолчанию считают, что элемент script содержит инструкции javascript.

Используемый нами код javascript содержит одно выражение:

```
document.write("<h2>Первая программа на JavaScript</h2>");
```

Код javascript может содержать множество инструкций и каждая инструкция завершается точкой с запятой. Наша инструкция вызывает метод document.write(), который выводит на веб-страницу

некоторое содержимое, в данном случае это заголовок <h2>Первая программа на JavaScript</h2>.

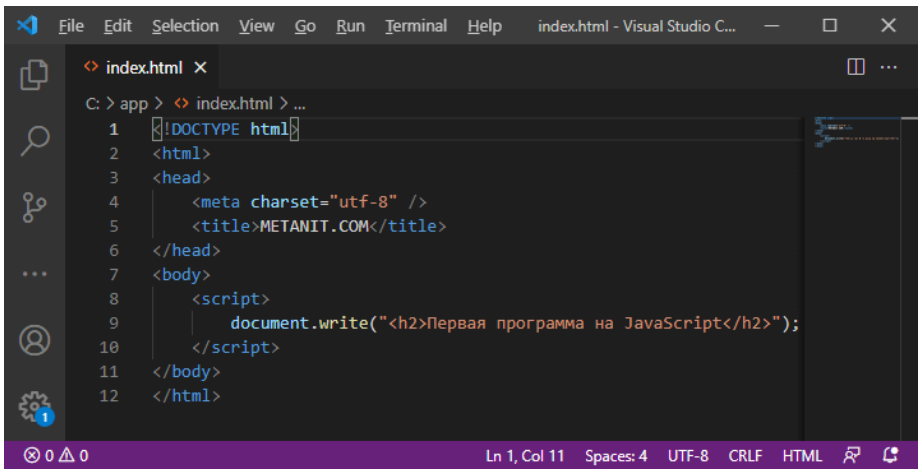

Рис.9.2. Вид файла в текстовом редакторе Visual Studio Code.

Теперь, когда веб-страница готова, откроем ее в веб-браузере:

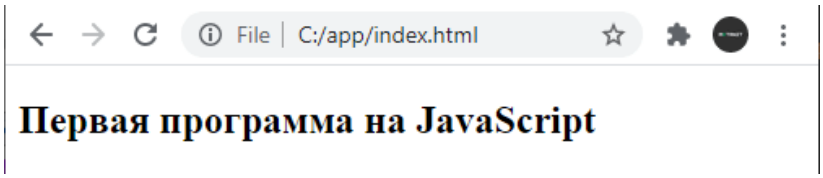

Рис.9.3. Открыть в веб-браузере.

Контрольные вопросы.

1. Что такое JavaScript?
2. Что такое JavaScript простыми словами?
3. Для чего используют JavaScript?
4. Какие сайты написаны на JavaScript?
5. Что дает знание JavaScript?

ЛАБОРАТОРНАЯ РАБОТА №10

Работа с операторами языка Java Script

Цель работы: Приобрести навыки работы с операторами языка JavaScript.

Задание: Введение в JavaScript. Работа с операторами языка Java Script.

Методические указания:

Выполнение кода JavaScript.

Когда браузер получает веб-страницу с кодом html и javascript, то он ее интерпретирует. Результат интерпретации в виде различных

элементов - кнопок, полей ввода, текстовых блоков и т.д., мы видим перед собой в браузере. Интерпретация веб-страницы происходит последовательно сверху вниз.

Когда браузер встречает на веб-странице элемент <script> с кодом javascript, то вступает в действие встроенный интерпретатор javascript. И пока он не закончит свою работу, дальше интерпретация веб-страницы не идет.

Рассмотрим небольшой пример и для этого изменим страницу index.html из прошлой темы следующим образом:

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8" />
   <title>METANIT.COM</title>
   <script>
      document.write("Начальный текст");
   </script>
</head>
<body>
   <h2>Первый заголовок</h2>
   <script>
      document.write("Первый текст");
   </script>
   <h2>Второй заголовок</h2>
   <script>
      document.write("Второй текст");
   </script>
</body>
</html>
```

Здесь три вставки кода javascript - один в секции <head> и по одному после каждого заголовка.

Откроем веб-страницу в браузере и мы увидим, что браузер последовательно выполняет код веб-страницы:

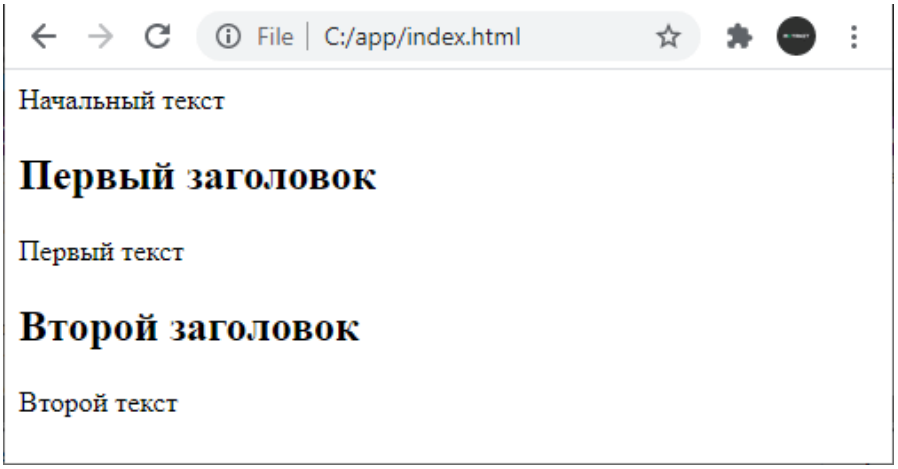

Рис.10.1. Просмотр веб-страницы в браузере.

Здесь мы видим, что вначале выполняется код javascript из секции head, который выводит на веб-страницу некоторый текст:

document.write("Начальный текст");

Далее выводится первый стандартный html-элемент <h2>:

<h2>Первый заголовок</h2>

После этого выполняется вторая вставка кода на javascript:

document.write("Первый текст");

Затем выводится второй html-элемент <h2>:

<h2>Второй заголовок</h2>

И в конце выполняется последняя вставка кода на javascript:

document.write("Второй текст");

document.write("Начальный текст");

Далее выводится первый стандартный html-элемент <h2>:

document.write("Второй текст");

После этого браузер закончит интерпретацию веб-страницы, и веб-страница окажется полностью загружена. Данный момент очень важен, поскольку может влиять на производительность. Поэтому нередко вставки кода javascript идут перед закрывающим тегом *</body>*, когда основная часть веб-страницы уже загружена в браузере.

Основы синтаксиса javascript

Прежде чем переходить к детальному изучению основ языка программирования javascript, рассмотрим некоторые базовые моменты его синтаксиса.

Код javascript состоит из инструкций. Каждая инструкция представляет некоторое действие. И для отделения инструкций друг от друга в javascript после инструкции ставится точка с запятой:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>METANIT.COM</title>
</head>
<body>
    <script>
        document.write("2 + 5 = "); var sum = 2 + 5; document.write(sum);
    </script>
</body>
</html>
```

Здесь в коде javascript определены три инструкции:

document.write("2 + 5 = ")

Выводит на страницу текст "2 + 5 = "

var sum = 2 + 5;

С помощью оператора var определяет переменную sum, которая равна сумме 2 + 5

document.write(sum);

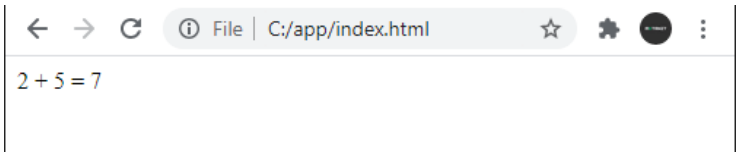

Рис.10.2. Выводит на страницу значение переменной sum (то есть суму 2 + 5)

Консоль браузера и console.log

Незаменимым инструментом при работе с JavaScript является консоль браузера, которая позволяет производить отладку программы. Во многих современных браузерах есть подобная консоль. Например, чтобы открыть консоль в Google Chrome, нам надо перейти в меню Дополнительные инструменты *(More Tools)* -> Инструменты разработчика *(Developer tools):*

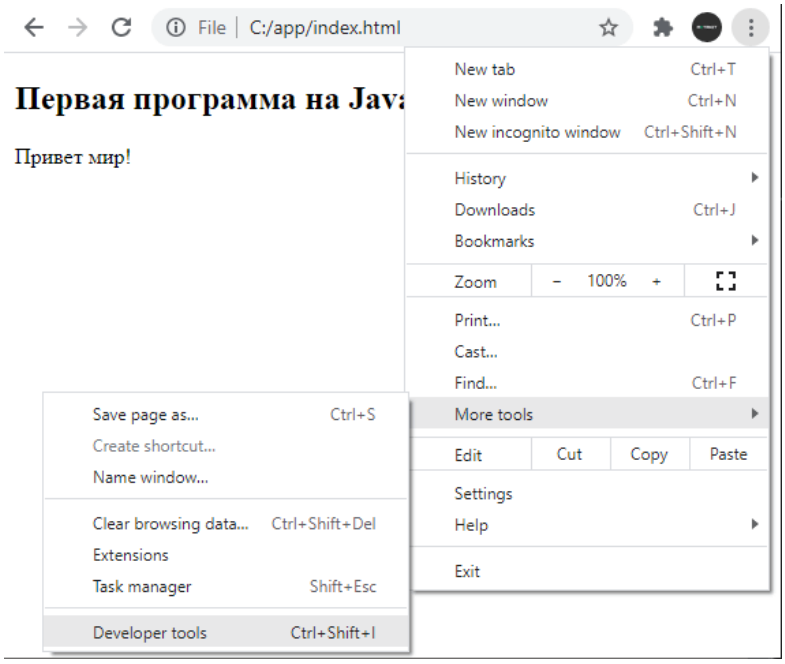

Рис.10.3. Инструменты разработчика *(Developer tools)*.

После этого внизу браузера откроется консоль:

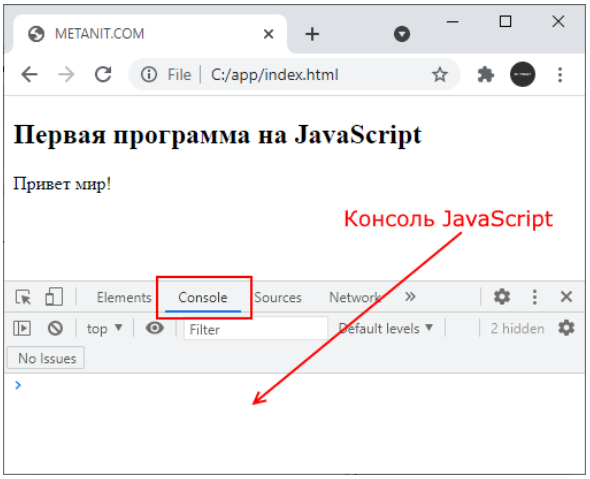

Рис.10.4. Консоль браузера.

Мы можем напрямую вводить в консоль браузера выражения JavaScript, и они будут выполняться. Например, введем в консоли следующий текст:

alert("Привет мир");

Функция alert() выводит в браузере окно с сообщением. В итоге после ввода этой команды и нажатия на клавишу Enter браузер выполнит эту функцию и отобразит нам окно с сообщением:

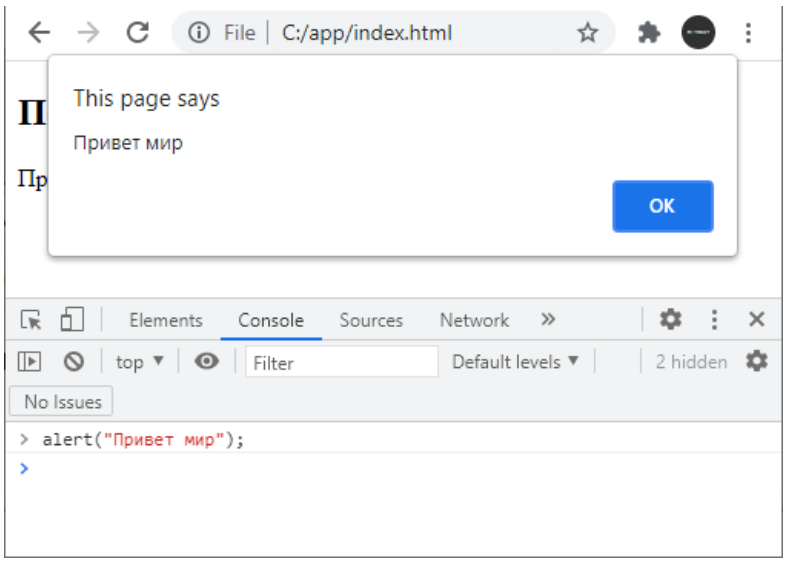

Рис.10.5. Окно сообщения в браузере.

Для вывода различного рода информации в консоли браузера используется специальная функция *console.log*(). Например, определим следующую веб-страницу:

```
<!DOCTYPE html>
<html>
        <head>
                <meta charset="utf-8" >
                <title>METANIT.COM</title>
        </head>
        <body>
    <script>
        let a = 5 + 8;
        console.log("Результат операции");
        console.log(a);
    </script>
        </body>
</html>
```

В коде javascript с помощью ключевого слова let объявляется переменная a, которой присваивается сумма двух чисел 5 и 8:

```
let a = 5 + 8;
```

Далее с помощью метода console.log() на консоль браузера выводится некоторое сообщение

```
console.log("Результат операции");
```

И в конце также с помощью метода console.log() на консоль браузера выводится значение переменной a.

console.log(a);

И после запуска веб-страницы в браузере мы увидим в консоли результат выполнения кода:

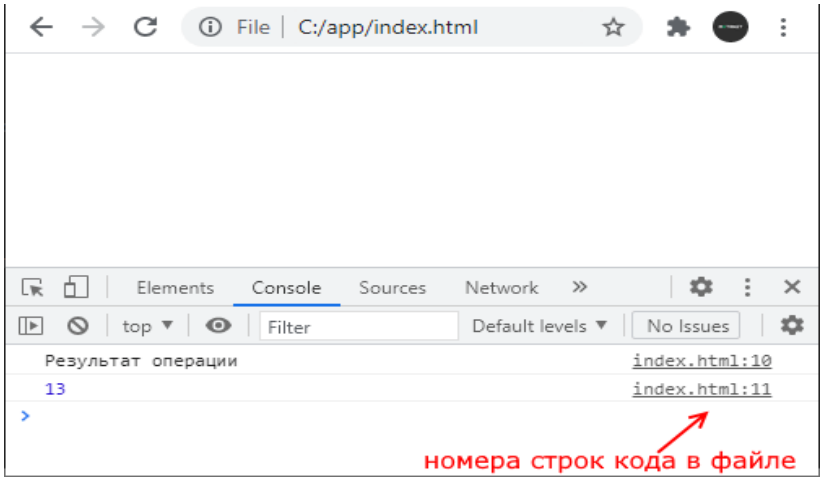

Рис.10.6. Результат выполнения кода.

Что очень полезно, в консоли браузера вы также можете заметить номера строк кода, где именно выполнялся вывод на консоль.

В дальнейшем мы часто будем обращаться к консоли и использовать функцию console.log.

Причем подобный код мы могли бы ввести в самой консоли:

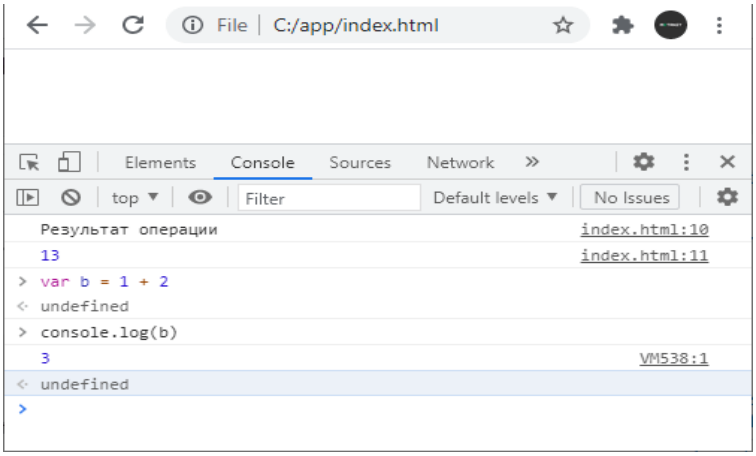

Рис.10.7. Код можно ввести в самой консоли.

Также последовательно вводим инструкции и после ввода каждой инструкции нажимаем на Enter.

Если нам надо, чтобы код в консоли переносился на новую строку без выполнения, то в конце выражения javascript нажимаем на

комбинацию клавиш *Shift + Enter*. После ввода последней инструкции для выполнения введенного кода javascript нажимаем на Enter.

Контрольные вопросы.

1. Для чего используется оператор "&&"?
2. Для чего используется оператор "||"?
3. Является ли использование унарного плюса (оператор "+") самым быстрым способом преобразования строки в число?
4. Какие сайты написаны на JavaScript?
5. Что дает знание JavaScript?

ЛАБОРАТОРНАЯ РАБОТА №11

Типы данных и математические операции в Java Script

Цель работы: Приобрести навыки работы с типами данных и математическими операциями в JavaScript.

Задание: Работа с типами данных и математическими операциями в Java-скрипте

Методические указания:

Типы данных

Все используемые данные в javascript имеют определенный тип. В JavaScript имеется восемь типов данных:

String: представляет строку

Number: представляет числовое значение

BigInt: предназначен для представления очень больших целых чисел

Boolean: представляет логическое значение true или false

Undefined: представляет одно специальное значение - undefined и указывает, что значение не установлено

Null: представляет одно специальное значение - null и указывает на отсутствие значения

Symbol: представляет уникальное значение, которое часто применяется для обращения к свойствам сложных объектов

Object: представляет комплексный объект

Первые семь типов представляют примитивные типы данных. Последний тип - Object представляет сложный, комплексный тип данных, который состоит из значений примитивных типов или других объектов.

Number

Тип Number представляет числа в JavaScript, которые могут быть целыми или дробными:

Например:

```
let x = 45;
let y = 23.897;
```

В качестве разделителя между целой и дробной частями, как и в других языках программирования, используется точка.

Тип BigInt добавлен в последних стандартах JavaScript для представления очень больших целых чисел, которые выходят за пределы диапазона типа number. Это не значит, что мы не можем совсем работать с большими числами с помощью типа number, но работа с ними в случае с типом number будет сопряжена с проблемами. Рассмотрим небольшой пример:

```
let num = 9007199254740991
console.log(num);       // 9007199254740991
console.log(num + 1);   // 9007199254740992
console.log(num + 2);   // 9007199254740992
```

Здесь переменной num присваивается максимальное значение. И далее прибавляем к ней некоторые значения и выводим на консоль результат. И результаты могут нас смутить, особенно в случае прибавления числа 2.

Для определения числа как значения типа BigInt в конце числа добавляется суффикс n:

```
let dimension = 19007n;
const value = 2545n;
```

Например, изменим из предыдущего примера тип number на bigint:

```
let num = 9007199254740991n
console.log(num);       // 9007199254740991n
console.log(num + 1n);  // 9007199254740992n
console.log(num + 2n);  // 9007199254740993n
console.log(num + 3n);  // 9007199254740994n
```

Тип Boolean

Тип Boolean представляет булевые или логические значения true (верно) и false (ложно):

```
let isAlive = true;
let isDead = false;
```

Строки String

Тип String представляет строки. Для определения строк применяются кавычки, причем, можно использовать как двойные, так одинарные, так и косые кавычки. Единственно ограничение: тип закрывающей кавычки должен быть тот же, что и тип открывающей, то есть либо обе двойные, либо обе одинарные.

Математические операции.

JavaScript поддерживает все базовые математические операции:

Сложение:

```
let x = 10;
let y = x + 50;
```

Вычитание:

```
let x = 100;
let y = x - 50;
```

Умножение:

```
let x = 4;
let y = 5;
let z = x * y;
```

Деление:

```
let x = 5;
let y = 2;
let z = x / y;
console.log(z); // 2.5
```

Деление по модулю (оператор %) возвращает остаток от деления:

```
let x = 5;
let y = 2;
let z = x % y;
console.log(z); // 1
```

Результатом будет 1, так как наибольшее целое число, которое меньше или равно 5 и при этом делится на 2 равно 4, а 5 - 4 = 1.

Возведение в степень. Оператор ** возводит число в определенную степень:

```
число ** степень
```

Например:

```
const n = 2 ** 3;
console.log(n);  // 8
const x = 3;
const y = 2;
const z = x ** y;
```

```
console.log(z);  // 9
```
Инкремент:
```
let x = 5;
x++; // x = 6
```
Оператор инкремента ++ увеличивает переменную на единицу. Существует префиксный инкремент, который сначала увеличивает переменную на единицу, а затем возвращает ее значение. И есть постфиксный инкремент, который сначала возвращает значение переменной, а затем увеличивает его на единицу:
```
// префиксный инкремент
let x = 5;
let z = ++x;
console.log(x); // 6
console.log(z); // 6
// постфиксный инкремент
let a = 5;
let b = a++;
console.log(a); // 6
console.log(b); // 5
```
Постфиксный инкремент аналогичен операции:
```
a = a + 1; // a++
```
Декремент уменьшает значение переменной на единицу. Также есть префиксный и постфиксный декремент:
```
// префиксный декремент
let x = 5;
let z = --x;
console.log(x); // 4
console.log(z); // 4
// постфиксный декремент
let a = 5;
let b = a--;
console.log(a); // 4
console.log(b); // 5
```
Как и принято в математике, все операции выполняются слева направо и различаются по приоритетам: сначала операции инкремента и декремента, затем выполняются умножение и деление, а потом сложение и вычитание. Чтобы изменить стандартный ход выполнения операций, часть выражений можно поместить в скобки:
```
let x = 10;
```

```
let y = 5 + (6 - 2) * --x;
console.log(y); //41
```

Контрольные вопросы.

1. Для чего используется оператор "&&"?
2. Для чего используется оператор "||"?
3. Является ли использование унарного плюса (оператор "+") самым быстрым способом преобразования строки в число?
4. Какие сайты написаны на JavaScript?
5. Что дает знание JavaScript?

ЛАБОРАТОРНАЯ РАБОТА №12

Условные конструкции и циклы в JavaScript

Цель работы: Приобрести навыки работы с условиями и циклами в JavaScript.

Задание: Работа с условиями и циклами в JavaScript и выполнение задач с их использованием.

Методические указания:

Условные конструкции

Условные конструкции позволяют выполнить те или иные действия в зависимости от определенных условий.

Конструкция if..else

Конструкция if..else проверяет некоторое условие и если это условие верно, то выполняет некоторые действия. Простейшая форма конструкции if..else:

```
if(условие){
    некоторые действия
}
```

После ключевого слова if в круглых скобках идет условие, а после условия - блок кода с некоторыми действиями. Если это условие истинно, то затем выполняются действия, которые помещены в блоке кода.Например:

```
const income = 100;
if(income > 50) {
    console.log("доход больше 50");
}
```

Здесь в конструкции if используется следующее условие: income > 50. Если это условие возвращает true, то есть если константа income

имеет значение больше 50, то браузер отображает сообщение. Если же значение income меньше 50, то никакого сообщения не отображается.

Если блок кода содержит одну инструкцию, как в случае выше, то конструкцию можно упростить, убрав фигурные скобки и поместив действия сразу после условия:

```
const income = 100;
if(income > 50) console.log("доход больше 50");
```

или перенести действия на следующую строку

```
const income = 100;
if(income > 50)
   console.log("доход больше 50");
```

Причем условия могут быть сложными:

```
const income = 100;
const age = 19;
if(income > 50 && age > 18){
 console.log("доход больше 50");
 console.log("возраст больше 18");
}
```

Проверка наличия значения

Конструкция if позволяет проверить наличие значения. Например:

```
let myVar = 89;
if(myVar){
  console.log(`Переменная myVar имеет значение: ${myVar}`);
}
```

Если переменная myVar имеет значение, как в данном случае, то в условной конструкции она возвратит значение true.

Противоположный вариант:

```
let myVar;
if(myVar){
   console.log(`Переменная myVar имеет значение: ${myVar}`);
}
```

Здесь переменная myVar не имеет значения. (В реальности она равна undefined) Поэтому условие в конструкии if возвратит false, и действия в блоке конструкции if не будут выполняться.

Но нередко для проверки значения переменной используют альтернативный вариант - проверяют на значение undefined и null:

```
if (myVar !== undefined && myVar !== null) {
   console.log(`Переменная myVar имеет значение: ${myVar}`);
```

```
}
```

Выражение else

Выше мы рассмотрели, как определить действия, которые выполняются, если условие после if истинно. Но что, если мы хотим также выполнять еще один набор инструкций, если условие ложно? В этом случае можно использовать блок else. Данный блок содержит инструкции, которые выполняются, если условие после if ложно, то есть равно false:

```
if(условие){
   действия, если условие истинно
}
else{
   действия, если условие ложно
}
```

То есть, если условие после if истинно, выполняется блок if. Если условие ложно, выполняется блок else. Например:

```
const income = 45;
if(income > 50){
 console.log("Доход больше 50");
}
else{
   console.log("Доход меньше или равен 50");
}
```

Здесь константа income равна 45, поэтому условие после оператора if возвратит false, и управление перейдет к блоку else.

Также если блок else содержит одну инструкцию, то можно сократить конструкцию:

```
const income = 45;
if(income > 50) console.log("Доход больше 50");
else console.log("Доход меньше или равен 50");
```

Альтернативные условия и else if

С помощью конструкции else if мы можем добавить альтернативное условие к блоку if. Например, выше в условие значение income может быть больше определенному значению может быть меньше, а может быть равно ему.

```
const income = 50;
if(income > 50) {
   console.log("Доход больше 50");
}
```

```
else if(income === 50){
    console.log("Доход равен 50");
}
else{
    console.log("Доход меньше 50");
}
```

В данном случае выполнится блок else if. При необходимости мы можем использовать несколько блоков else if с разными условиями:

```
const income = 500;
if(income < 200){
 console.log("Доход ниже среднего");
}
else if(income>=200 && income<300){
   console.log("Чуть ниже среднего");
}
else if(income>=300 && income<400){
    console.log("Средний доход");
}
else{
    console.log("Доход выше среднего");
}
```

True или *false*

В javascript любая переменная может применяться в условных выражениях, но не любая переменная представляет тип boolean. И в этой связи возникает вопрос, что возвратит та или иная переменная - true или false? Много зависит от типа данных, который представляет переменная:

undefined

Возвращает false

null

Возвращает false

Если константа/переменная равна false, то возвращается false. Соответствено если константа/переменная равна true, то возвращается true

Возвращает false, если число равно 0 или NaN (Not a Number), в остальных случаях возвращается true

Например, следующая переменная будет возращать false:

```
let x = NaN;
if(x){  // false
```

```
 }
```

String

Возвращает false, если константа/переменная равна пустой строке, то есть ее длина равна 0, в остальных случаях возвращается true

```
const emptyText = "";   // false - так как пустая строка
const someText = "javascript";  // true - строка не пустая
```

Object

Всегда возвращает true

```
const user = {name:"Tom"};  // true
const car = {}; // true
```

Конструкция switch..case является альтернативой использованию конструкции if..else и также позволяет обработать сразу несколько условий:

```
const income = 300;
switch(income){
  case 100 :
        console.log("Доход равен 100");
        break;
    case 200 :
        console.log("Доход равен 200");
        break;
    case 300 :
        console.log("Доход равен 300");
        break;}
```

После ключевого слова switch в скобках идет сравниваемое выражение. Значение этого выражения последовательно сравнивается со значениями, помещенными после оператора case. И если совпадение будет найдено, то будет выполняться определенный блок case.

В конце каждого блока case ставится оператор break, чтобы избежать выполнения других блоков.

Однако при необходимости можно сразу обработать несколько условий:

```
const income = 200;
switch(income){
   case 100 :
    case 200 :
        console.log("Доход равен 100 или 200");
        break;
```

```
    case 300 :
        console.log("Доход равен 300");
        break;
}
```

В данном случае для условия, когда income равно 100 и 200, выполняются одни и те же действия.Если мы хотим также обработать ситуацию, когда совпадения не будет найдено, то можно добавить необязательный блок default:

```
const income = 300;
switch(income){
  case 100 :
        console.log("Доход равен 100");
        break;
    case 200 :
        console.log("Доход равен 200");
        break;
    case 300 :
        console.log("Доход равен 300");
        break;
    default:
        console.log("Доход неизвестной величины");
        break;
```

Контрольные вопросы.

1. Что такое примитивные типы данных в JavaScript?
2. Как рассчитать числа Фибоначчи в JavaScript?
3. Для чего используются операторы break и continue в JavaScript?
4. Каких известных людей из мира JS знаете?
5. Сравните ключевые слова var, let, const.

ЛАБОРАТОРНАЯ РАБОТА №13

Введение в PHP. Общий обзор языка программирования PHP

Цель работы: Введение в PHP. Узнайте о языке программирования PHP.

Задание: Чтобы иметь общую информацию о языке программирования PHP.

Методические указания:

Введение в PHP. Общий обзор языка программирования PHP.

На сегодняшний день PHP является одним из наиболее распространенных языков веб-программирования. Подавляющие большинство сайтов и веб-сервисов в интернете написано с помощью PHP. По некоторым оценкам PHP применяется более чем на 80% сайтов, среди которых такие сервисы, как facebook.com, vk.com, baidu.com и другие. И такая популярность неудивительна. Простота языка позволяет быстро и легко создавать сайты и порталы различной сложности.

PHP был создан в 1994 году датским программистом Расмусом Лердорфом и изначально представлял собой набор скриптов на другом языке Perl. Позже этот набор скриптов был переписан в интерпретатор на языке Си. И с самого возникновения PHP (сокращение от PHP: Hypertext Preprocessor - PHP: Препроцессор гипертекста) представлял удобный набор инструментов для упрощенного создания веб-сайтов и веб-приложений.

Какие преимущества предоставляет PHP?

Для всех наиболее распространенных операционных системам (Windows, MacOS, Linux) есть свои версии пакетов разработки на PHP, а это значит, что вы можете создавать веб-сайты на любой из этих операционных систем.

PHP может работать в связке с различными веб-серверами: Apache, Nginx, IIS

Простота и легкость освоения. Как правило, уже имея небольшой опыт в программировании на PHP, можно создавать простенькие веб-сайты

PHP похож на язык Си, поэтому, зная Си или один из языков с сиподобным синтаксисом, будет проще овладеть PHP

PHP поддерживает работу с множеством систем баз данных (MySQL, MSSQL, Oracle, Postgre, MongoDB и другие)

Распространенность хостинговых услуг и их дешевизна. Так как, как правило, хостинговые компании размещают веб-сайты на PHP на веб-серверах Apache или Nginx, которые работают на одной из операционных систем семейства Linux. И веб-серверы, и операционные системы на базе Linux бесплатны, что снижает общую стоимость использования хостинга

Постоянное развитие. PHP продолжает развиваться, выходят все новые версии, которые несут новые функции, адаптируя язык программирования к новым вызовам. И, как правило, перейти на новую версию не составляет труда.

К настоящему моменту (декабрь 2022) текущей стабильной версией PHP является PHP 8.2

А теперь установим все необходимые инструменты, которые потребуются для программирования на PHP.

Установка PHP

Есть разные способы установки всего необходимого программного обеспечения. Мы можем устанавливать компоненты по отдельности, а можем использовать уже готовые сборки на подобие Denwer или EasyPHP. В подобных сборках компоненты уже имеют начальную настройку и уже готовы для создания сайтов. Однако рано или поздно разработчикам все равно приходится прибегать к установке и конфигурации отдельных компонентов, подключения других модулей. Поэтому мы будем устанавливать все компоненты по отдельности. В качестве операционной системы будет использоваться Windows.

Что подразумевает установка PHP? Во-первых, нам нужен интерпретатор PHP. Во-вторых, необходим веб-сервер, например, Apache, с помощью которого мы сможем обращаться к ресурсам создаваемого нами сайта.

Для установки PHP перейдем на офсайт разработчиков https://www.php.net/downloads. На странице загрузок мы можем найти различные дистрибутивы для операционной системы Linux. Если нашей операционной системой является Windows, то нам надо загрузить один из пакетов со страницы https://windows.php.net/download.

Загрузим zip-пакет последнего выпуска PHP, учитывая разрядность операционной системы, на которую надо установить PHP. Для 64-х разрядной:

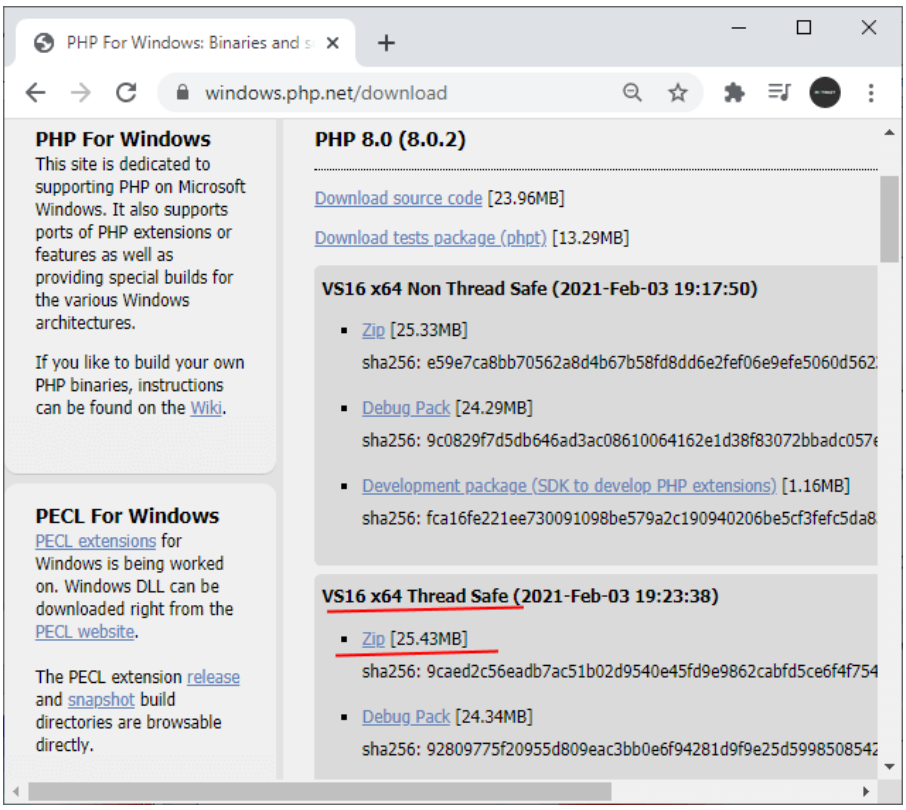

Рис.12.1. Загрузите zip-пакет последней версии PHP.

Интерпретатор PHP имеет две версии: Non Thread Safe и Thread Safe. В чем разница между ними? Версия Thread Safe позволяет задействовать многопоточность, тогда как Non Thread Safe - однопоточная версия. Выберем версию Thread Safe.

Распакуем загруженный архив в папку, которую назовем php. Пусть эта папка у нас будет располагаться в корне диска C.

Теперь нам надо выполнить минимальную конфигурацию PHP. Для этого зайдем в распакованный архив и найдем там файл php.ini-development.

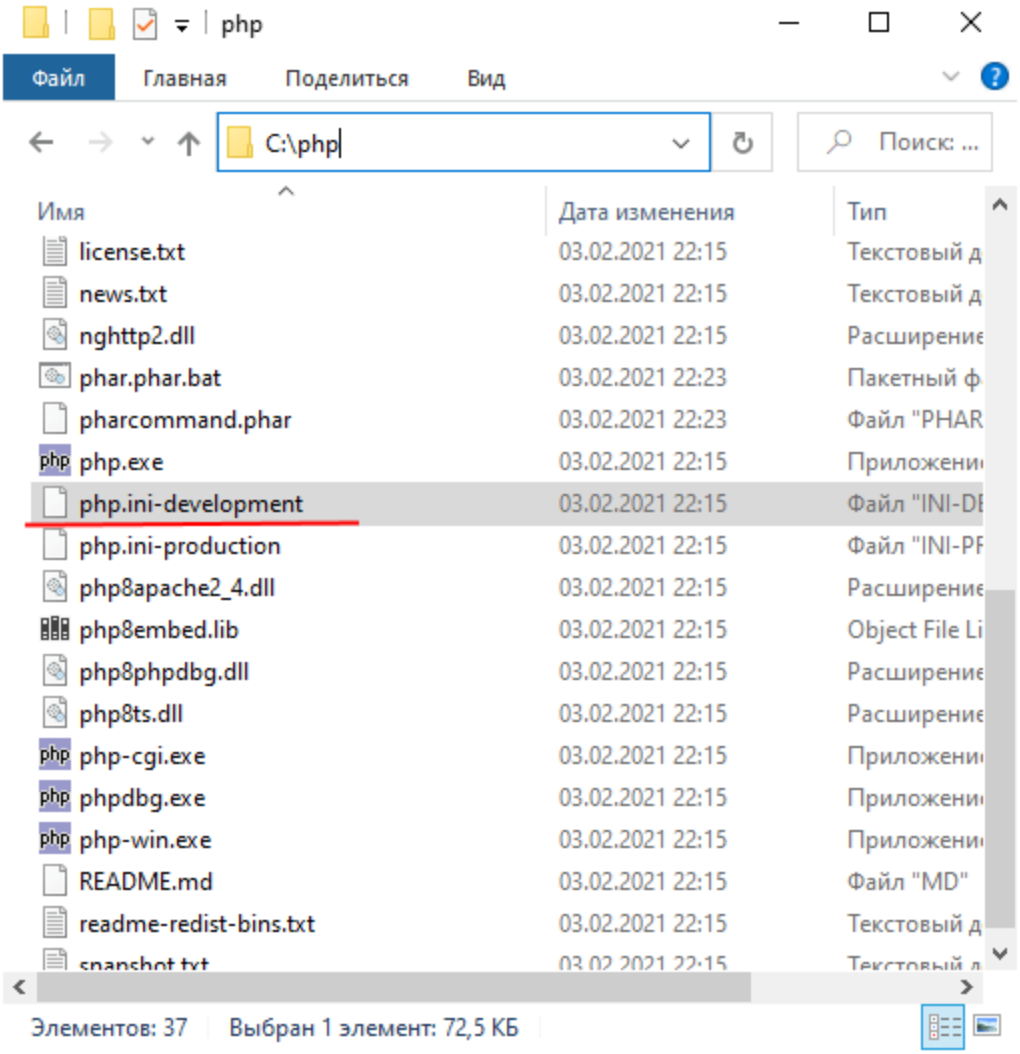

Рис.12.2. файл php.ini-development.

Это файл начальной конфигурации интерпретатора. Переименуем этот файл в php.ini и затем откроем его в текстовом редакторе.

Найдем в файле строку:

;extension_dir = "ext"

Эта строка указывает на каталог с подключаемыми расширениями для PHP. Расширения позволяют задействовать нам некоторую дополнительную функциональность, например, работу с базой данных. Все расширения находятся в распакованном каталоге ext.

Раскомментируем эту строку, убрав точку с запятой и укажем полный путь к папке расширений php:

extension_dir = "C:\php\ext"

Остальное содержимое файла оставим без изменений.

Теперь установим веб-сервер.

Контрольные вопросы.

1. Что такое ссылки?
2. Каковы основные операции с использованием ссылок?
3. Назовите простые типы данных, поддерживаемые в PHP.
4. Что такое инкремент и декремент, в чем разница между префиксным и постфиксным инкрементом и декрементом?
5. Что такое рекурсия?
6. В чем разница между =, == и ===?
7. Какие знаете принципы ООП?

ЛАБОРАТОРНАЯ РАБОТА №14

Основы PHP. Основы синтаксиса. Встраивание кода PHP.

Цель работы: Приобрести навыки кодирования на PHP. Основы синтаксиса.

Задание: Написание кода и создание веб-страниц на PHP. Основы синтаксиса.

Методические указания:

Основы PHP. Основы синтаксиса.

Встраивание кода PHP.

При создании первой программы на PHP уже были затронуты некоторые основные принципы создания скриптов на языке PHP. Теперь рассмотрим их более подробно.

Программа или скрипт на PHP, как правило, находится в файле расширением .php. Хотя разработчики могут также вставлять код php и в файлы с расширениями .html/.htm.

Документ PHP может содержать как разметку html, так и код на языке php. Для перехода от разметки html к коду php используются теги <?php и ?>, между которыми идет код php. Данные теги служат указанием интерпретатору PHP, что их содержимое надо интерпретировать как код php, а не разметку html.

Например, определим в папке, где хранятся файлы веб-сайта (исходя из прошлых тем это должна быть папка c:\localhost), файл hello.php. Определим в этом файле следующий код:

```
<?php
  echo "Привет мир!";
?>
```

Собственно код PHP состоит из набора инструкций. Здесь использована одна инструкция echo "Привет мир!";. Эта инструкция

представляет встроенную команду echo, которая выводит на веб-страницу некоторое значение. Выводимое значение указывается после команды echo - в данном случае это строка "Привет мир!". Каждая отдельная инструкция в PHP завершается точкой с запятой.

Поскольку в данном случае файл "hello.php" находится в корневой папке веб-сервера, то для обращения к этому скрипту в адресной строке браузера надо ввести адрес http://localhost/hello.php. В итоге при обращении к этому скрипту мы увидим в веб-браузере следующую страницу:

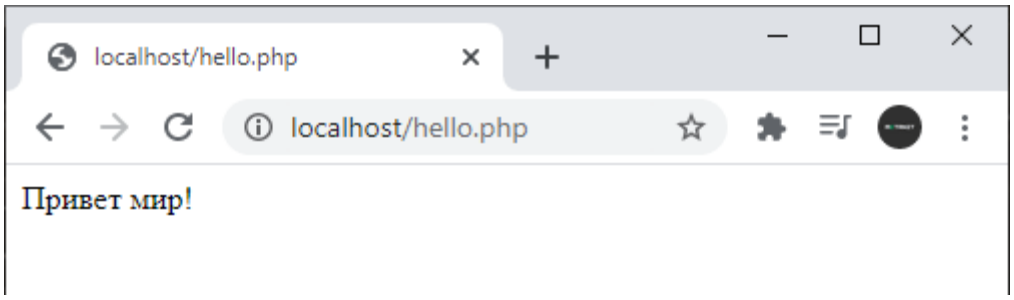

Рис. 14.1. Результат echo "Привет, мир!"

Для оформления кода PHP также можно использовать краткую версию тегов: <? и ?>. Для этого в файле php.ini надо изменить строку:

short_open_tag = Off

на

short_open_tag = On

Когда пользователь обращается к скрипту в адресной строке браузера, набирая, например, http://localhost/hello.php, то веб-сервер передает его интерпретатору PHP. Затем интерпретатор обрабатывает код и генерирует на его основе html-разметку. И затем сгенерированный html-код отправляется пользователю.

В случае выше определенного скрипта hello.php сгенерированная разметка будет выглядеть следующим образом:

Привет мир!

Но естественно мы можем также добавить некоторый код html. Например, изменим скрипт hello.php следующим образом:

```
<!DOCTYPE html>
<html>
  <head>
```

```
        <title>METANIT.COM</title>
        <meta charset="utf-8" />
    </head>
    <body>
        <h1>Сайт на PHP</h1>
<?php
echo "Привет мир!";
?>
    </body>
</html>
```

Как уже указывалось выше, интерпретатор с помощью тегов <?php ... ?> сможет понять, что весь текст между этими тегами следует рассматривать как код PHP. А весь код вне этих тегов рассматривается как код html.

В этом случае интерпретатор сформирует следующий html-код:

```
<!DOCTYPE html>
<html>
    <head>
        <title>METANIT.COM</title>
        <meta charset="utf-8" />
    </head>
    <body>
        <h1>Сайт на PHP</h1>
        Привет мир!
    </body>
</html>
```

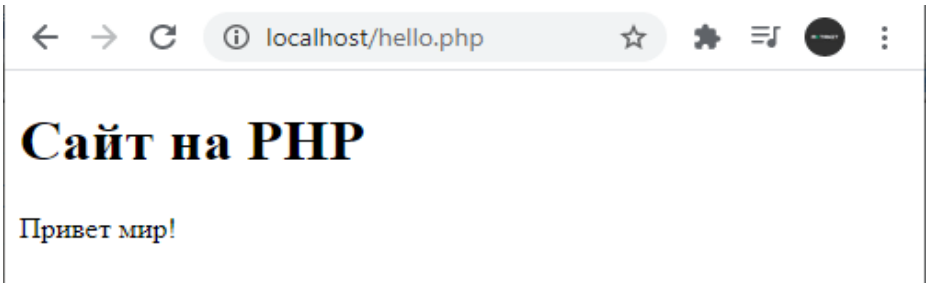

Рис. 14.2. Результат html-код.

При этом естественно мы можем использовать больше инструкций, а также встраивать код php в различных частях веб-страницы. Например:

```
<!DOCTYPE html>
<html>
  <head>
        <title>METANIT.COM</title>
        <meta charset="utf-8" />
  </head>
  <body>
        <h1>
<?php
echo "Первый сайт на PHP";
?>
        </h1>
        <div>
<?php
echo "<h2>Заголовок параграфа</h2>";
echo "Текст параграфа";
?>
        </div>
  </body>
</html>
```

В данном случае код PHP встраивается в двух местах. В первом случае - внутри элемента <h1>. Во втором случае внутри элемента

Это даст нам следующий результат:

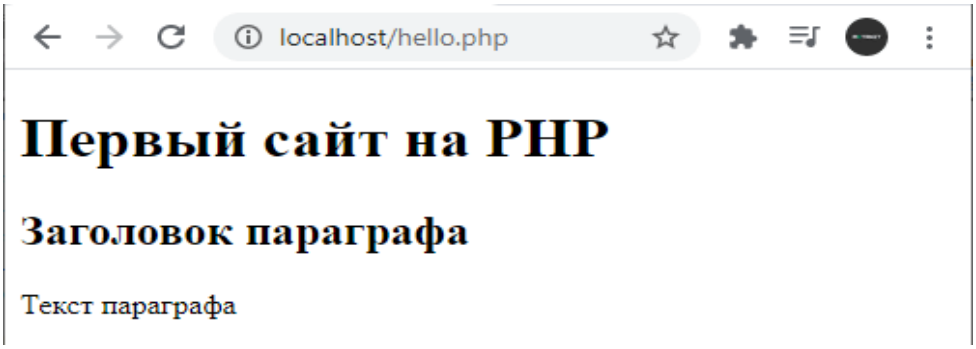

Рис. 14.3. Результат код PHP.

Причем при использовании функции echo мы можем включать в выводимый текст html-код, как в случае с выражением:

```
echo "<h2>Заголовок параграфа</h2>";
```

Хотя выше код php определялся в файле с расширением .php, но равным образом мы также можем определять код в файлах с расширением .html, и они также будут обрабатываться интерпретатором PHP.

Сокращенная версия тегов php

Если нам надо вывести на веб-страницу одно какое-нибудь значение, то мы можем использовать специальную форму тегов php - <?= ... ?> - после знака = ("равно") ставится выводимое выражение. Например:

```
<!DOCTYPE html>
<html>
  <head>
        <title>METANIT.COM</title>
        <meta charset="utf-8" />
  </head>
  <body>
        <h1>
<?= "Первый сайт на PHP"; ?>
        </h1>
        <div>
2 + 2 =  <?= 2+2 ?>
        </div>
  </body>
</html>
```

В первом случае выводится строка <?= "Первый сайт на PHP"; ?>.

Во втором случае выводится результат выражения 2 + 2: <?= 2+2 ?>.

Результат работы скрипта:

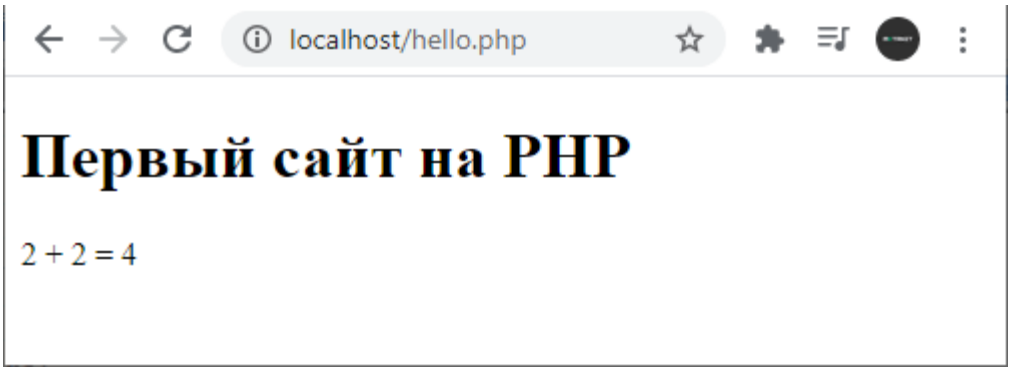

Рис. 14.4. Результат код PHP.

При создании веб-сайта мы можем использовать комментарии. Например, мы можем прокомментировать какое-либо действие, чтобы впоследствии иметь представление, что этот код делает:

```
<?php
echo "<p>Привет мир!</p>"; // вывод сообщения
// echo "Пока мир";
?>
```

Знак // предваряет однострочный комментарий, и все что идет после этого знака на одной строке, будет считаться комментарием и не будет выполняться интерпретатором. При обработке интерпретатор просто будет пропускать комментарии.

Если нам надо закомментировать несколько строк, то мы можем использовать многострочный комментарий /* текст комментария*/:

```
<?php
echo "<p>Привет мир!</p>"; // вывод сообщения
/*
многострочный комментарий
вывод результата арифметического выражения
echo "2 + 2 = " . (2+2);
*/
?>
```

Все строки внутри комментария также не будут обрабатываться интерпретатором.

Контрольные вопросы.

1. Чем отличаются ключевые слова: include и require, mysql_connect и mysql_pconnect?
2. Что такое интерфейсы? Используете ли вы их? Если да — расскажите об этом.
3. Что такое абстрактный класс и чем он отличается от интерфейса?
4. Может ли абстрактный класс содержать частный метод?

ЛАБОРАТОРНАЯ РАБОТА №15

Отправка данных на сервер. Получение данных из строки запроса

Цель работы: Приобрести навыки отправки данных на Сервер и получения данных из строки запроса в PHP.

Задание: Выполнение задач по отправке данных на сервер и получению данных из строки запросов в PHP.

Методические указания:

Отправка данных на сервер. Получение данных из строки запроса.

Самым простым способом передачи данных на сервер приложению PHP извне представляет передача данных через строку запроса.

Строка запроса представляет набор параметров, которые помещаются в адресе после вопросительного знака. При этом каждый параметр определяет название и значение. Например, в адресе:

http://localhost/user.php?name=Tom&age=36

Часть ?name=Tom&age=36 представляет строку запроса, в которой есть два параметра name и age. Для каждого параметра определено имя и значение, которые отделяются знаком равно. Параметр name имеет значение "Tom", а параметр age - значение 36. Друг от друга параметры отделяются знаком амперсанда.

Например, определим следующий скрипт user.php со следующим содержимым:

```
<?php
$name = "не определено";
$age = "не определен";
if(isset($_GET["name"])){

   $name = $_GET["name"];
}
if(isset($_GET["age"])){

   $age = $_GET["age"];
}
echo "Имя: $name <br> Возраст: $age";
?>
```

Когда мы вводим в адресную строку браузера некий адрес и нажимаем на оправку, то серверу отправляется запрос типа GET. В PHP по умолчанию определен глобальный ассоциативный массив $_GET, который хранит все значения, передаваемые в запроса GET. Используя ключи передаваемых данных, мы можем из массива $_GET получить передаваемые значения.

При отправки строки запроса ключами в этом массиве будут названия параметров, а значениями - значения параметров.

Например, в строке запроса передается параметр name=Tom. Соответственно, чтобы получить значение параметра name из запроса, обращаемся по соответствующему ключу:

$name = $_GET["name"]; // Tom

Однако стоит учитывать, что в адресной строке необязательно будет использоваться строка запроса или конкретно данный параметр. Поэтому перед получением значения параметра сначала смотрим, а передан ли вообще такой параметр:

if(isset($_GET["name"])){

Теперь обратимся к этому скрипту, например, так http://localhost/user.php?name=Tom&age=36:

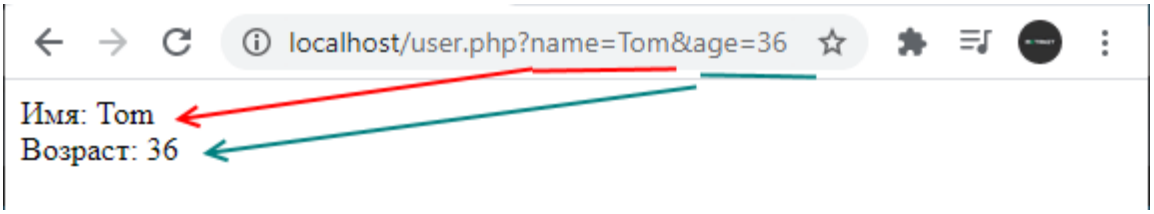

Рис.15.1.Скрипт.

Если мы не передадим значения какого-либо параметра, то соответствующая переменная будет использовать значение по умолчанию:

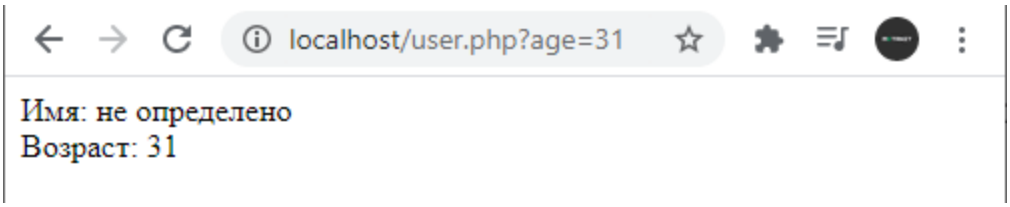

Рис.15.2.Значение по умолчанию.

Одним из основных способов передачи данных веб-сайту является обработка форм. Формы представляют специальные элементы разметки HTML, которые содержат в себе различные элементы ввода - текстовые поля, кнопки и т.д. И с помощью данных форм мы можем ввести некоторые данные и отправить их на сервер. А сервер уже обрабатывает эти данные.

Создание форм состоит из следующих аспектов:

Создание элемента <form><form> в разметке HTML

Добавление в этот элемент одно или несколько поле ввода

Установка метода передачи данных. Чаще всего используются методы GET или POST

Установка адреса, на который будут отправляться введенные данные

POST-запросы

Итак, создадим новую форму. Для этого определим новый файл form.php, в которое поместим следующее содержимое:

```
<!DOCTYPE html>
<html>
<head>
<title>METANIT.COM</title>
<meta charset="utf-8" />
</head>
<body>
<h3>Форма ввода данных</h3>
<form action="user.php" method="POST">
    <p>Имя: <input type="text" name="name" /></p>
    <p>Возраст: <input type="number" name="age" /></p>
    <input type="submit" value="Отправить">
</form>
</body>
</html>
```

Атрибут action="user.php" элемента form указывает, что данные формы будет обрабатывать скрипт user.php, который будет находиться с файлом form.php в одной папке. А атрибут method="POST" указывает, что в качестве метода передачи данных будет применяться метод POST.

Теперь определим файл user.php, который будет иметь следующее содержание:

```
<?php
$name = "не определено";
$age = "не определен";
if(isset($_POST["name"])){

    $name = $_POST["name"];
}
if(isset($_POST["age"])){

    $age = $_POST["age"];
```

```
}
echo "Имя: $name <br> Возраст: $age";
?>
```

Для обработки запросов типа POST в PHP используется встроенная глобальная переменная $_POST. Она представляет ассоциативный массив данных, переданных с помощью метода POST. Используя ключи, мы можем получить отправленные значения. Ключами в этом массиве являются значения атрибутов name у полей ввода формы.

Например, так как атрибут name поля ввода возраста имеет значение age (<input type="number" name="age" />), то в массиве $_POST значение этого поля будет представлять ключ "age": $_POST["age"]

И поскольку возможны ситуации, когда поле ввода будет не установлено, то в этом случае желательно перед обработкой данных проверять их наличие с помощью функции isset(). И если переменная установлена, то функция isset() возвратит значение true.

Теперь мы можем обратиться к скрипту form.php и ввести в форму какие-нибудь данные:

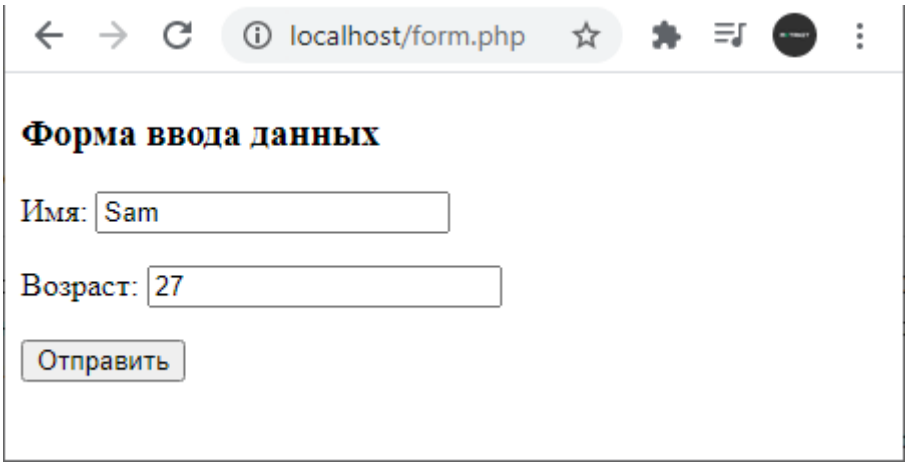

Рис.15.3. Результат form.php.

И по нажатию кнопки введенные данные методом POST будут отправлены скрипту user.php:

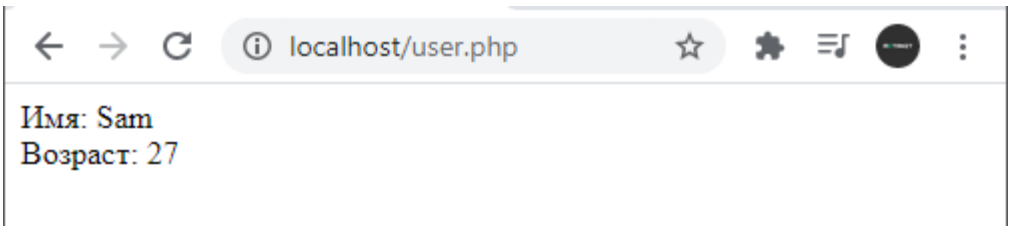

Рис.15.4. Скрипт user.php.

Необязательно отправлять данные формы другому скрипту, можно данные формы обработать в том же файле формы. Для этого изменим файл form.php следующим образом:

```
<!DOCTYPE html>
<html>
<head>
<title>METANIT.COM</title>
<meta charset="utf-8" />
</head>
<body>
<?php
$name = "не определено";
$age = "не определен";
if(isset($_POST["name"])){
    $name = $_POST["name"];
}
if(isset($_POST["age"])){

    $age = $_POST["age"];
}
echo "Имя: $name <br> Возраст: $age";
?>
<h3>Форма ввода данных</h3>
<form method="POST">
    <p>Имя: <input type="text" name="name" /></p>
    <p>Возраст: <input type="number" name="age" /></p>
    <input type="submit" value="Отправить">
</form>
</body>
</html>
```

Поскольку в данном случае мы отправляем данные этому же скрипту - то есть по тому же адресу, то у элемента форма можно не устанавливать атрибут action.

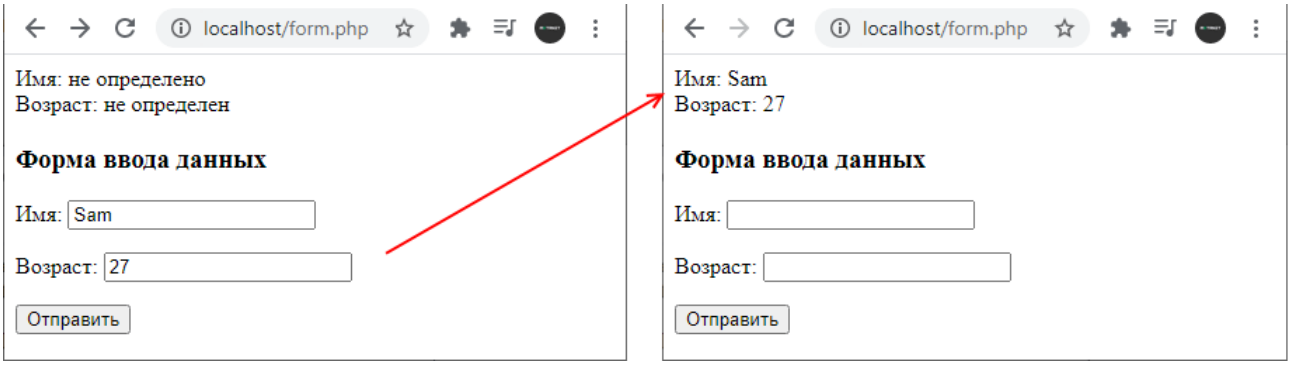

Рис.15.5. Данные отправляются в тот же скрипт.

Стоит отметить, что в принципе мы можем отправлять формы и запросом GET, в этом случае для получения тех же значений формы применяется массив $_GET, который был рассмотрен в прошлой теме:

```
<!DOCTYPE html>
<html>
  <head>
        <title>METANIT.COM</title>
        <meta charset="utf-8" />
  </head>
  <body>
<?php
$name = "не определено";
$age = "не определен";
if(isset($_GET["name"])){

   $name = $_GET["name"];
}
if(isset($_GET["age"])){

   $age = $_GET["age"];
}
echo "Имя: $name <br> Возраст: $age";
?>
        <h3>Форма ввода данных</h3>
```

```
        <form method="GET">
        <p>Имя: <input type="text" name="name" /></p>
        <p>Возраст: <input type="number" name="age" /></p>
        <input type="submit" value="Отправить">
        </form>
    </body>
</html>
```

Контрольные вопросы.

1. В чем разница между замыканием в PHP и JavaScript?
2. Какие версии PHP до сих пор поддерживаются?
3. В чем разница между GET и POST?
4. Чем отличаются операторы BREAK и CONTINUE?
5. Есть ли разница между одинарными и двойными кавычками?

ЛИТЕРАТУРА

1. V.G. Olifer, N. A.Olifer. Computer networks. Principles, technologies,protocols. St. Petersburg: Publishing House. "Peter," in 2000.
2. R. Darnell, "JavaScript: Handbook." - St. Petersburg: Publishing House."Peter," in 2008.
3. Duvanov A.A. WEB. Construction of HTML. - St. Petersburg: BHV, 2003.
4. Stephen Speynaur, Valerie Kuersia. Directory WEB-master. - Ed. BHV Barrett D. JavaScript. Web-professionals. - Kiev: BHV - Kyiv, 2001.
5. Brandenbau D. JavaScript: a collection of recipes. - St. Petersburg.: Peter,2000.
6. Budilov B. JavaScript, XML and the Document Object Model. - St.Petersburg.: S & T, 2001.
7. R. Wagner JavaScript. Encyclopedia member (+ CD-ROM). - Kiev:DiaSoft, 2001.
8. Vayk A. JavaScript in examples. - Kiev: DiaSoft, 2000.
9. Rev. E. Vander JavaScript for "Dummies." - Dialectic, 2001.
10. P. Weiner JAVA programming languages, and JavaScript. - M: LORI,2000.
11. Garnaev A Web-programming in Java and JavaScript. - St. Petersburg.:BHV St. Pereburg, 2002.
12. Darnell R. JavaScript. Handbook. - St. Petersburg.: Peter, 2000.
13. Dmitriev M. Tutorial JavaScript. - St. Petersburg.: BHV St. Pereburg, 2001.
14. Dmitriev M. JavaScript. Quick start. - St. Petersburg.: BHV St. Pereburg,2002.
15. Rob Larsen. Beginning HTML & CSS. Wiley Publishing, Inc. Indianapolis, Indiana, 2013.
16. Zokirova T.A., Sharipov B.A., Rasulova N.A. "Web - dasturlash" fanidan o‘quv qo‘llanma – T.: TDIU, 2009. – 216б.
17. Зокирова Т.А., Ходиева Р.М., Мусаева М.А. Web дастурлаш. Ўқув қўлланма. ТДИУ. Тошкент – 2006 й. – 224б.
18. Робин Никсон., Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5. Учебное пособие. Издательство Питер. 2017.-768с.

19. Левин М.П. Самоучитель разработки web-сайтов: HTML, CSS, графика, анимация, раскрутка, видеокурс DVD. М.:Изд.:Триумф., 2007., - 400с.
20. Дротов В.А. JAVA SCRIPT в Web – дизайне. − СПб.: «БХВ-Петер-бург», 2002.- 880 с.
21. www.search.re.uz - Информационно-поисковая система Узбекистана.
22. www.linux.org.ru
23. www.freebsd.com
24. www.intuit.ru
25. www.ziyonet.uz

СОДЕРЖАНИЕ

Редактор: Ахметжанова Г.М.