ЗАДАНИЯ И МЕТОДИЧЕСКИЕ УКАЗАНИЯ
по практическим работам
по дисциплине бакалавриата
«Разработка клиент-серверных приложений»
(направление подготовки:
09.03.04 «Программная инженерия»)

Содержание

Содержание отчетов по практическим работам

Отчеты по практическим работам оформляются в электронном виде с именами **Петров (Пр1).doc** (или *.docx, *.rtf, *.odt, *.pdf).

Отчеты оформляются по практическим работам: №1, №2, №3, №4, №5.

Образец титульного листа к отчетам прилагается в конце документа.

Практическая работа №1. Статический html-документ

(4 часа)

Цель работы

Изучить основы языка разметки гипертекста HTML 5.

Задание

Создать html-документ, содержащий заголовок, текстовую часть, оформленную разными шрифтами, список, содержащий перечень предметов выбранной предметной области (животные, еда, транспорт, фильмы и т.д.) в виде ссылок на картинки, соответствующие элементам списка. При нажатии на ссылку должен открываться еще один html-документ с соответствующей картинкой.

В разметке html-документа **ВО ВСЕХ ВАРИАНТАХ** использовать:

– тег <DOCTYPE>;

– тег <meta> для определения кодировки текста;

– тег <!-- --> комментария;

– теги: <p>,
, , , <hr>, один из <h1> ÷ <h6>, <b>, <i>, <u>, <sub>, <sup>, <pre>, <mark>, <details>, <summary>;

– продемонстрировать отличие в тегах и ;

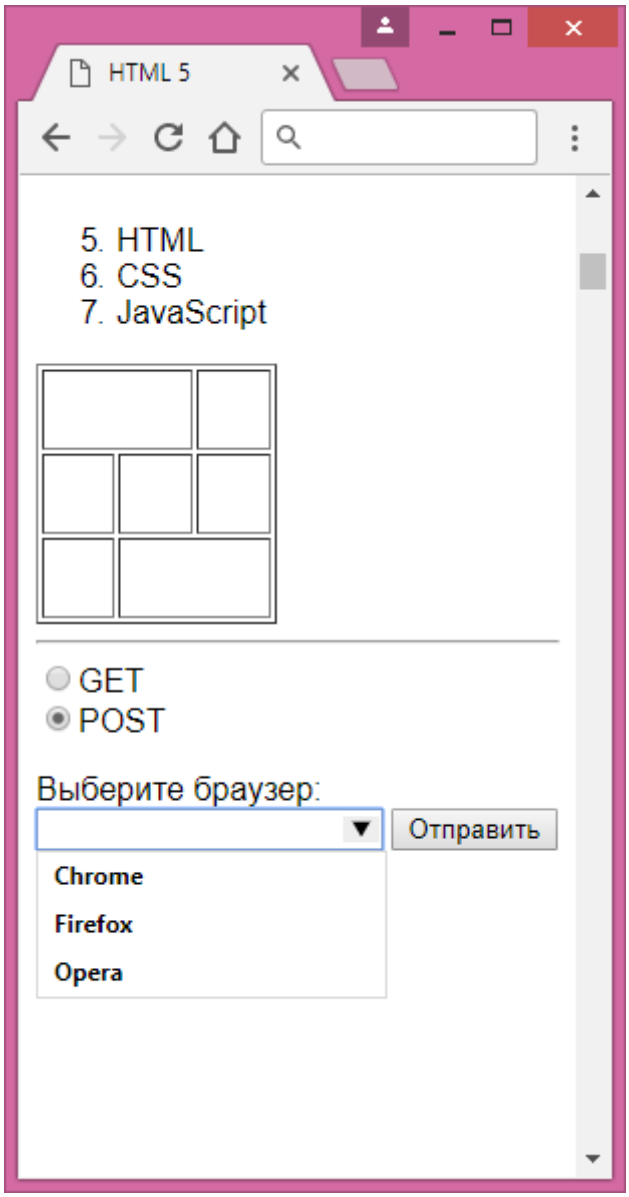
– теги <figure>, <img>, <figcaption> для вставки изображения;

– тег <a> для разметки гиперссылок, разметить ссылки: **на другой документ**, **в пределах** размечаемого **документа**, **на email**;

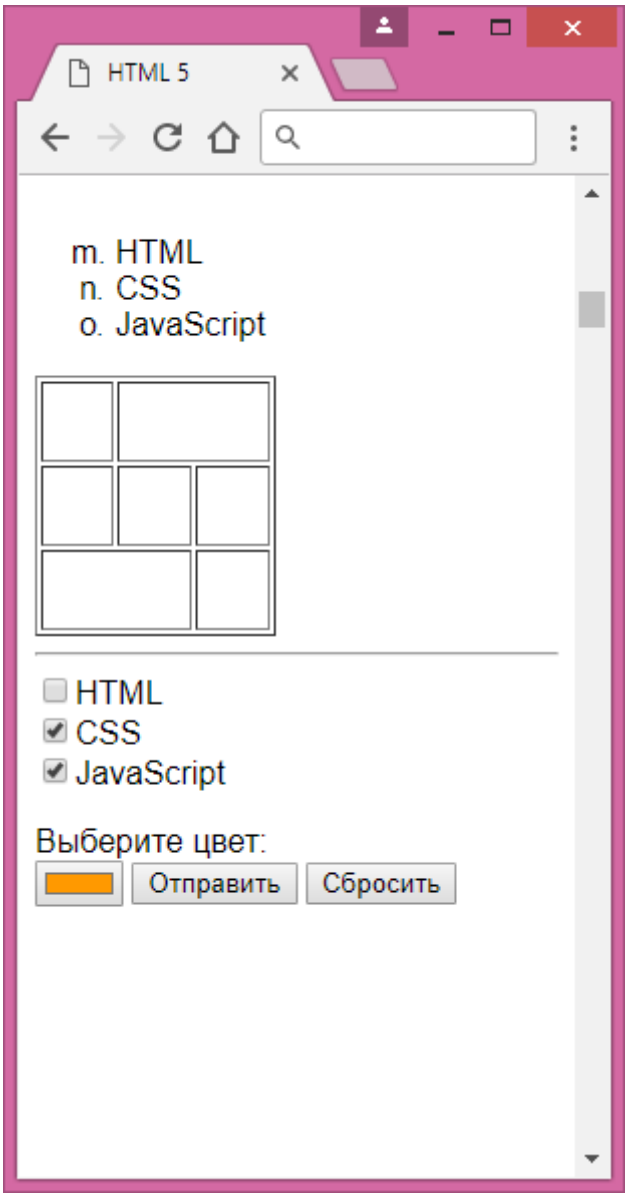
– тег <li> для разметки списка.

Задания по вариантам:

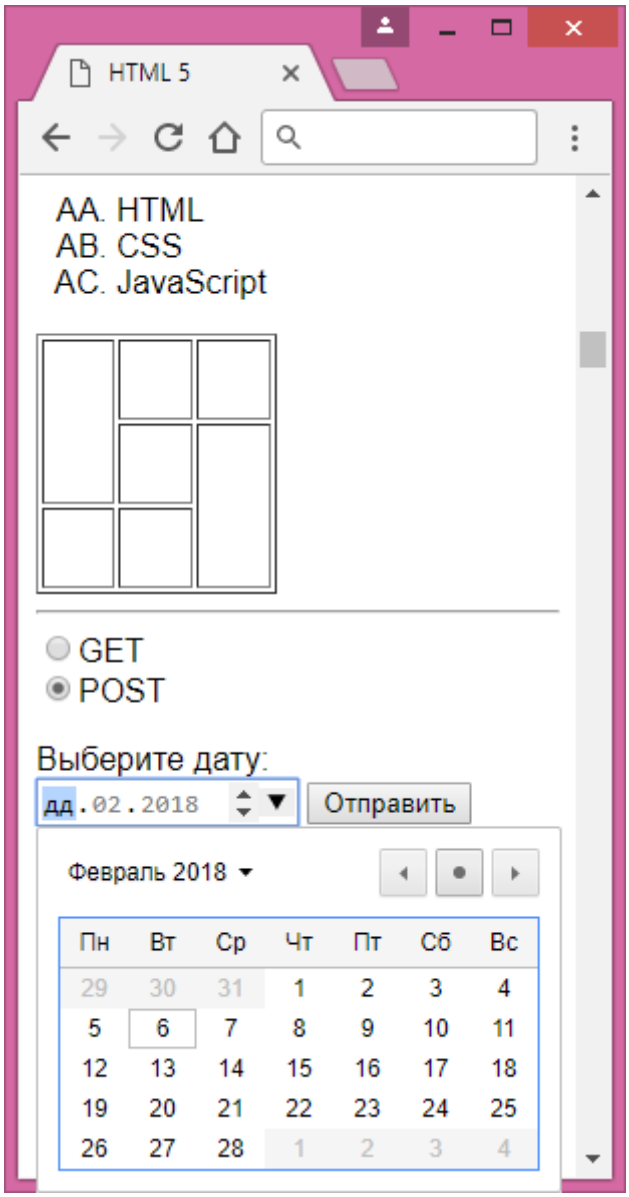
Вариант 1:



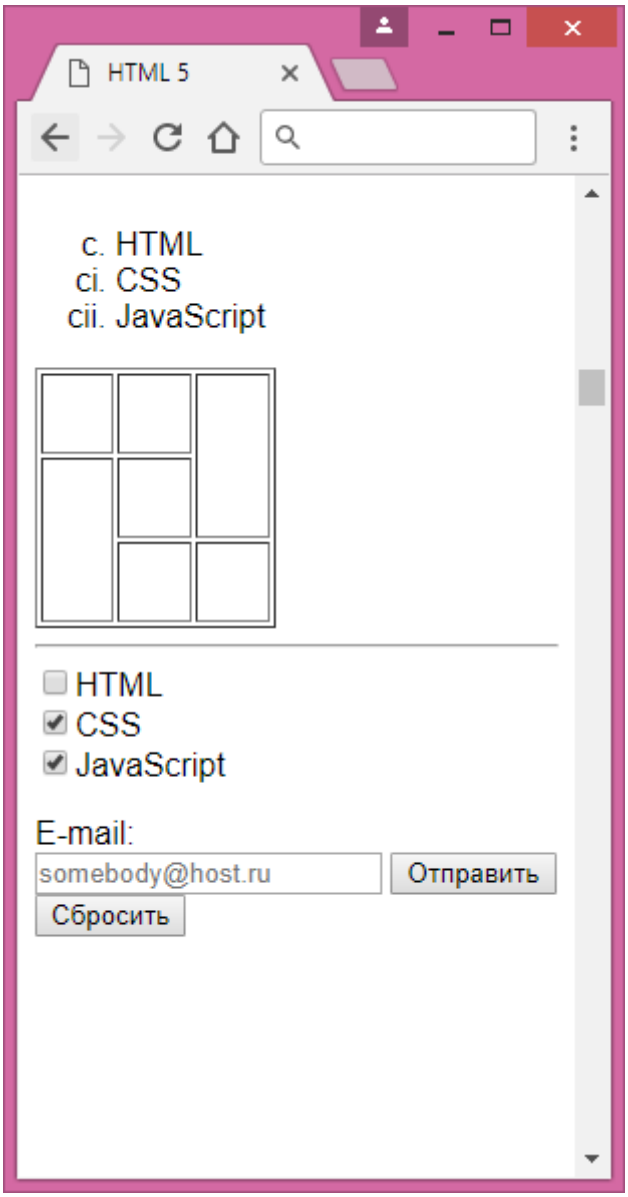
Вариант 2:



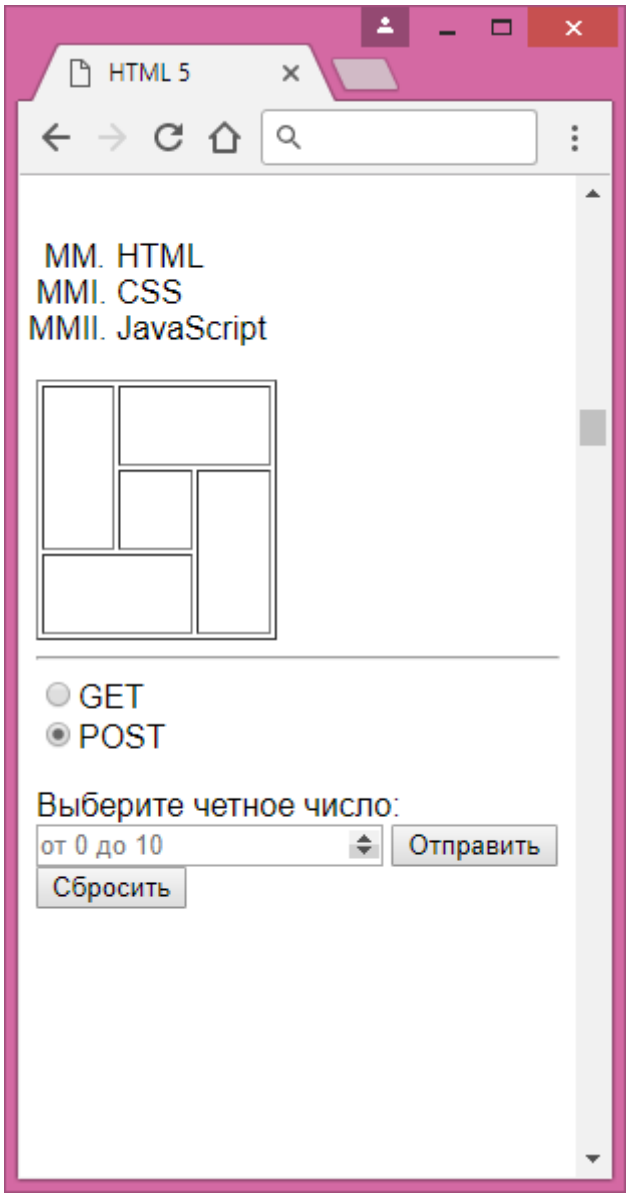
Вариант 3:



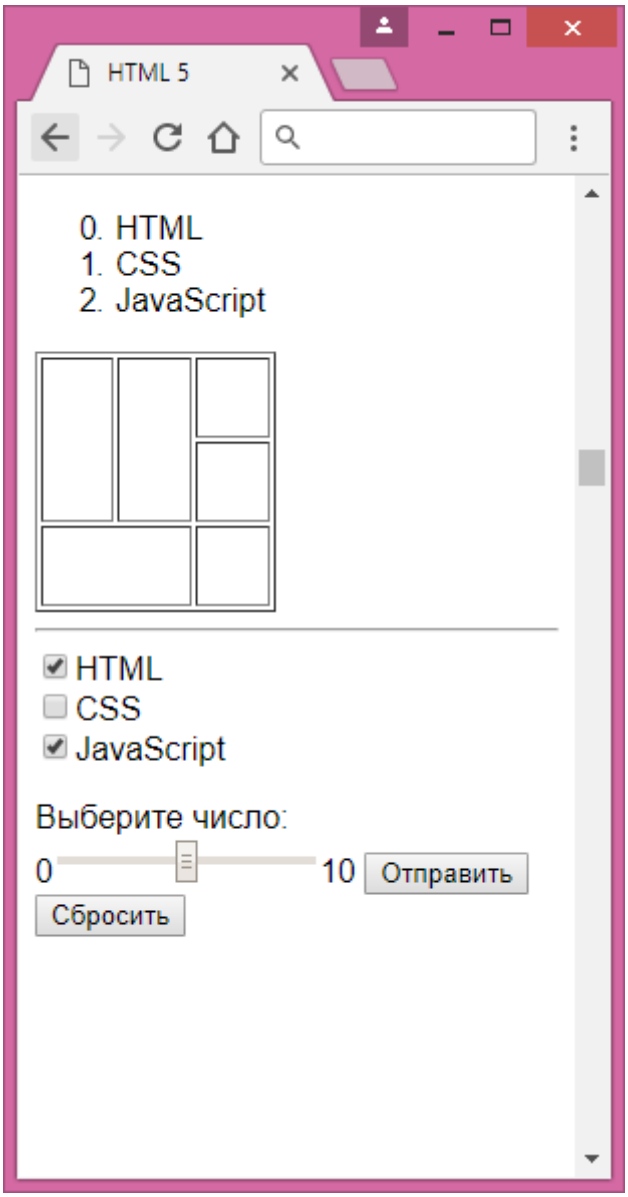
Вариант 4:



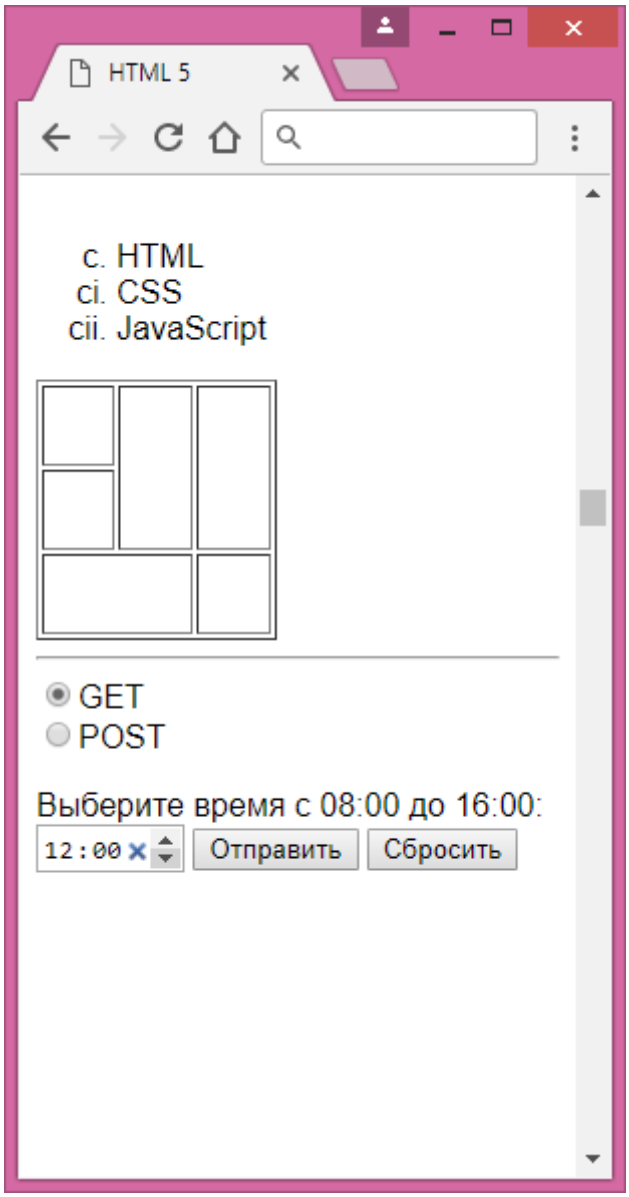
Вариант 5:



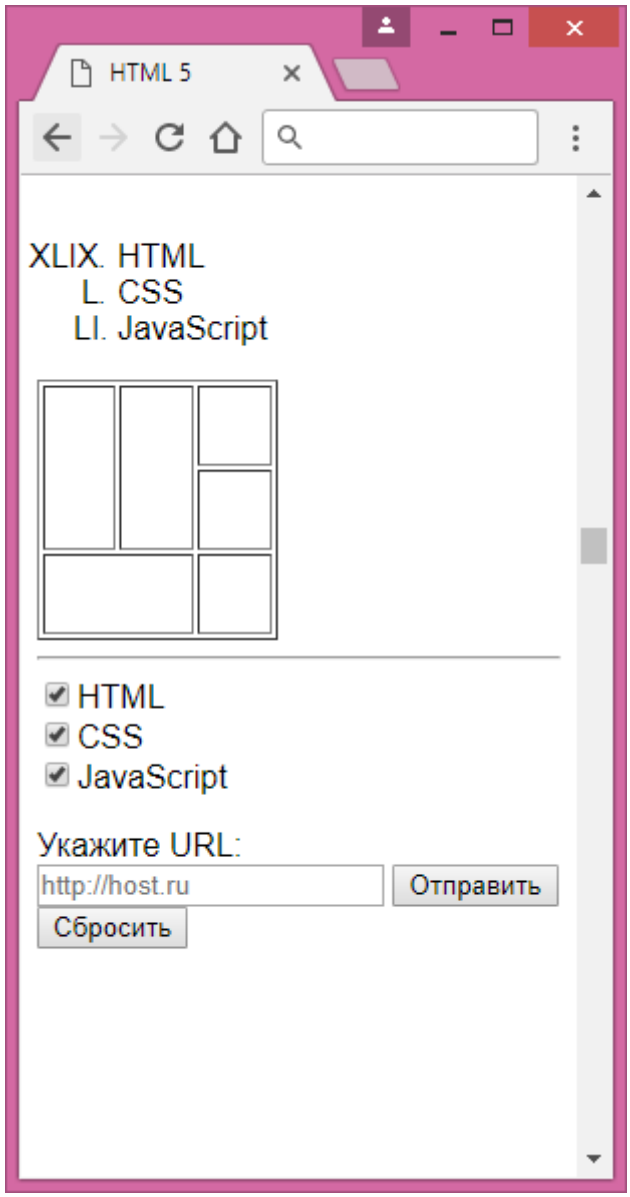
Вариант 6:



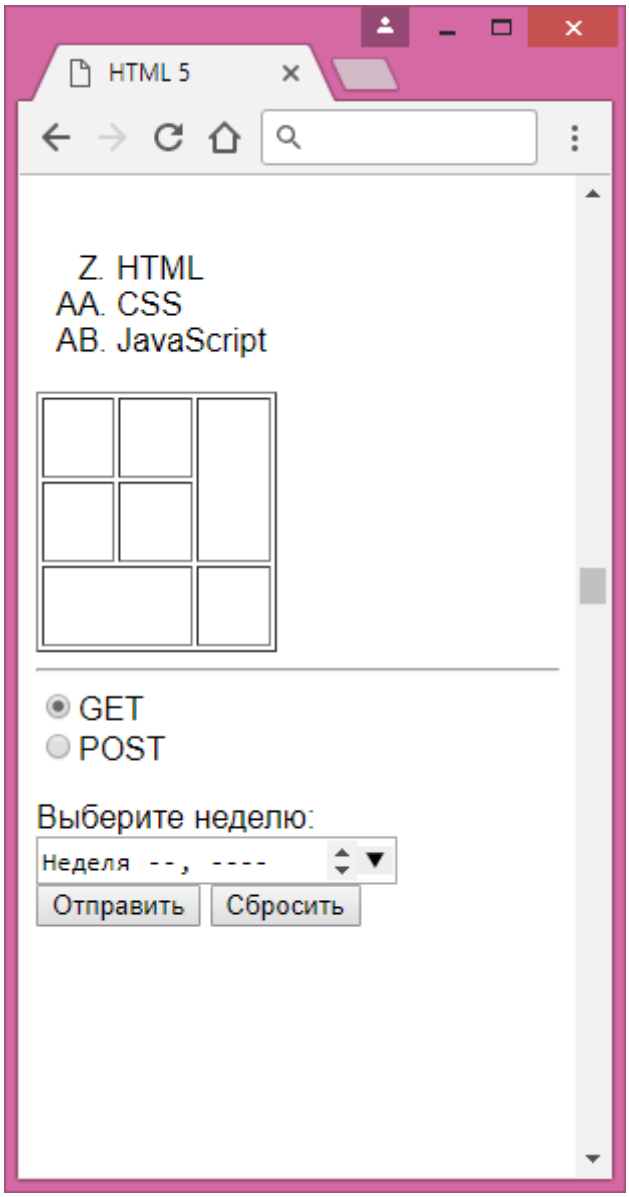
Вариант 7:



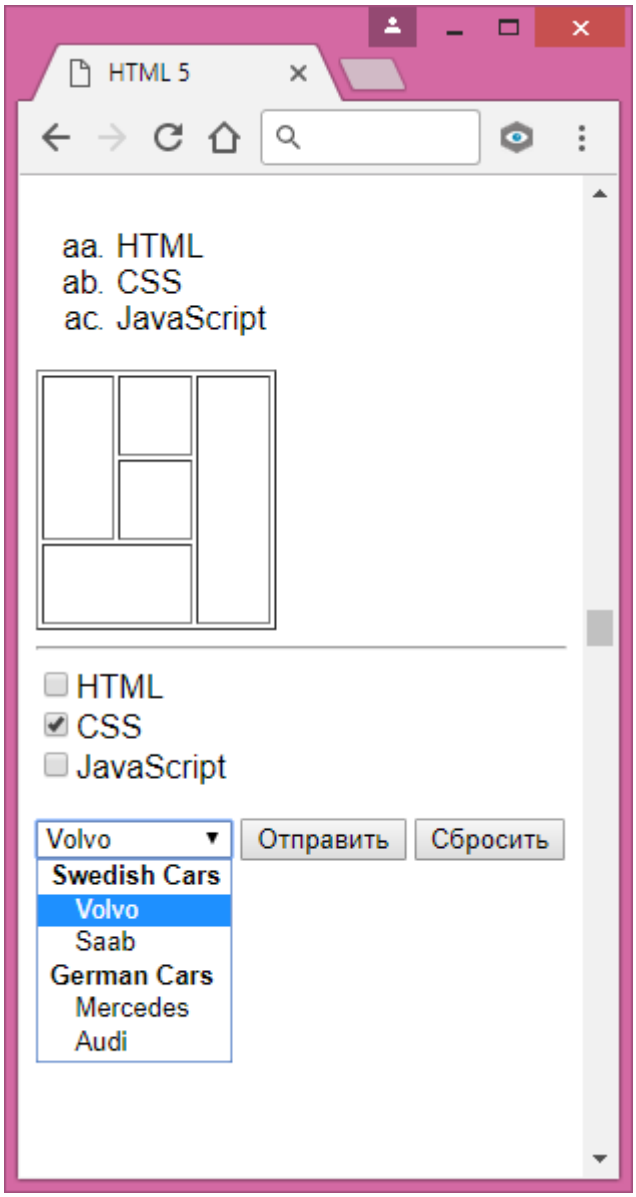
Вариант 8:



Вариант 9:



Вариант 10:



Порядок выполнения лабораторной работы

1) Запустить текстовый редактор.

2) Разметить html-документ в соответствии с выбранной темой, пошагово выполняя пункты раздела «Простейшие» учебника http://umnik.rikt.ru/informat/Library/html2/. Для справки о тегах и их атрибутах использовать справочники, расположенные по адресам: https://www.w3schools.com/html/default.asp, https://html5book.ru/html-html5.

Стандарт HTML 5 расположен по адресу https://www.w3.org/standards/techs/html.

3) Проверить соответствие выполненной разметки стандарту HTML 5, использовав валидатор WWW Консорциума, расположенный по адресу http://validator.w3.org.

Содержание отчета

В отчет по лабораторной работе входит:

- цель работы;
- задание;
- порядок выполнения лабораторной работы;
- html-разметка созданного документа, включая созданные стили;
- скриншот с результатами валидации html-документа;
- скриншот оформленного html-документа;
- выводы.

Теоретические сведения

Язык разметки гипертекста HTML (Hypertext markup language) — язык разметки, используемый для создания гипертекстовых html-документов, отображаемых браузером.

Гипертекст — форматированный текст, содержащий ссылки на другие документы (гиперссылки).

Разметка — вставка в текст дополнительных служебных символов, каждый из которых является командой, указывающей браузеру, как следует отображать документ.

Язык разметки гипертекста HTML не является языком программирования.

Основным элементом языка разметки гипертекста HTML является тег (tag).

Теги содержат указания браузеру о способах отображения документа.

С помощью тегов в html-документ вставляются файлы, содержащие дополнительные данные (например, графику) и размечаются гиперссылки, посредством которых данный html-документ связывается с другими html-документами.

Как правило, теги состоят из начального и конечного элементов, между которыми размещаются текст и другие элементы html-документа. Имя конечного тега совпадает с именем начального, но перед именем конечного тега ставится косая черта.

Базовый синтаксис тега:

```
<name>
Содержимое тега
</name>
```

где <name> — это начальный элемент тега, содержащий имя тега, а </name> — конечный элемент тега.

В начальном элементе тега может располагаться перечень атрибутов тега. Атрибуты тега следуют за именем и отделяются друг от друга одним или несколькими пробелами. Порядок записи атрибутов в начальном элементе тега значения не имеет. Значение атрибута, если имеется, следует за знаком равенства, стоящим после имени атрибута. Если значение атрибута — одно слово или число, то его можно указать после знака равенства, не заключая в кавычки. Все остальные значения необходимо заключать в кавычки, особенно если они содержат пробелы. Если атрибут не указан, браузером используется его значение по умолчанию.

Регистр символов в именах тегов и атрибутов не учитывается.

<name attribute_1=”value_1” attribute_2 … attribute_n=”value_n”>
Содержимое тега
</name>

Конечные теги никогда не содержат атрибутов.

При использовании вложенных тегов их нужно закрывать, соблюдая правильную вложенность.

В некоторых случаях конечные теги можно опускать. Тем не менее, рекомендуется использовать конечные элементы тегов, чтобы избежать ошибок в отображении html-документа браузером.

Некоторые теги, не имеющие конечного элемента, называются автономными тегами.

<name>

<name attribute_1=”value_1” attribute_2 … attribute_n=”value_n”>

Html-документ состоит из заголовка документа и тела документа:

<html>
<head>
<title>Название html-документа</title>
Заголовок html-документа
</head>

```
<body>
Тело html-документа
</body>
</html>
```

Весь html-документ заключается в тег <html>. Html-документ состоит из заголовка и тела, которые выделяются, соответственно, тегами <head> и <body>. В заголовке, с помощью тега <title>, указывается название html-документа, а также другие данные, которые браузер будет использовать при отображении документа.

Тело html-документа — та его часть, в которую помещается собственно содержимое html-документа. Тело включает предназначенный для отображения текст и управляющую разметку документа (теги), которые используются браузером.

Перечень тегов языка HTML и их атрибутов можно посмотреть в справочнике http://htmlbook.ru.

Практическая работа №2. Статический html-документ. Таблицы и фреймы

(4 часа)

Цель работы

Продолжение изучения основ языка разметки гипертекста HTML 5.

Задание

Оформить html-документ, полученный в результате выполнения лабораторной работы №1 «Статический html-документ», добавив таблицы и фреймы.

Использовать тэги:

– <table>, <tr>, <td>;

– <frame>, <frameset>.

В первом html-документе, созданном в лабораторной работе №1, добавить таблицу с разными размерами, цветами и текстовой наполненностью ячеек. Второй html-документ сделать в виде фреймов, в которых располагаются картинки, на которые были созданы ссылки в первой работе.

Порядок выполнения лабораторной работы

1) Запустить текстовый редактор.

2) Разметить html-документ в соответствии с выбранной темой, пошагово выполняя пункты раздела «Таблицы» и «Фреймы» учебника http://umnik.rikt.ru/informat/Library/html2/. Для справки о тегах и их атрибутах использовать справочники, расположенные по адресам: https://www.w3schools.com/html/default.asp, https://html5book.ru/html-html5. Стандарт HTML 5 расположен по адресу https://www.w3.org/standards/techs/html.

3) Проверить соответствие выполненной разметки стандарту HTML 5, использовав валидатор WWW Консорциума, расположенный по адресу http://validator.w3.org.

Содержание отчета

В отчет по лабораторной работе входит:

- цель работы;
- задание;
- порядок выполнения лабораторной работы;
- html-разметка созданного документа, включая созданные стили;
- скриншот с результатами валидации html-документа;
- скриншот оформленного html-документа;
- выводы.

Теоретические сведения

См. теоретические сведения лабораторной работы №1.

Практическая работа №3. Каскадные таблицы стилей CSS
(4 часа)

Цель работы

Изучить основы каскадных таблицы стилей CSS 3.

Задание

Оформить html-документ, полученный в результате выполнения лабораторной работы №1 и №2 «Статический html-документ», добавив каскадные таблицы стилей. Изменить оформление:

- документа в целом (фон и т.п.);
- текста;
- гиперссылок;
- списка;
- таблицы.

Использовать определение стилей для тегов и классы стилей, псевдоклассы.

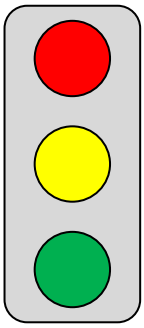
Использовать тэги:

- с помощью тега <link>;
- с помощью тега <style>;
- с помощью параметра style тега.

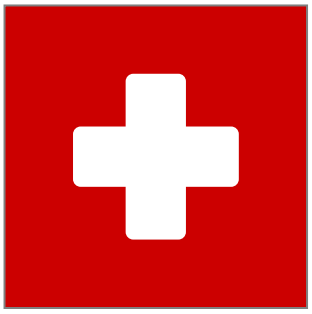
Продемонстрировать действие приоритетов при применении различных способов определения CSS.

Создать изображение в соответствии с вариантом, используя только свойства CSS.

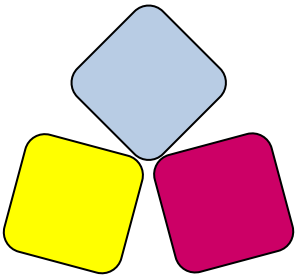
Вариант 1:



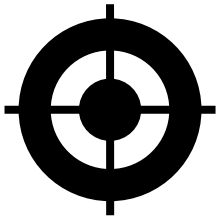
Вариант 2:



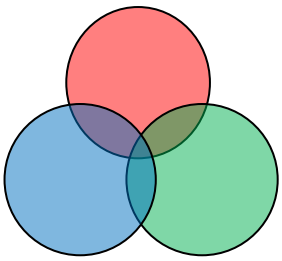
Вариант 3:



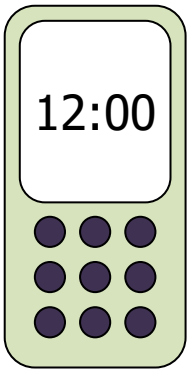
Вариант 4:



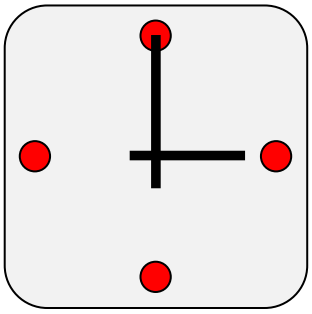
Вариант 5:



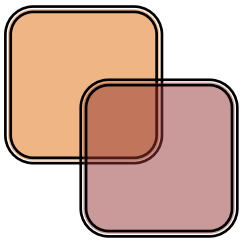
Вариант 6:



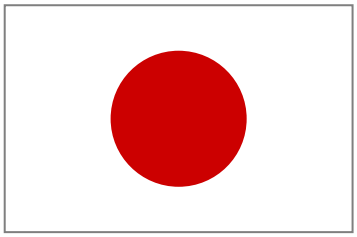
Вариант 7:



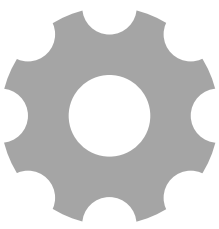
Вариант 8:



Вариант 9:



Вариант 10:



Порядок выполнения лабораторной работы:

1) запустить текстовый редактор;

2) оформить с помощью каскадных таблиц стилей html-документ, полученный в результате выполнения лабораторной работы №1, в соответствии с заданным вариантом. Для справки о свойствах CSS и их значениях использовать справочники, расположенные по адресам: https://www.w3schools.com/css/default.asp и http://htmlbook.ru/css. Стандарты CSS 3 расположены по адресу https://www.w3.org/Style/CSS;

3) протестировать оформленный документ в браузере.

Содержание отчета

В отчет по лабораторной работе входит:

- цель работы;
- задание;
- порядок выполнения лабораторной работы;
- html-разметка созданного документа, включая созданные стили;
- скриншот с результатами валидации html-документа;
- скриншот оформленного html-документа;
- выводы.

Теоретические сведения

Каскадные таблицы стилей CSS (Cascading style sheets) — формальный язык описания внешнего вида документа, созданного с использованием языка разметки гипертекста.

Каскадные таблицы стилей позволяют разделить описание логической структуры html-документа (выполненное с помощью языка разметки) и описание внешнего вида html-документа (выполненное с помощью CSS).

Существует три способа определения стилей: 1) в отдельном файле, подключаемом к html-документам, 2) с помощью тега <style> непосредственно в некотором html-документе и 3) с помощью атрибута style непосредственно в некотором теге.

Наиболее высокий приоритет имеет стиль, определенный в теге, затем следует определение стиля с помощью тега style и самым низким приоритетом обладают свойства, определенные в отдельном файле.

Каскад приоритетов особенно удобен при разработке больших проектов, например, сайтов, состоящих из большого числа html-документов. В этом случае общее оформление может быть вынесено в отдельный файл, в html-документе могут быть внесены изменения в стиль документа с помощью тега <style>, атрибут тега style позволяет изменить оформление одного тега.

Стили определяются парами свойств и значений, перечень пар заключается в фигурные скобки и пары разделяются точкой с запятой:

```
{property_1:value_1; property_2:value_2; … ; property_n:value_n}
```

где property — это свойство, а value — значение свойства.

Стиль можно определить для конкретного тега, например, задать для тега <body> отображение белого текста на черном фоне:

```
body
{background-color:black; color:white}
```

Можно определить «чистый» стиль, не привязанный заранее к конкретному тегу, в этом случае речь идет об определении класса стиля:

```
.small_silver
{font-size:10px; color:silver}
```

или

```
#big_gold
{font-size:150px; color:#D7B56D}
```

Применение класса стиля:

```
<p class=small_silver>Текст светло-серого цвета размером 10 пиксел</p>
```

или

```
<p id=big_gold>Текст светло-желтого цвета размером 150 пиксел</p>
```

Описание стилей для тегов или классов стилей выполняется одинаково как в отдельном файле, так и в теге <style>.

Файл со стилями должен иметь расширение *.css и быть подключен к html-документу с помощью тега <link>, расположенного в теге <head>.

```
<link href=”style.css” rel=”stylesheet” type=”text/css”>
```

Тег <style> также должен быть расположен в теге <head>, после тега <link>.

Стили, определяемые непосредственно в теге с помощью атрибута style:

```
<p style=”text-decoration-line:underline; color:rgb(255,0,0)”>Подчеркнутый текст красного цвета</p>
```

Возможно задание различных стилей отображения одного и того же html-документа в различных средах представления, например, на экране или печати с помощью атрибута media тега <link>.

Файл screen.css

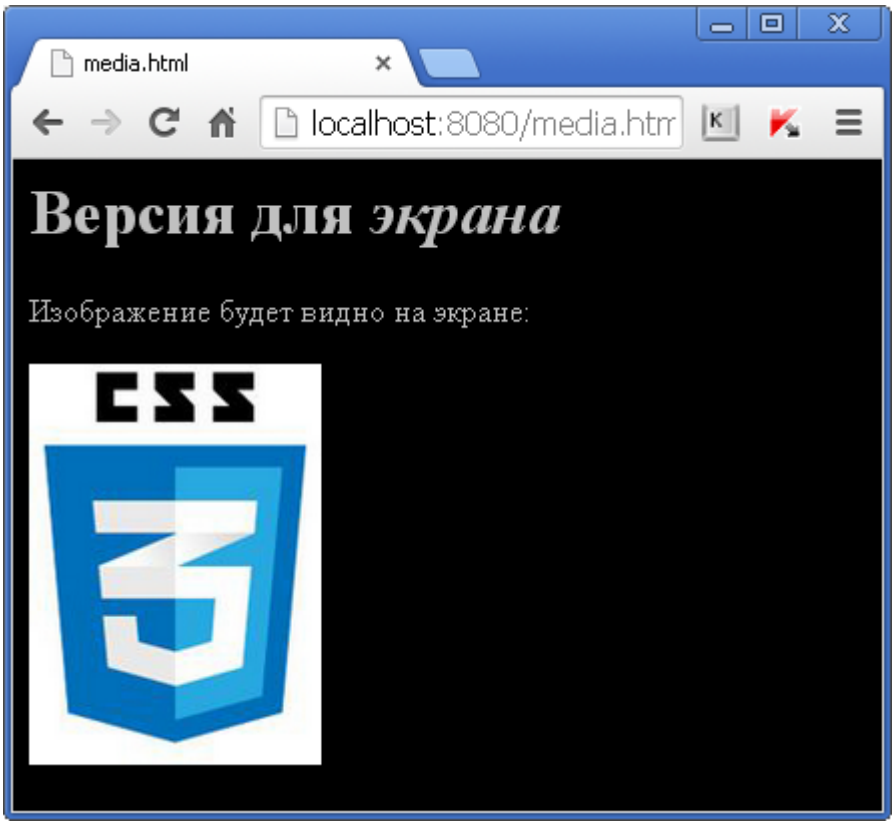
```
body
{color:silver; background:black}
.forprint
{display:none}
```

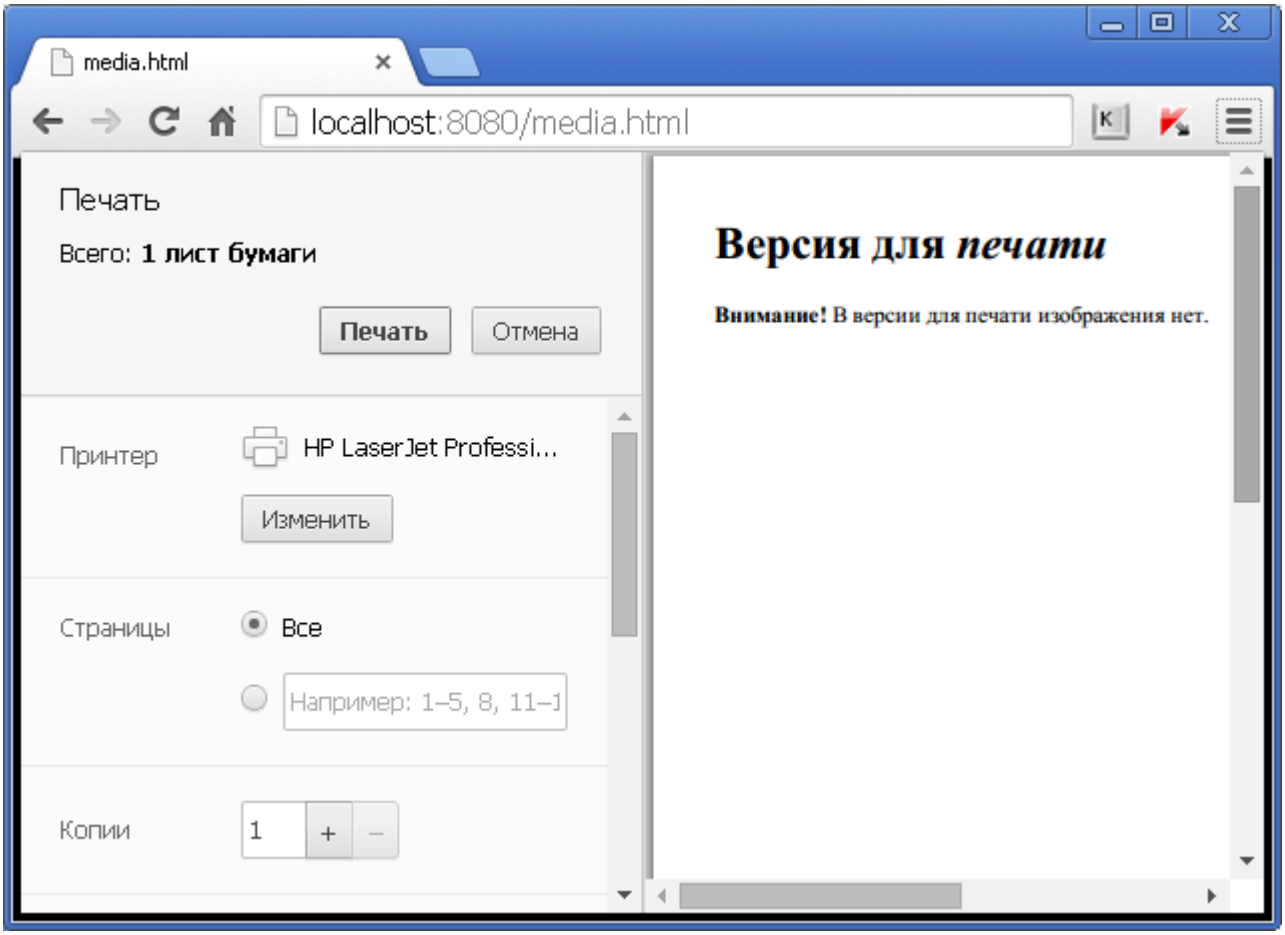
Файл print.css

```
body
{color:black; background:white}
.forscreen
{display:none}
```

Файл media.html

```
<html>
<head>
<link href=”screen.css” rel=”stylesheet” type=”text/css” media=”screen”>
<link href=”print.css” rel=”stylesheet” type=”text/css” media=”print”>
</head>
<body>
<h1>Версия для <i class=forscreen>экрана</i><i class=forprint>печати</i></h1>
<div class=forscreen>Изображение будет видно на экране:
<p><img src=css3.jpg height=200px>
</div>
<div class=forprint><b>Внимание!</b> В версии для печати изображения нет.</div>
</body>
</html>
```


media.html
localhost:8080/media.htm
Версия для экрана
Изображение будет видно на экране:


media.html
localhost:8080/media.html
Печать
Всего: 1 лист бумаги
Печать
Отмена
Принтер
HP LaserJet Professi...
Изменить
Страницы
Все
Например: 1–5, 8, 11–1
Копии
1
Версия для печати
Внимание! В версии для печати изображения нет.

Практическая работа №4. Web-сервер Apache

(4 часа)

Цель работы

Получить практические навыки в установке и выполнении базовой настройки web-сервера Apache.

Задание (типовое)

Установить web-сервер Apache, проверить правильность установки, выполнить настройку web-сервера, протестировать работу web-сервера, удалить web-сервер Apache.

Порядок выполнения лабораторной работы

1) Создать каталог disc:\webprog.
2) Установить web-сервер Apache в каталог disc:\webprog\Apache2.2 как консольное приложение.
3) Запустить web-сервер (Пуск/Все программы/Apache HTTP Server 2.2/Control Apache Server/Start Apache in Console).
4) Проверить правильность установки web-сервера, набрав в строке адреса браузера адрес http://127.0.0.1:8080.
5) Если web-сервер не запускается, посмотреть причину незапуска в файле disc:\webprog\Apache2.2\logs\error.log
6) Для остановки web-сервера использовать комбинацию клавиш Ctrl+C.
7) Ознакомиться с документацией по web-серверу Apache. Для этого в файле httpd.conf убрать комментарий в строке с директивой #Include conf/extra/httpd-manual.conf. Документация будет доступна по адресу http://127.0.0.1:8080/manual
8) Создать два виртуальных хоста на одном IP-адресе 127.0.0.1, настроив их на разные порты, например, 8081 и 8082. Расположить корневые каталоги

документов хостов соответственно в каталогах disc:\webprog\vh1 и disc:\webprog\vh2.

9) Файлы для регистрации доступа access.log и ошибок error.log расположить в каталоге disc:\webprog\vhlogs.

10) Создать файлы с описанием групп пользователей и отдельных пользователей, и расположить их в каталоге disc:\webprog\vhsecurity.

11) При настройке виртуальных хостов изменить, при необходимости, настройки для корневого каталога web-сервера.

12) В корневом каталоге для документов виртуального хоста vh1 создать несколько каталогов и файлов. Определить различные права доступа к различным каталогам и файлам:

- доступ разрешен всем;
- доступ разрешен отдельным пользователям;
- доступ разрешен группе пользователей;
- доступ разрешен всем зарегистрированным пользователям;
- доступ запрещен всем.

13) Протестировать работу SSI (Server Side Includes) — директив включения на стороне сервера.

14) В корневом каталоге для документов виртуального хоста vh2 организовать расширенную индексацию.

15) Перенести некоторые директивы (например, директивы для определения прав доступа, служебной индексации и т.п.) из основного конфигурационного файла в файлы .htaccess, расположенные непосредственно в каталогах, для которых выполняются настройки.

16) Удалить web-сервер Apache (Пуск/Панель управления/Установка и удаление программ).

17) Удалить каталог disc:\webprog.

Содержание отчета (отчет в бумажном виде):

- цель работы;
- задание;
- порядок выполнения лабораторной работы;
- дерево созданных каталогов;
- конфигурационные файлы;
- файлы с именами и группами пользователей;
- выводы по работе.

Теоретические сведения

Web-сервер Apache

Apache — один из популярных web-серверов в мире. В настоящее время программное обеспечение Apache установлено более чем на половине серверов.

Для настройки web-сервера Apache используются конфигурационные файлы:

– основной конфигурационный файл httpd.conf, расположенный в каталоге conf;

– дополнительные конфигурационные файлы, расположенные в каталоге conf\extra и подключаемые основному конфигурационному файлу httpd.conf по мере необходимости с помощью директивы Include;

– конфигурационные файлы .htaccess, расположенные непосредственно в каталогах, для которых выполняются настройки.

В конфигурационных файлах с помощью директив определяется, как web-сервер должен работать с ресурсами, отвечая на запрос, указывается, с какими файлами пользователи могут выполнять определенные операции. Настройка конфигурационных файлов web-сервера Apache — самый ответственный шаг после его установки.

Web-сервер Apache читает основной конфигурационный файл httpd.conf однократно при запуске. Если web-сервер работает, то при изменении конфигурационного файла httpd.conf следует перезапустить web-сервер.

В конфигурационном файле httpd.conf и файлах .htaccess содержатся директивы, которые управляют работой web-сервера Apache.

В конце основного конфигурационного файла httpd.conf перечислены директивы Include, позволяющие подключить дополнительные конфигурационные файлы из каталога conf\extra.

Виртуальные хосты

Web-сервер Apache позволяет настроить виртуальные хосты.

Виртуальные хосты позволяют разместить более чем один web-сайт, используя один экземпляр web-сервера. Виртуальный хост может быть как «привязанным к IP-адресу» (IP-based), что позволяет использовать отдельный IP-адрес для каждого web-сайта, так и «привязанным к имени» (name-based), что позволяет использовать один и тот же IP-адрес для нескольких web-сайтов, различая виртуальных хосты по именам или номерам портов.

Для организации виртуальных хостов используются директивы Listen, NameVirtualHost и блочная директива VirtualHost (все примеры приведены для name-based виртуального хоста, определяемого номером порта и web-сервера Apache, установленного как консольное приложение).

Директива Listen задает номер порта, который «слушает» web-сервер. В конфигурационном файле может присутствовать несколько директив Listen.

```
Listen 8081
```

Директива NameVirtualHost позволяет создать name-based виртуальный хост со своим номером порта.

```
NameVirtualHost 127.0.0.1:8081
```

Блочная директива <VirtualHost> позволяет задать директивы, определяющие режимы работы виртуального хоста.

```
<VirtualHost 127.0.0.1:8081>
CustomLog …/…/access.log common
ErrorLog …/…/error.log
DocumentRoot …/…/www
<Directory …/…/www>
Options …
…
</Directory>
<Files …/…/test.html>
…
</Files>
…
</VirtualHost>
```

Для настройки виртуального хоста можно использовать практически все директивы web-сервера Apache. Узнать, разрешена ли директива для использования в блочной директиве </VirtualHost> можно в локальной документации, доступной по адресу http://127.0.0.1:8080/manual. Директиву разрешено использовать в блочной директиве </VirtualHost> в случае, если в описании директивы в разделе Context указан virtual host.

Рекомендуется для каждого виртуального хоста с помощью директивы DocumentRoot задавать отдельный каталог для документов web-сайта, так как именно по этой причине и создаются виртуальные хосты.

Файлы регистрации доступа и ошибок могут быть одними и теми же для нескольких виртуальных хостов.

Блочные директивы <Directory> и <Files> предназначены для задания директив, применяемых к соответствующим каталогам и файлам (например, для организации доступа к каталогу или файлу).

Организация доступа

Для организации доступа к каталогам и файлам используются директивы Allow, Deny, AuthType, AuthName, AuthGroupFile, AuthUserFile и Require.

Директивы Allow, Deny позволяют открыть / закрыть доступ для всех пользователей или пользователям, пришедшим с определенного хоста, домена или IP-адреса.

```
Allow from all / Deny from all
Allow from apache.org / Deny from apache.org
Allow from .net / Deny from .net
Allow from 192.168.1.104 / Deny from192.168.1.104
Allow from 192.168 / Deny from192.168
```

Порядок применения директив Allow и Deny определяется директивой Order.

```
Order Deny,Allow
#Если клиент упомянут в директиве Deny, ему запрещается доступ при условии,
#что он не упомянут в директиве Allow. Если ни в одной из директив клиент
#не упомянут, доступ ему разрешается.

Order Allow,Deny
#Доступ клиенту, который упомянут в директиве Allow, разрешен,
#если только он не упомянут в директиве Deny. Если ни в одной из директив
#клиент не упомянут, доступ ему запрещается.
```

Директивы AuthType, AuthName, AuthGroupFile, AuthUserFile и Require позволяют открыть / закрыть доступ для зарегистрированных пользователей.

Директива AuthType задает тип контроля полномочий.

```
AuthType Basic
```

Директива AuthName задает область, в которой действительны имена и пароли пользователей.

```
AuthName Test
```

Директивы AuthGroupFile и AuthUserFile задают имена текстовых файлов, в которых содержится информация о группах и пользователях, входящих в группы и именах пользователях и паролях. Файлы имеют следующий формат:

```
group1:user1 user2 …
group2:user3 user4 …
…

user1:password1
user2:password2
…
```

Пароли пользователей могут храниться как в незашифрованном, так и в зашифрованном виде. Для шифрования паролей используется утилита bin\htpasswd.exe. Для получения справочной информации по работе с утилитой следует запустить утилиту с ключом –?.

Имена файлов групп и пользователей выбираются произвольно, как и их расположение, единственное соображение безопасности заключается в том, что каталог с файлами лучше располагать выше каталога, заданного директивой DocumentRoot.

Директива Require определяет права доступа для отдельных пользователей, групп пользователей, всех зарегистрированных пользователей.

```
Require user user1 user2 …
#доступ разрешен перечисленным пользователям

Require group group1 group2 …
#доступ разрешен перечисленным группам пользователей

Require valid-user
#доступ разрешен всем зарегистрированным пользователям
```

Служебная индексация

В случае если каталог, заданный директивой DocumentRoot, не содержит индексного файла (директива DirectoryIndex), web-сервер Apache создает служебный индексный файл. Параметр Indexes директивы Options разрешает формирование служебного индексного файла.

Для изменения вида служебного индексного файла можно включить расширенную индексацию директивой IndexOptions.

```
IndexOptions FancyIndexing
```

Для расширенной индексации можно использовать директивы AddIcon, AddDescription, HeaderName, ReadmeName, IndexIgnore (описание директив для расширенной индексации см. в локальной документации в разделе «Reference Manual/Run-time Configuration Directives»).

Директивы включения на стороне сервера (SSI — Server Side Includes)

Директивы SSI, содержащиеся в документах, позволяют реализовать на серверной стороне выполнение некоторых действий и включить результаты этих действий в документы перед их отправкой клиенту. Аналогичные возможности, гораздо более широкие, предоставляют серверные скриптовые языки.

Формат директивы:

```
<!--#directive attribute=value attribute=value … -->
```

Директивы SSI оформляются как комментарии.

Для настройки web-сервера Apache для работы с директивами SSI используются директивы Options, AddOutputFilter и AddType.

```
Options Includes
#разрешает использование директив SSI
```

```
AddOutputFilter INCLUDES .ssi
#задает соответствие между расширением имени файла и фильтром,
#который будет обрабатывать ответ сервера перед отправкой клиенту

AddType text/html .ssi
#задает соответствие между расширением имени файла и media-типом
```

Описание директив включения на стороне сервера см. в локальной документации (расположено в разделе «Server Side Includes (SSI)/Basic SSI directives»).

Практическая работа №5. Динамический html-документ

(4 часа)

Цель работы

Изучить основы клиентского скриптового языка JavaScript, работу с объектной моделью документа DOM (Document Object Model), познакомиться с возможностями, предоставляемые фреймворком jQuery.

Задание

Создать клиентский скрипт на языке JavaScript, выполняющий действия в соответствии с вариантом. Использовать возможности, предоставляемые объектной моделью документа DOM, использовать фреймворк jQuery (или аналогичный).

Вариант 1:

Игра на внимание и скорость реакции. Дана квадратная таблица с числами от 1 до N, расположенными в случайном порядке в ячейках таблицы. Цифры чисел в разных ячейках таблицы имеют разные цвет и размер. Игрок должен последовательно щелкнуть на всех числах за заданный интервал времени (пропорциональный размеру таблицы). При правильно выполненном щелчке цвет фона ячейки изменяется.

Вариант 2:

Калькулятор цвета. Отобразить таблицу, фоны ячеек которой окрашены в web-гарантированные цвета. По щелчку левой кнопки мыши на образце цвета изменяется цвет текста документа, по щелчку правой кнопки мыши — цвет фона документа, также появляется окно с шестнадцатеричным кодом цвета. Предусмотреть три поля для задания цветовых составляющих и отображения цвета, в отдельном, например, окне. Среди прочего использовать возможности, предоставляемые фреймворком jQuery.

Вариант 3:

Игра «Жизнь».

Игра моделирует жизнь поколений гипотетической колонии живых клеток на прямоугольном игровом поле, которые выживают, размножаются или погибают в соответствии со следующими правилами.

Для каждого поколения (шага игры) применяются следующие правила: каждая живая клетка, количество соседей которой меньше двух или больше трёх, погибает; каждая живая клетка, у которой от двух до трёх соседей, живёт до следующего хода; каждая мёртвая клетка, у которой есть ровно три соседа, оживает. Соседи клетки – это все соседние с ней клетки по горизонтали, вертикали и диагонали, всего восемь соседей.

Правила применяются ко всему игровому полю одновременно, а не к каждой из клеток по очереди. То есть подсчёт количества соседей происходит в один момент перед следующим шагом, и изменения, происходящие в соседних клетках, не влияют на новое состояние клетки.

Среди прочего использовать возможности, предоставляемые фреймворком jQuery.

Вариант 4:

Создание эффекта анимированного текста. В тексте символ за символом изменяется цвет и размер очередного символа. Предыдущий символ становится прежним. Предусмотреть возможность выбора основного и дополнительного цвета и размера символов. Среди прочего использовать возможности, предоставляемые фреймворком jQuery.

Вариант 5:

За указателем мыши перемещаются часы и дата (предусмотреть возможность установки часов и календаря). Среди прочего использовать возможности, предоставляемые фреймворком jQuery.

Вариант 6:

Тест на скорость реакции. После щелчка по кнопке в тестовом поле случайным образом, через случайные промежутки времени появляются изображения, по которым нужно успеть щелкнуть. Попадание обозначается каким-либо образом (например, «взрывом» изображения). Тестирование можно прекратить щелчком по кнопке, но не ранее, чем через некоторый отрезок времени. Выводится результат — процент удачных щелчков. Среди прочего использовать возможности, предоставляемые фреймворком jQuery.

Вариант 7:

Игра «Пятнадцать». Костяшки с числами передвигаются с помощью клавиш управления курсором. Исходно костяшки расставляются случайным образом. Правильная расстановка костяшек определяется программно.

Вариант 8:

Игра «Парные карточки». Игровое квадратное поле случайным образом заполнено парами карточек с одинаковыми изображениями. В начале игры некоторый интервал времени все карточки лежат изображениями вверх. Затем они «переворачиваются». Щелчком нужно «переворачивать» пары карточек. При переворачивании одинаковых карточек они исчезают. При переворачивании неодинаковых карточек они возвращаются в прежнее положение. Замерять время, потраченное игроком на поиск всех пар карточек.

Вариант 9:

Игра «Падающие мячи». По игровому полю сверху вниз в случайном порядке падают мячи, которые нужно ловить корзиной, передвигаемой горизонтально вдоль нижней границы игрового поля. Игру можно начать и прекратить щелчком по соответствующей кнопке. Со временем скорость падения мячей может увеличиваться. После остановки игры выводится

результат — процент пойманных мячей. Среди прочего использовать возможности, предоставляемые фреймворком jQuery.

Вариант 10:

Просмотр набора изображений со сменой подписей к изображениям с помощью кнопок «Назад» и «Далее». При просмотре первого изображения блокируется кнопка «Назад», при просмотре последнего — кнопка «Далее». Среди прочего использовать возможности, предоставляемые фреймворком jQuery.

Фото 1. Фудзияма.

Порядок выполнения лабораторной работы:

1) создать html-документ;
2) написать скрипт в соответствии с заданным вариантом. Для справки по языку Javascript можно использовать источники, расположенные по адресам http://learn.javascript.ru и http://javascript.ru. Для справки по фреймворку jQuery можно использовать источники, расположенные по адресам http://jquery.com и http://jquery-docs.ru;
3) протестировать созданный документ.

Содержание отчета:

- цель работы;
- задание;

- порядок выполнения лабораторной работы;
- разметка html-документа с исходным кодом скрипта;
- скриншот html-документа;
- выводы.

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА - Российский технологический университет»

РТУ МИРЭА

Институт Информационных Технологий
Кафедра инструментального и прикладного программного обеспечения
(ИППО)

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №1

по дисциплине
«Разработка клиент-серверных приложений»

Выполнил студент группы ИКБО-01-17 **Иванов И.И.**

Принял ассистент кафедры ИППО **Строганкова Н.В.**

Практическая работа выполнена «__»_____201__ г. *(подпись студента)*

«Зачтено» «__»_____201__ г. *(подпись руководителя)*

Москва 2019