МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«МОСКОВСКИЙ АВТОМОБИЛЬНО-ДОРОЖНЫЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (МАДИ)»
ВОЛЖСКИЙ ФИЛИАЛ МАДИ

УТВЕРЖДАЮ
Зав. кафедрой ИиТТП
Петрова А.В.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ ПО ДИСЦИПЛИНЕ «ИНТЕРНЕТ ПРОГРАММИРОВАНИЕ»

Направление подготовки
09.03.01 «Информатика и вычислительная техника»

Направленность (профиль)
Автоматизированные системы обработки информации и управления в отраслях транспортно-дорожного комплекса

Квалификация
Бакалавр

Кафедра: ИиТТП

Чебоксары

Оглавление

Лабораторная работа №1. Web-сервер Apache
(6 часов)

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 электронном виде):

- отчет сохранить в файле с именем АВТ-000 Иванов (лр1).doc;
- титульный лист (образец можно скачать по адресу http://ermak.cs.nstu.ru/webprog/wp_labwork_title_page.doc);
- 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»).

Лабораторная работа №2. Статический html-документ (2 часа)

Цель работы:

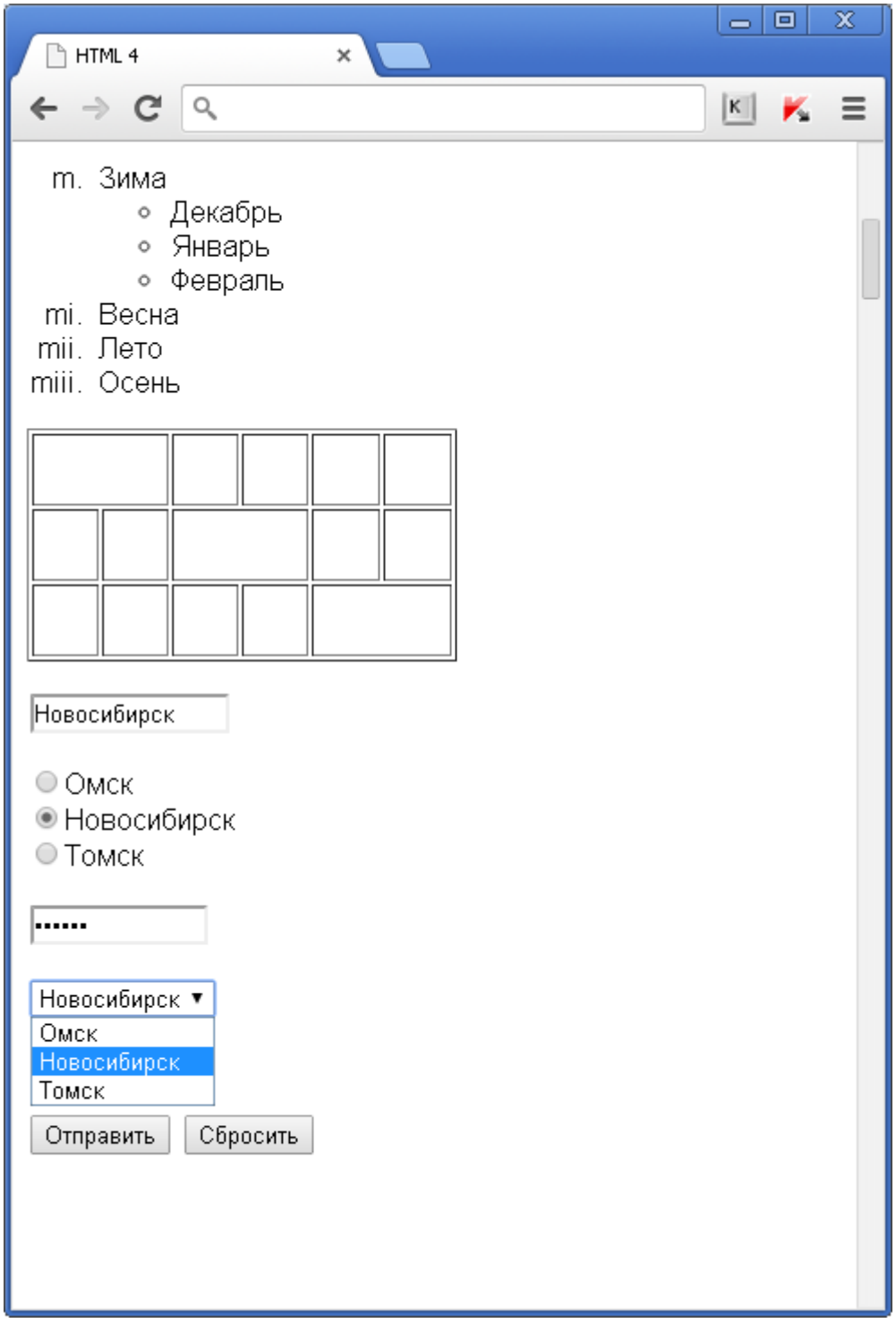
Изучить основы языка разметки гипертекста HTML 4.

Задание:

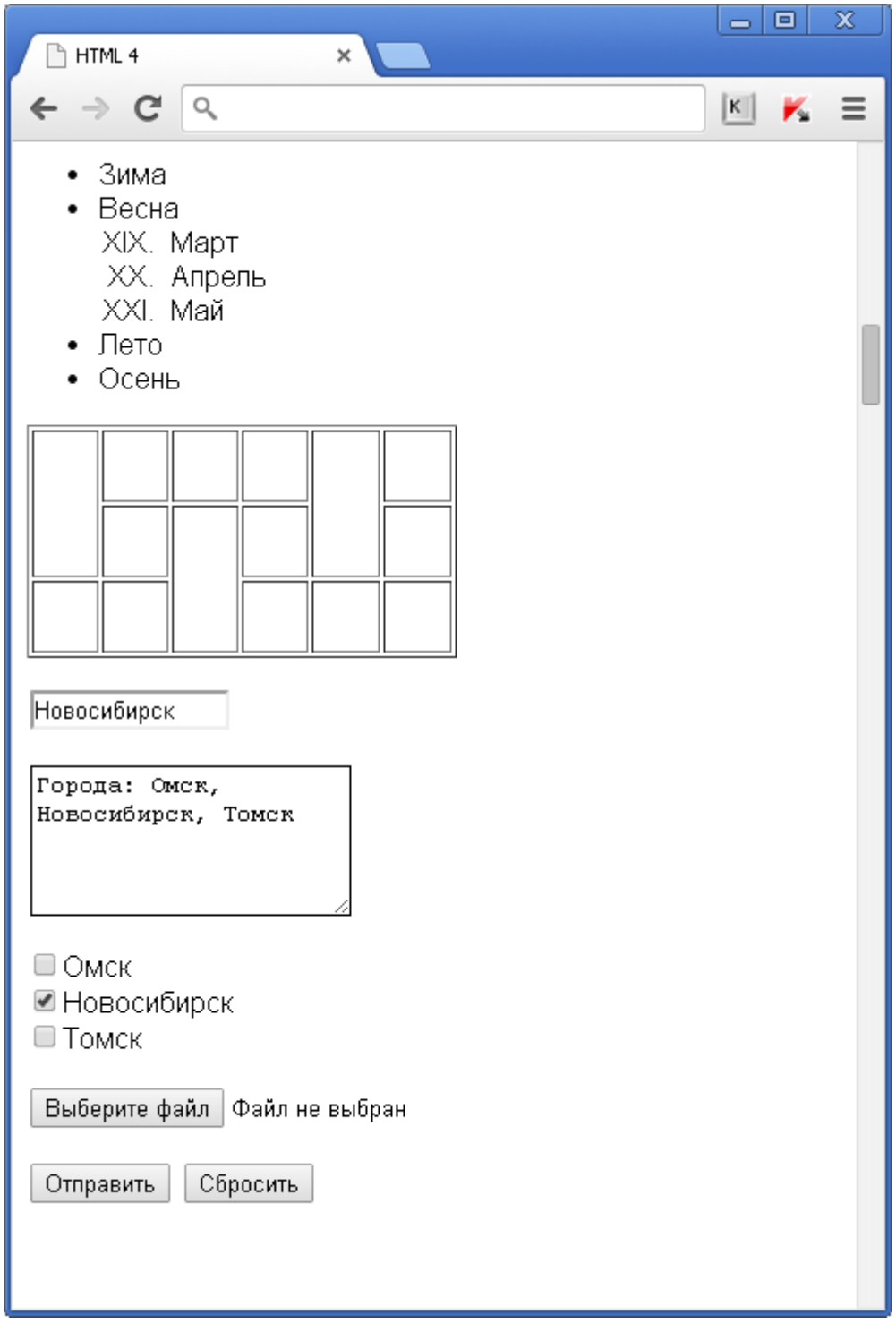
Создать html-документ, в разметке документа использовать:

- тег для определения кодировки кириллицы <meta>;
- тег комментария <!-- -->;
- теги форматирования текста: <p>,
, , , <hr>, <h1> ÷ <h6>, <b>, <i>, <u>, <sub>, <sup>, <pre>, <tt>, продемонстрировать отличия тегов <p> и
, и ;
- тег для разметки изображения <img>;
- тег для разметки гиперссылок <a>, разметить ссылки на другой документ, в пределах размечаемого документа, на email;
- с помощью параметров тега <body> изменить цвет фона документа, цвет текста, цвета непосещенных и посещенных ссылок документа, используя цвета из web-безопасной (гарантированной) палитры;
- теги для разметки списка, таблицы, формы в соответствии с вариантом.

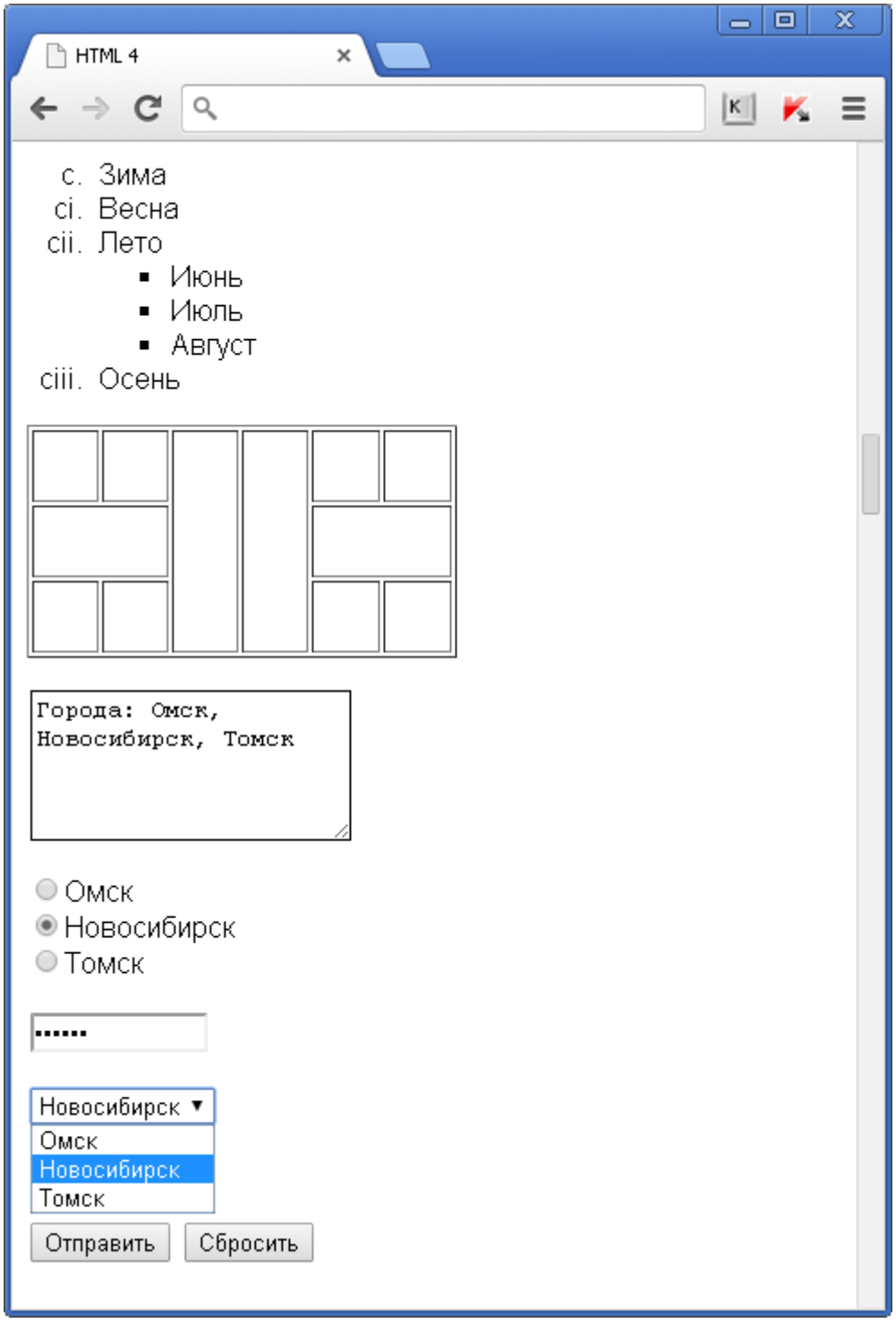
Вариант 1:


HTML 4
m. Зима
Декабрь
Январь
Февраль
mi. Весна
mii. Лето
miii. Осень
Новосибирск
Омск
Новосибирск
Томск
Новосибирск
Омск
Новосибирск
Томск
Отправить
Сбросить

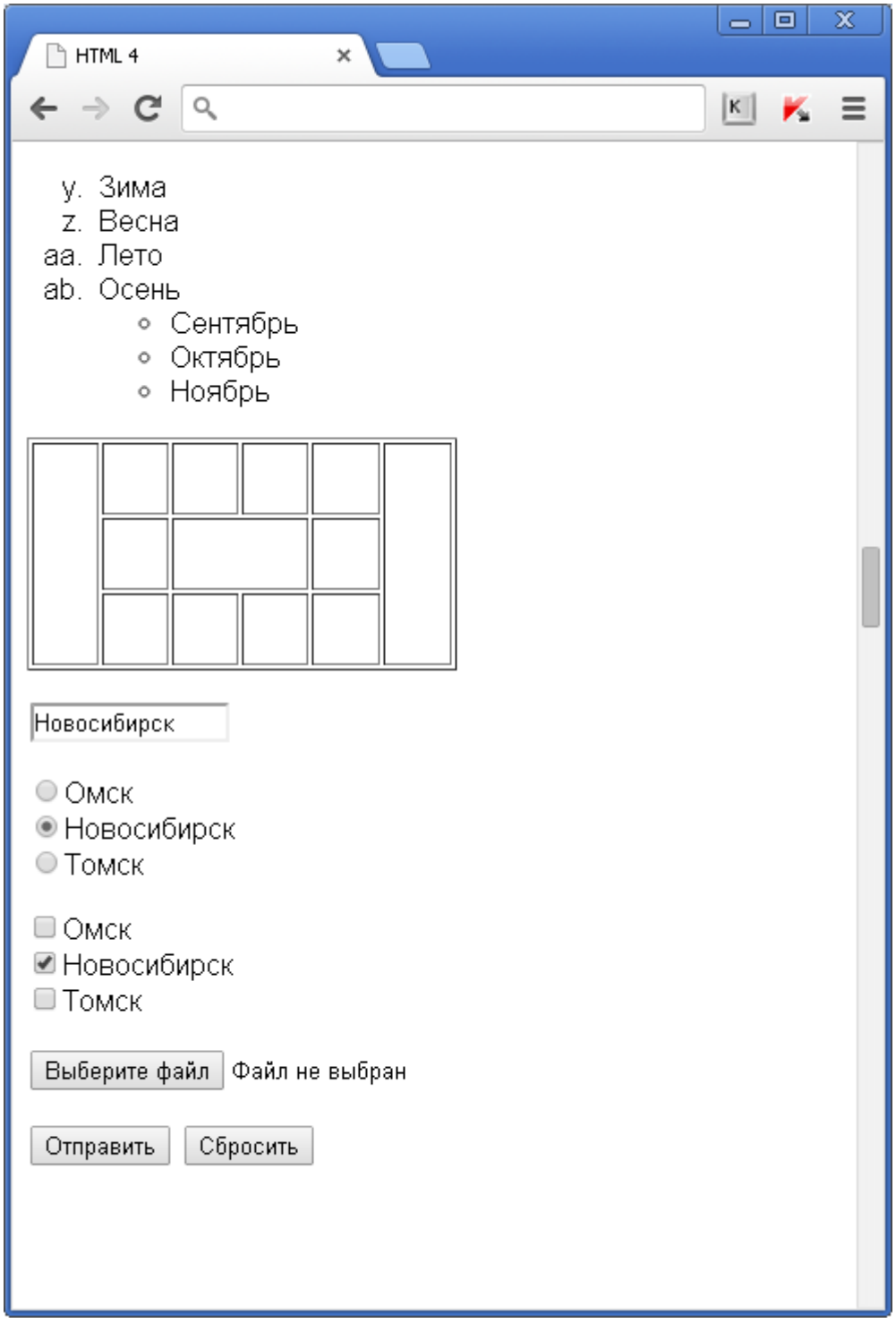
Вариант 2:


HTML 4
Зима
Весна
XIX. Март
XX. Апрель
XXI. Май
Лето
Осень
Новосибирск
Города: Омск, Новосибирск, Томск
Омск
Новосибирск
Томск
Выберите файл
Файл не выбран
Отправить
Сбросить

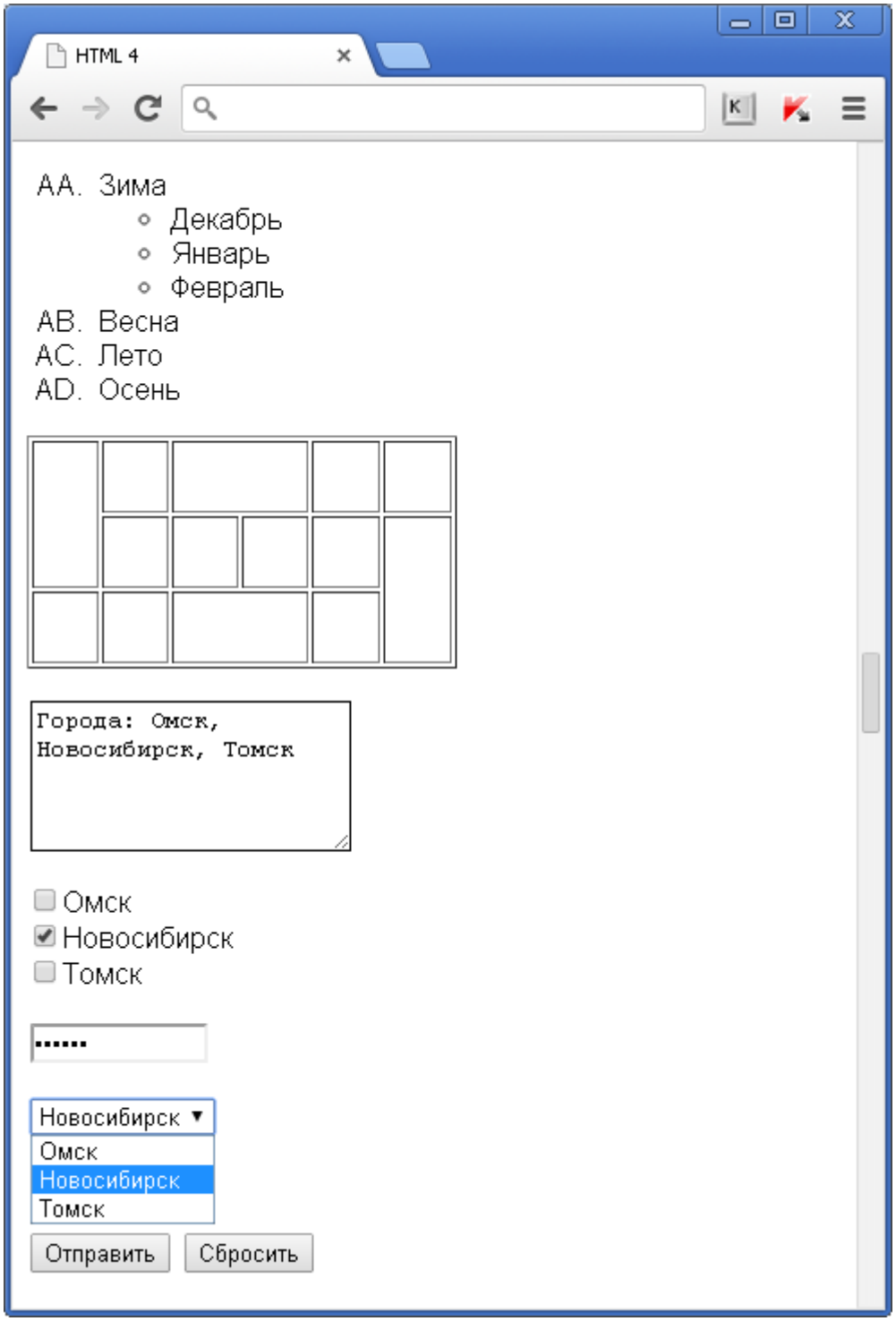
Вариант 3:


HTML 4
c. Зима
ci. Весна
cii. Лето
Июнь
Июль
Август
ciii. Осень
Города: Омск, Новосибирск, Томск
Омск
Новосибирск
Томск
Новосибирск
Омск
Новосибирск
Томск
Отправить
Сбросить

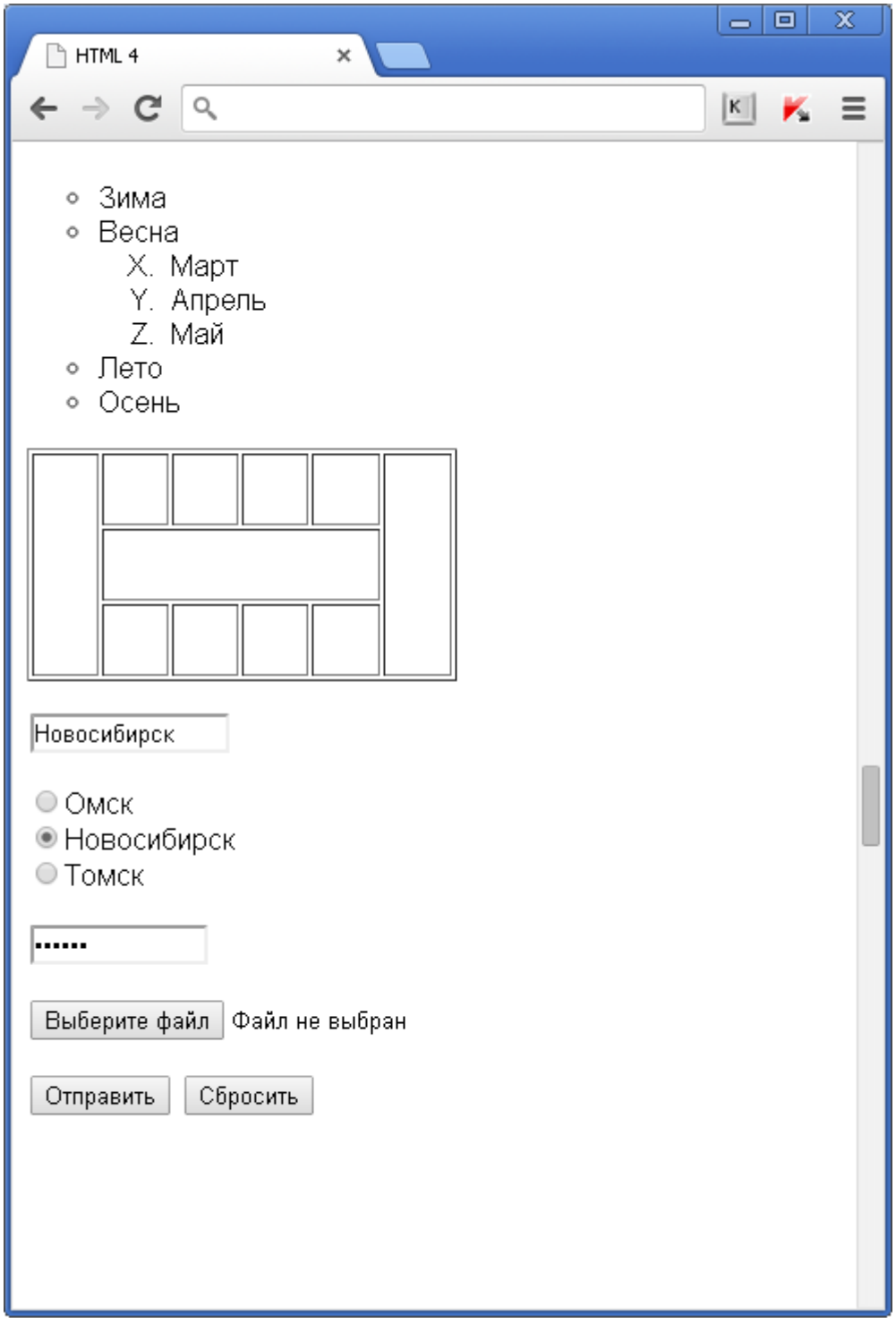
Вариант 4:


HTML 4
y. Зима
z. Весна
aa. Лето
ab. Осень
Сентябрь
Октябрь
Ноябрь
Новосибирск
Омск
Новосибирск
Томск
Омск
Новосибирск
Томск
Выберите файл
Файл не выбран
Отправить
Сбросить

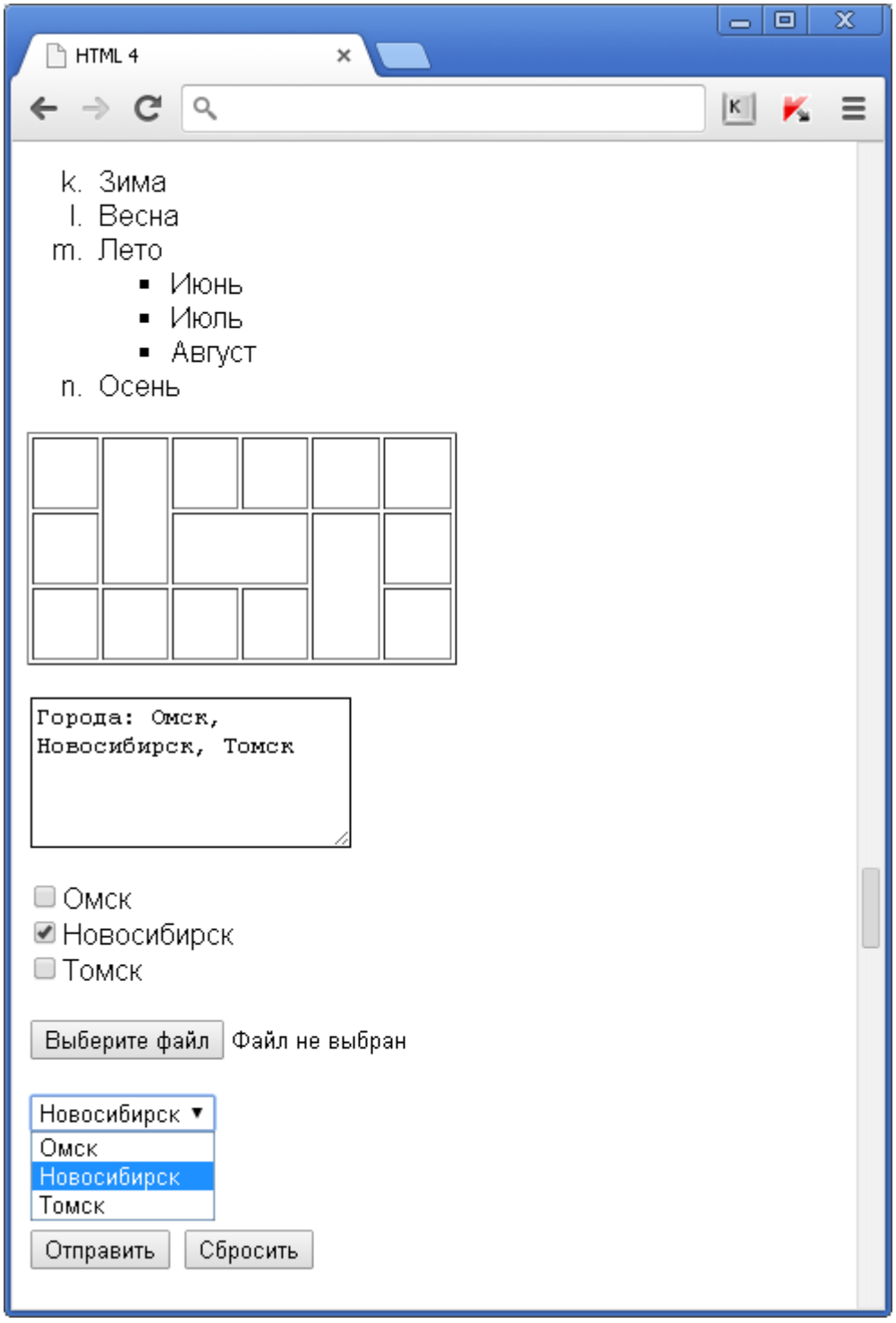
Вариант 5:


HTML 4
AA. Зима
Декабрь
Январь
Февраль
AB. Весна
AC. Лето
AD. Осень
Города: Омск, Новосибирск, Томск
Омск
Новосибирск
Томск
Новосибирск
Омск
Новосибирск
Томск
Отправить
Сбросить

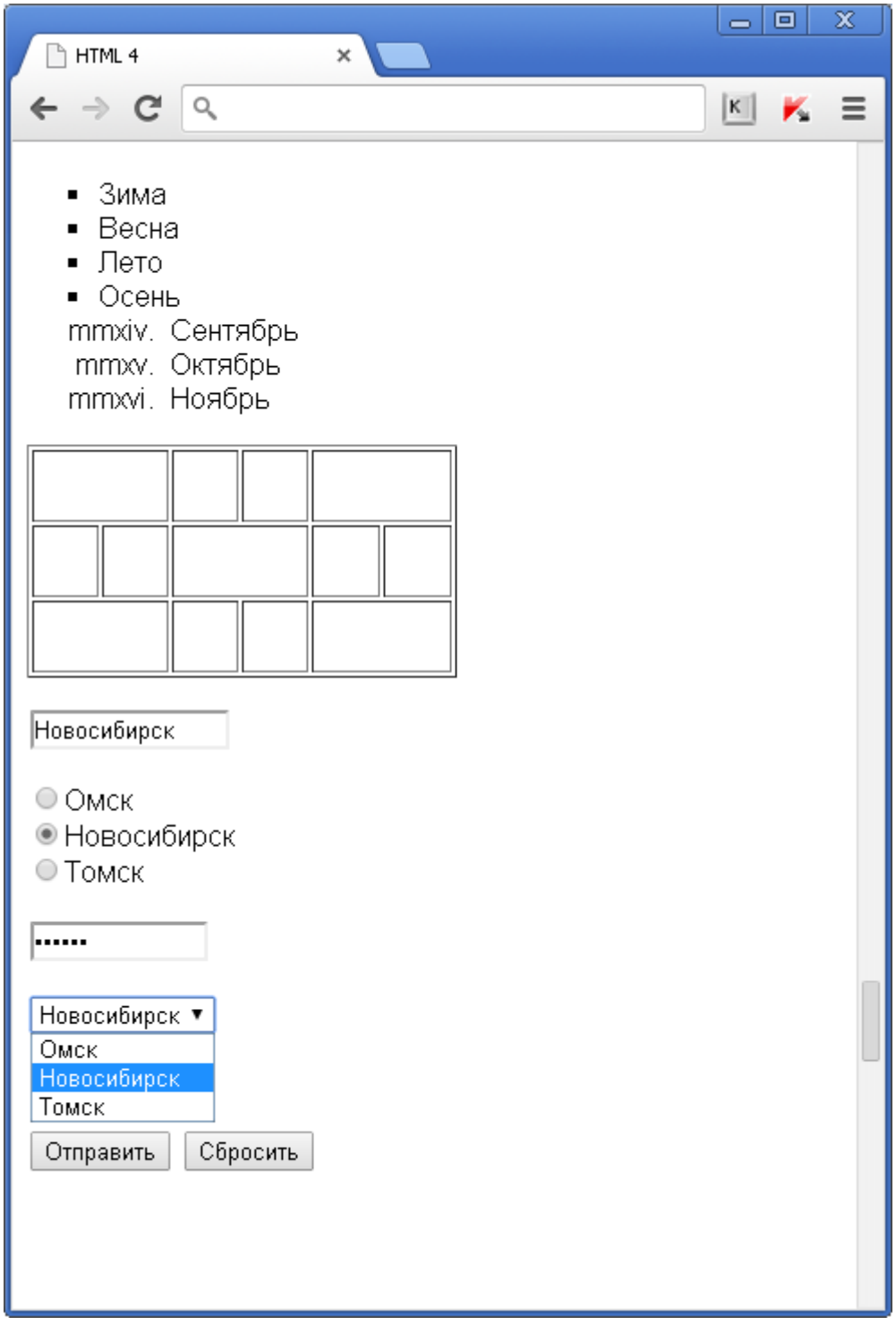
Вариант 6:


HTML 4
Зима
Весна
X. Март
Y. Апрель
Z. Май
Лето
Осень
Новосибирск
Омск
Новосибирск
Томск
Выберите файл
Файл не выбран
Отправить
Сбросить

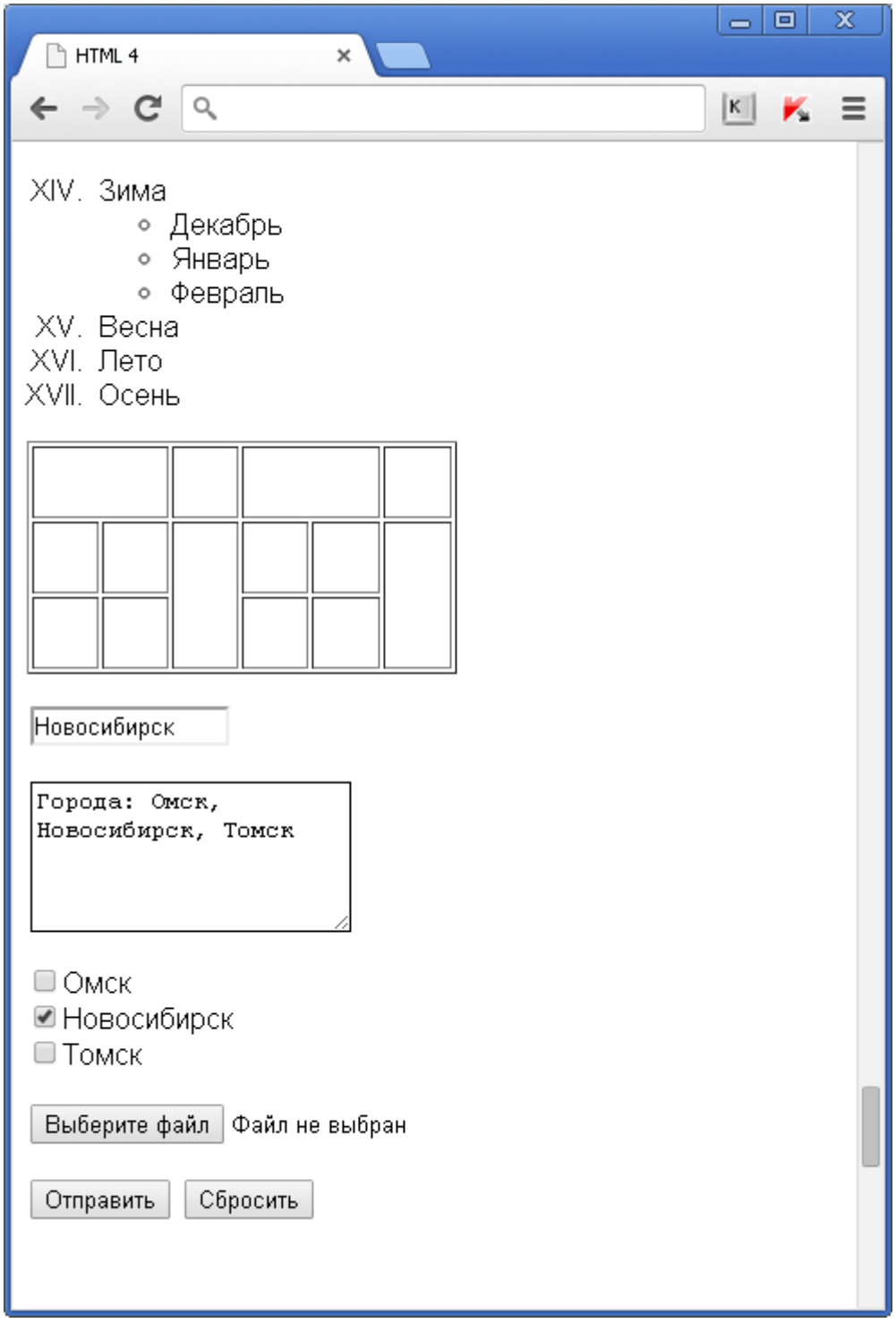
Вариант 7:


HTML 4
k. Зима
l. Весна
m. Лето
Июнь
Июль
Август
n. Осень
Города: Омск, Новосибирск, Томск
Омск
Новосибирск
Томск
Выберите файл
Файл не выбран
Новосибирск
Омск
Новосибирск
Томск
Отправить
Сбросить

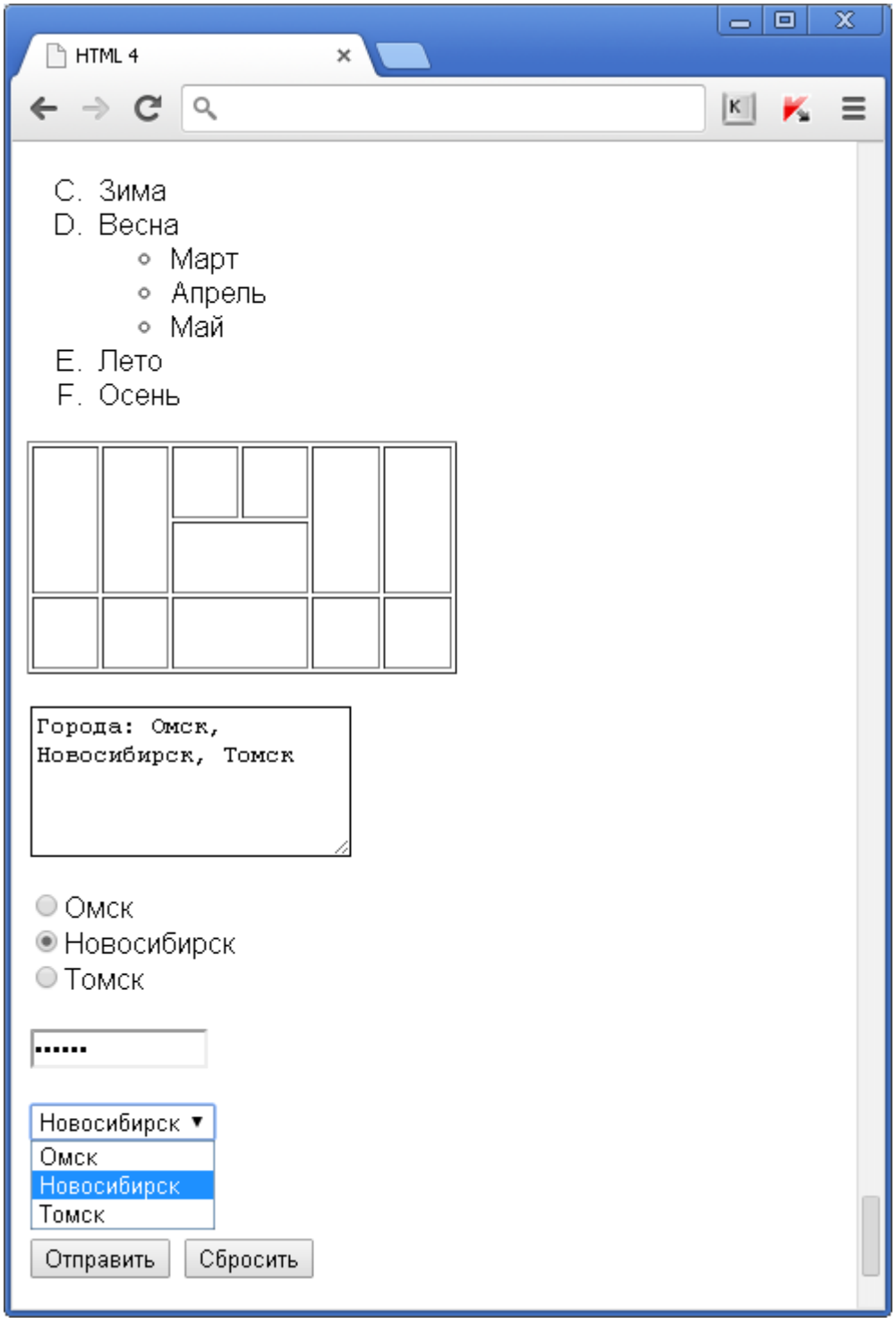
Вариант 8:


HTML 4
Зима
Весна
Лето
Осень
mmxiv. Сентябрь
mmxv. Октябрь
mmxvi. Ноябрь
Новосибирск
Омск
Новосибирск
Томск
Новосибирск
Омск
Новосибирск
Томск
Отправить
Сбросить

Вариант 9:


HTML 4
XIV. Зима
Декабрь
Январь
Февраль
XV. Весна
XVI. Лето
XVII. Осень
Новосибирск
Города: Омск, Новосибирск, Томск
Омск
Новосибирск
Томск
Выберите файл
Файл не выбран
Отправить
Сбросить

Вариант 10:


HTML 4
C. Зима
D. Весна
Март
Апрель
Май
E. Лето
F. Осень
Города: Омск, Новосибирск, Томск
Омск
Новосибирск
Томск
Новосибирск
Омск
Новосибирск
Томск
Отправить
Сбросить

Порядок выполнения лабораторной работы:

1. Запустить текстовый редактор.
2. Разметить html-документ в соответствии с заданным вариантом. Для справки о тегах и их атрибутах можно использовать справочник, расположенный по адресу http://htmlbook.ru.
3. Протестировать созданный документ, по крайней мере, в двух браузерах, отметить различия в отображении документа.

Содержание отчета (отчет в электронном виде):

- отчет сохранить в файле с именем АВТ-000 Иванов (лр2).doc;
- титульный лист (образец можно скачать по адресу http://ermak.cs.nstu.ru/webprog/wp_labwork_title_page.doc);
- цель работы;
- задание;
- порядок выполнения лабораторной работы
- html-разметка созданного документа;
- скриншоты html-документа для различных браузеров;
- выводы по работе.

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.

Спецификация языка разметки гипертекста HTML 4.01 расположена на сайте WWW-консорциума по адресу http://www.w3.org/TR/1999/REC-html401-19991224.

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

Цель работы:

Изучить основы каскадных таблицы стилей CSS 3.

Задание:

Изменить html-документ, полученный в результате выполнения лабораторной работы №2 «Создание статического html-документа», изменив в нем с помощью каскадных таблиц стилей:

- текст (шрифт, размер, цвет, поля, обрамление);
- гиперссылки (цвет непосещенных и посещенных ссылок);
- документ (фон);
- список (маркеры или нумерацию);
- таблицу (границы, фон).

Использовать определение стилей для тегов и классы стилей, псевдоклассы.

Использовать три способа определения каскадных таблиц стилей:

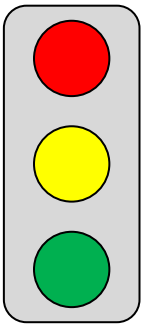
- с помощью тега <link>;
- с помощью тега <style>;
- с помощью параметра style тега.

Продемонстрировать действие приоритетов при применении различных способов определения CSS;

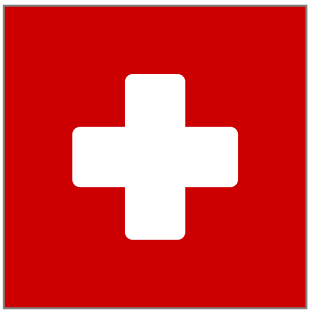
Создать два слоя, частично перекрывающих друг на друга.

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

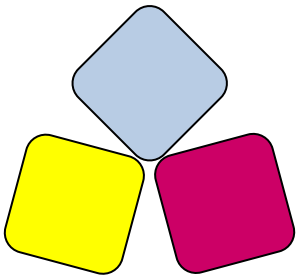
Вариант 1:



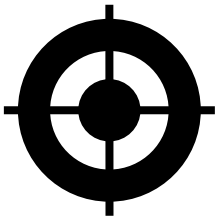
Вариант 2:



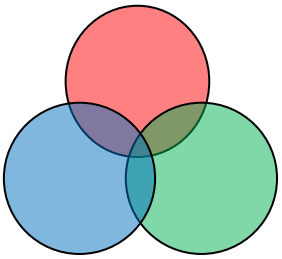
Вариант 3:



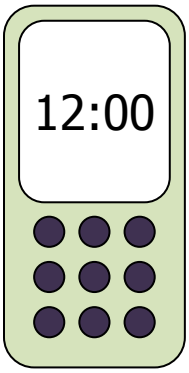
Вариант 4:



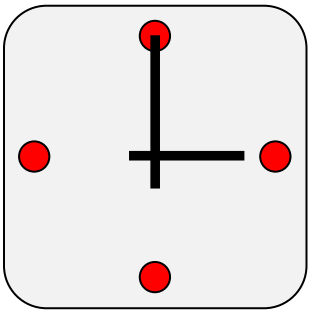
Вариант 5:



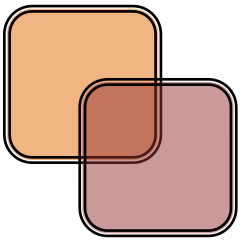
Вариант 6:



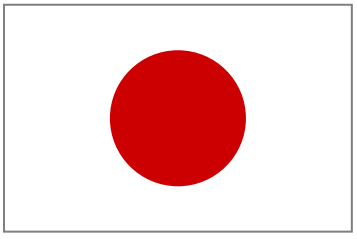
Вариант 7:



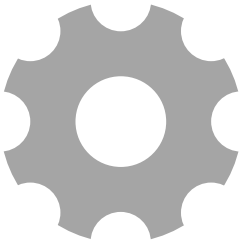
Вариант 8:



Вариант 9:



Вариант 10:



Порядок выполнения лабораторной работы:

1. Запустить текстовый редактор.
2. Изменить с помощью каскадных таблиц стилей html-документ в соответствии с заданным вариантом. Для справки о свойствах и их значениях можно использовать справочник, расположенный по адресу http://htmlbook.ru.
3. Протестировать созданный документ в браузере.

Содержание отчета (отчет в электронном виде):

- отчет сохранить в файле с именем АВТ-000 Иванов (лр3).doc;
- титульный лист (образец можно скачать по адресу http://ermak.cs.nstu.ru/webprog/wp_labwork_title_page.doc);
- цель работы;
- задание;
- порядок выполнения лабораторной работы
- код созданного html-документа, включая созданные стили;
- скриншот html-документа;
- выводы по работе.

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

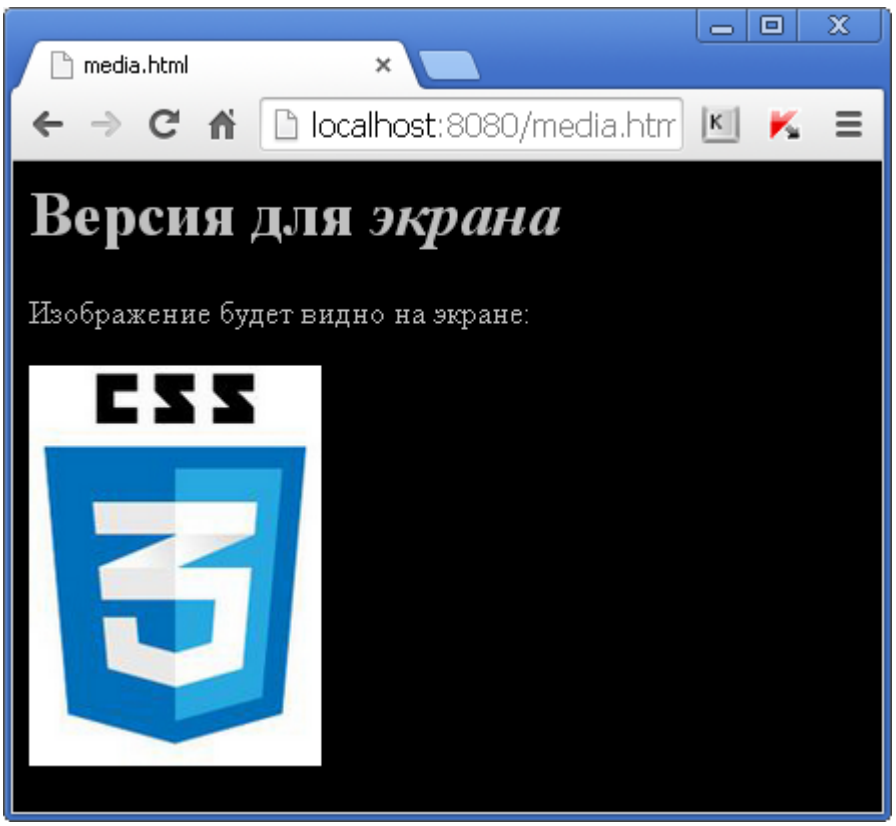
```
body
        {color:silver; background:black}
.forprint
        {display:none}
```

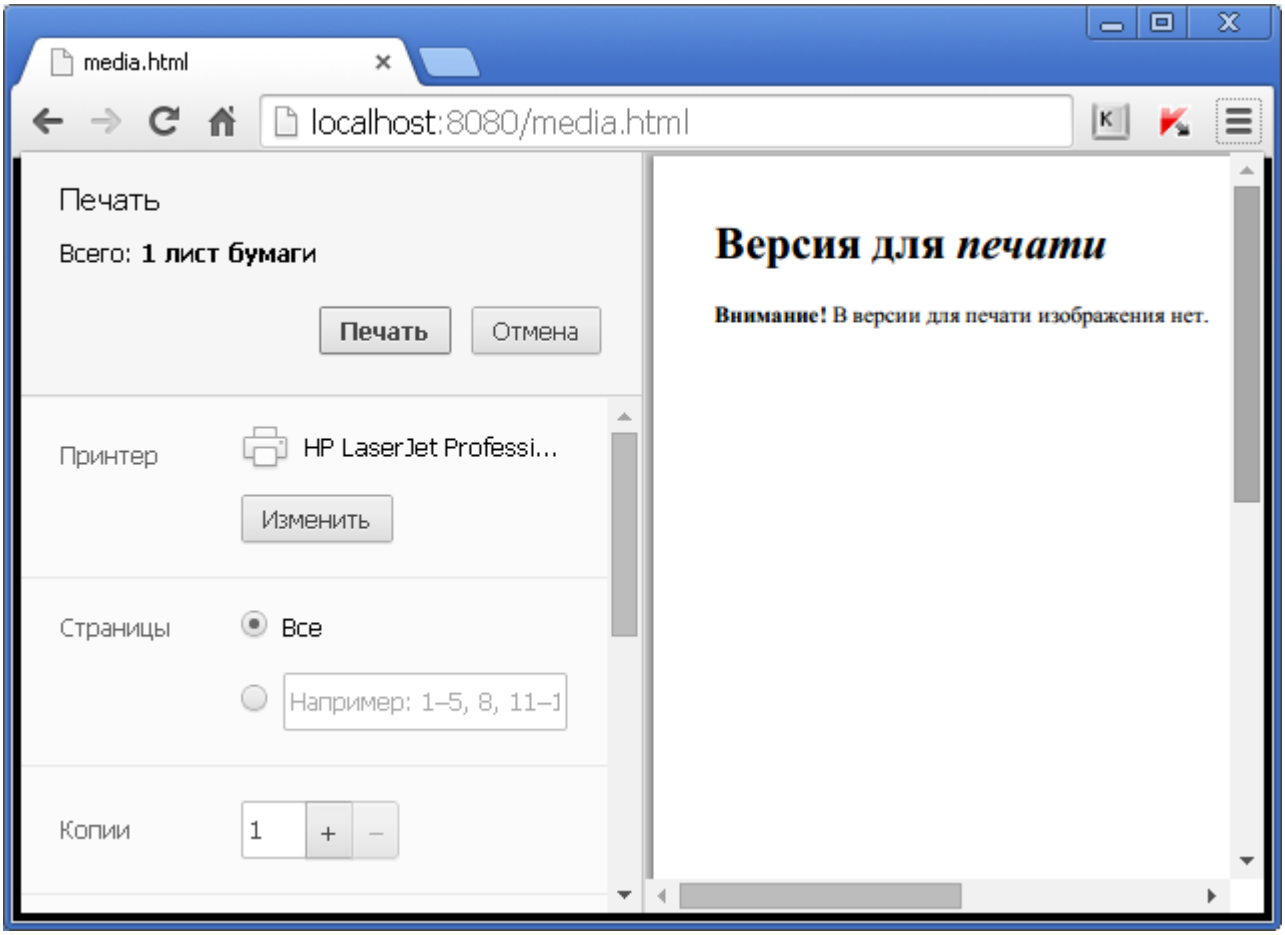
Файл print.css

```
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.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.

Лабораторная работа №4. Динамический html-документ (4 часа)

Цель работы:

Изучить основы клиентского скриптового языка JavaScript, работу с объектной моделью документа DOM (Document Object Model), работу с cookie, 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 аналог).

Вариант 1:

Сборка мозаики. Элементы мозаики перетаскиваются указателем мыши. Предусмотреть возможность автоматической сборки. Положение элементов в собранной мозаике фиксировано. Среди прочего использовать возможности, предоставляемые фреймворком jQuery.

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 даты). 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:

Три линейки с бегунками для каждой цветовой составляющей. Изменение положения каждого из бегунков влечет за собой изменение цвета фона документа. Среди прочего использовать возможности, предоставляемые фреймворком jQuery.

Вариант 8:

Калькулятор на четыре действия (с нажимающимися кнопками). Среди прочего использовать возможности, предоставляемые фреймворком jQuery.

Вариант 9:

Игра «Падающие мячи». По игровому полю сверху вниз в случайном порядке падают мячи, которые нужно ловить корзиной, передвигаемой с помощью клавиатуры горизонтально вдоль нижней границы игрового поля. Игру можно начать и прекратить щелчком по соответствующей кнопке. Со временем скорость падения мячей увеличивается. 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 (отчет в электронном виде):

- отчет сохранить в файле с именем АВТ-000 Иванов (лр4).doc;
- титульный лист (образец можно скачать по адресу http://ermak.cs.nstu.ru/webprog/wp_labwork_title_page.doc);
- цель работы;
- задание;
- порядок выполнения лабораторной работы
- разметка html-документа с исходным кодом скрипта;
- скриншот html-документа;
- выводы по работе.

Теоретические сведения

Теоретические материалы доступны по адресу http://learn.javascript.ru.

Лабораторная работа №5.
CGI-скрипт.
(4 часа)

Цель работы:

Получить практические навыки в написании и отладке CGI-скрипта на языке программирования, имеющем средства для работы с интерфейсом CGI.

Задание:

Во всех вариантах заданий необходимо разработать CGI-скрипт, реализующий некоторый тест, и два счетчика выполнения теста — общий счетчик, значение которого хранится на серверной стороне, и счетчик конкретного посетителя, значение которого хранится на клиентской стороне в виде cookie.

Тест должен содержать не менее трех вопросов с не менее чем тремя вариантами ответа на каждый вопрос.

Данные, введенные пользователем, пересылаются на серверную сторону, обрабатываются CGI-скриптом, который «на лету» формирует документ с результатами прохождения теста и новыми значениями счетчиков прохождения теста.

CGI-скрипт следует написать так, чтобы он мог принимать данные, присланные как методом GET, так и методом POST.

Вариант 1:

Проверка знаний правил дорожного движения.

Вариант 2:

Проверка знания таблицы умножения.

Вариант 3:

Психологический тест.

Вариант 4:

Проверка знания языка разметки гипертекста HTML.

Вариант 5:

Проверка знания каскадных таблиц стилей CSS.

Вариант 6:

Проверка словарного запаса иностранного языка.

Вариант 7:

Проверка знания языка программирования JavaScript.

Вариант 8:

Проверка знания директив web-сервера Apache.

Вариант 9:

Проверка знания языка программирования C++.

Вариант 10:

Проверка знания языка программирования PHP.

Порядок выполнения лабораторной работы:

4. Для выполнения лабораторной работы установить и настроить web-сервер или воспользоваться программным комплексом Denwer (http://denwer.ru).
5. Создать html-документ с формой.
6. Написать CGI-скрипт в соответствии с заданным вариантом.
7. Протестировать созданный CGI-скрипт (при тестировании использовать методы передачи данных GET и POST).

Содержание отчета (отчет в электронном виде):

- отчет сохранить в файле с именем АВТ-000 Иванов (лр5).doc;
- титульный лист (образец можно скачать по адресу http://ermak.cs.nstu.ru/webprog/wp_labwork_title_page.doc);
- цель работы;
- задание;
- порядок выполнения лабораторной работы
- разметка html-документа;
- исходный код скрипта;
- скриншоты html-документа с исходной формой и документом, сформированным CGI-скриптом;
- выводы по работе.

Теоретические сведения

CGI (Common Gateway Interface) – общий шлюзовой интерфейс

Один из способов формирования динамических html-документов (документов, создаваемых программно на серверной стороне «на лету») заключается в использовании CGI-скриптов.

CGI — это интерфейс, используемый для связи внешней программы, работающей на серверной стороне, с web-сервером.

Интерфейс CGI разработан таким образом, что для написания серверного CGI-скрипта можно использовать любой язык программирования, имеющий средства для работы со стандартными устройствами ввода/вывода.

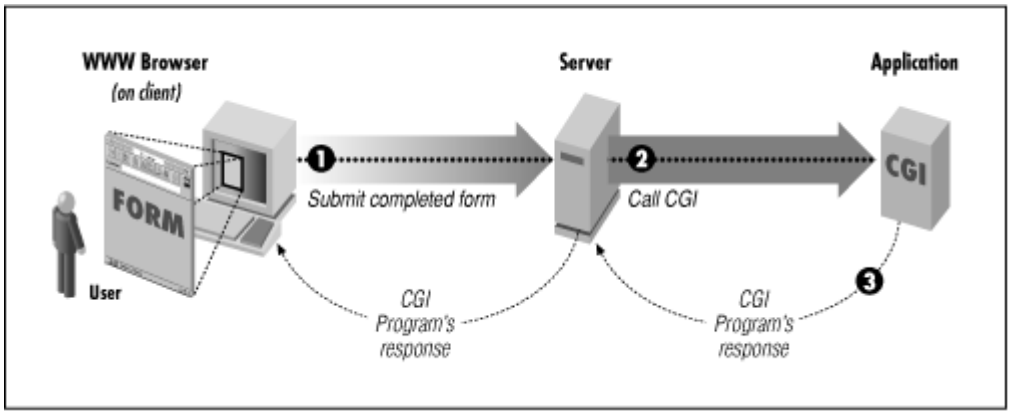
CGI-скрипт, как правило, помещается в каталог cgi (или cgi-bin) web-сервера, но это требование необязательно, так как CGI-скрипт может располагаться в любом каталоге, но при этом большинство web-серверов требуют дополнительной настройки.

CGI-скрипт, использующий CGI-интерфейс, получает информацию от клиента, обрабатывает ее, и возвращает результат (динамически сформированный html-документ, гиперссылку на существующий html-документ, графическое изображение и т.д.) Так как CGI-скрипт — это программа, она должна быть оттранслирована для той операционной системы, под управлением которой работает web-сервер.

На стороне клиента отображается форма, размеченная тегом <form>, содержащая некоторые поля для ввода данных и кнопку для отсылки данных. После заполнения полей и нажатия кнопки данные в запросе клиента пересылаются на сторону сервера, где web-сервер передает присланные данные CGI-скрипту, используя CGI.

После обработки полученных данных CGI-скрипт создает документ и передает его web-серверу, который в ответе сервера возвращает документ на сторону клиента.

Передача информации от клиента к серверу и передача сформированного документа от сервера к клиенту изображена на рисунке.



1 — клиент формирует запрос, включая в него данные, внесенные в поля формы, запрос отсылается web-серверу.

2 — web-сервер, используя CGI, передает присланные в запросе данные CGI-скрипту.

3 — CGI-скрипт на основе данных формирует документ, возвращает его web-серверу, который, в свою очередь, формирует ответ сервера, включая в него документ, созданный CGI-скриптом, и возвращает ответ клиенту.

Для создания формы используется тег <form>.

```
<form action=URL method=GET | POST>
…
</form>
```

Атрибут action определяет url GCI-скрипта, обрабатывающего присланные данные.

Атрибут method определяет метод передачи данных. По умолчанию используется метод get.

Метод GET

Метод GET предполагает передачу данных GCI-скрипту через переменные среды (environment variables), устанавливаемые на стороне сервера.

Для передачи данных, присланных методом GET, используется переменная QUERY_STRING. Значением переменной QUERY_STRING будет

строка, содержащая данные в формате name1=value1&name2=value2& … &nameN=valueN, где name — это имя поля формы, value — значение, определенное пользователем для поля формы.

Метод POST

При использовании метода POST GCI-скрипт получает присланные данные через стандартный поток ввода.

Объем переданных данных (в байтах) можно получить через переменную окружения CONTENT_LENGTH.

Формирование возвращаемого документа

Вне зависимости от метода передачи данных, GET или POST, результат своей работы GCI-скрипт должен направить в стандартный поток вывода.

При формировании возвращаемого документа GCI-скрипт должен предварить документ хотя бы одним заголовком ответа сервера, определяющим media-тип возвращаемого документа, заголовком Content-type. Заголовок должен быть отделен от собственно возвращаемого документа пустой строкой, как того требует структура ответа сервера.

Чаще всего GCI-скрипт используется для создания html-документов на основе данных, полученных от клиента. В этом случае заголовок ответа сервера должен определять media-тип возвращаемого документа как текст в формате html (Content-type: text/html), за которым необходимо вывести пустую строку, отделяющую заголовок от html-документа.

Web-сервер возвращает результат, сформированный GCI-скриптом, клиенту, возможно дополняя его статусной строкой и другими заголовками ответа сервера.

GCI-скрипт может сформировать полный ответ (со всеми заголовками ответа сервера). В этом случае web-сервер ничего не изменяет в результате работы GCI-скрипта, только пересылает его клиенту «как есть».

Пример: на стороне клиента в поля формы вносятся имя и возраст, в зависимости от возраста возвращаются разные приветствия (рассматриваются два варианта: для методов GET и POST).

Метод GET

HTML-документ, содержащий форму:

```
<html>
<form action=http://localhost/cgi/hello.exe method=get>
<p>ИМЯ<input type=text name=name>
<p>ВОЗРАСТ<input type=text name=age>
<p><input type=submit>
</form>
```

CGI-приложение (файл hello.cpp)

```
void main()
{
int age;
char *name=new char[256];
char *query_string=new char[256];
query_string=getenv("QUERY_STRING");

//query_string=”name=Maria&age=18”
//из строки извлекаются подстроки “Maria” и “18”
//и присваиваются переменным name и age соответственно

cout<<”Content-type: text/html\n\n”;
cout<<”<html>”;
if(age<=16) cout<<”Привет, ”;
if(age>16) cout<<”Здравствуйте, ”;
cout<<name<<”</html>”;
}
```

Метод POST

HTML-документ, содержащий форму:

```
<html>
<form action=http://localhost/cgi/hello.exe method=post>
<p>ИМЯ<input type=text name=name>
<p>ВОЗРАСТ<input type=text name=age>
<p><input type=submit>
</form>
```

CGI-приложение (файл hello.cpp)

```
void main()
{
int age;
char *name=new char[256];
int length=atoi(getenv("CONTENT_LENGTH"));
char * string=new char[length+1];
cin≫string;

//string=”name=Maria&age=18”
//из строки извлекаются подстроки “Maria” и “18”
//и присваиваются переменным name и age соответственно

cout<<”Content-type: text/html\n\n”;
cout<<”<html>”;
if(age<=16) cout<<”Привет, ”;
if(age>16) cout<<”Здравствуйте, ”;
cout<<name<<”</html>”;
}
```

Работа с cookie

Cookie устанавливаются на клиентской стороне web-сервером с помощью заголовка ответа сервера Set-Cookie.

Set-Cookie: name=value

В случае, если на клиентской стороне cookie уже установлены, информация о cookie пересылается на серверную сторону в запросе браузера-клиента в заголовке Cookie. Присланные cookie на серверной стороне присваиваются переменной окружения HTTP_COOKIE.

Лабораторная работа №6. PHP-скрипт.
(4 часа)

Цель работы:

Получить практические навыки в написании и отладке PHP-скрипта.

Задание:

Во всех вариантах заданий необходимо разработать PHP-скрипт, реализующий некоторый тест и счетчик выполнения теста.

Тест должен содержать не менее десяти вопросов с не менее чем тремя вариантами ответа на каждый вопрос. На некоторые вопросы может предлагаться несколько правильных вариантов ответов. Вопросы должны быть разделены на две темы.

Результаты теста должны отображаться в браузере и сохраняться в файле, доступном по ссылке на странице с результатами теста. Кроме результатов на странице и в файле должны быть указаны дата и время прохождения теста.

Вариант 1:

Проверка знаний правил дорожного движения.

Вариант 2:

Проверка знания таблицы умножения.

Вариант 3:

Психологический тест.

Вариант 4:

Проверка знания языка разметки гипертекста HTML.

Вариант 5:

Проверка знания каскадных таблиц стилей CSS.

Вариант 6:

Проверка словарного запаса иностранного языка.

Вариант 7:

Проверка знания языка программирования JavaScript.

Вариант 8:

Проверка знания директив web-сервера Apache.

Вариант 9:

Проверка знания языка программирования C++.

Вариант 10:

Проверка знания языка программирования PHP.

Порядок выполнения лабораторной работы:

1. Для выполнения лабораторной работы установить и настроить web-сервер Apache и интерпретатор PHP (интерпретатор PHP установить как модуль web-сервера Apache).
2. Создать html-документ с формой.
3. Написать PHP-скрипт в соответствии с заданным вариантом.
4. Протестировать созданный PHP-скрипт.

Содержание отчета (отчет в электронном виде):

- отчет сохранить в файле с именем АВТ-000 Иванов (лр6).doc;
- титульный лист (образец можно скачать по адресу http://ermak.cs.nstu.ru/webprog/wp_labwork_title_page.doc);
- цель работы;
- задание;
- порядок выполнения лабораторной работы
- разметка html-документа;
- исходный код скрипта;
- скриншоты html-документа с исходной формой и документом, сформированным PHP-скриптом;
- файл с результатами тестирования;
- выводы по работе.

Теоретические сведения

Установка интерпретатора PHP как модуля web-сервера Apache

Интерпретатор PHP может быть установлен для работы в двух режимах: как модуль web-сервера Apache или как обработчик CGI-скриптов.

Для установки интерпретатора PHP как модуля web-сервера Apache достаточно распаковать zip-архив с дистрибутивом, например, на диск C:\php и создать копию файла php.ini-production с именем php.ini в той же папке.

В файле php.ini можно выполнить настройки путем изменения параметров соответствующих директив.

Директива error_reporting задает уровень протоколирования ошибки. Параметр директивы может быть либо числом, либо именованной константой. Параметр E_ALL позволяет отображать предупреждения и ошибки всех уровней.

```
error_reporting = E_ALL
```

Директива extension позволяет загрузить необходимые динамические расширения.

```
extension=php_gd2.dll       ;для работы с графической библиотекой
extension=php_mysql.dll     ;для работы с СУБД MySQL
```

Директива display_errors позволяет выводить сообщения об ошибках на экран вместе с остальным выводом, либо скрывать сообщения об ошибках от пользователя. Для отладки скриптов рекомендуется использовать директиву display_errors с параметром On.

```
display_errors = On
```

После отладки скриптов предупреждения и сообщения об ошибках можно скрывать от пользователя, выводя их в файл, расположенный на стороне сервера. Директива error_log задает расположение файла с предупреждениями и сообщениями об ошибках.

```
display_errors = Off
error_log = c:\php\phperror.log
```

Директива short_open_tag определяет сокращенную или полную форму записи тега для вставки php-скрипта в html-разметку.

```
short_open_tag = Off        ;<?php … ?> и <script> … </script>
или
short_open_tag = On         ;дополнительно <? … ?>
```

Для настройки web-сервера Apache в основной конфигурационный файл httpd.conf следует добавить директивы LoadModule, AddHandler и PHPIniDir.

```
LoadModule php5_module “c:/php/php5apache2_2.dll”
AddHandler application/x-httpd-php .php
PHPIniDir “c:/php”
```

Установка интерпретатора PHP как обработчика CGI-скриптов

Для установки интерпретатора PHP как обработчика CGI-скриптов настройки файла php.ini выполняются также, как было описано выше, для настройки web-сервера Apache в основной конфигурационный файл httpd.conf следует добавить директивы ScriptAlias, AddType и Action.

```
ScriptAlias /php/ “c:/php/”
AddType application/x-httpd-php .php
Action application/x-httpd-php “/php/php-cgi.exe”
```

Теоретические материалы также доступны по адресу http://www.php.su/learnphp, http://phpclub.ru/manrus.

Лабораторная работа №7. Графическая библиотека PHP GD (4 часа)

Цель работы:

Получить практические навыки в использовании графической библиотеки PHP GRAPHICS DRAW (GD).

Задание:

Во всех вариантах заданий необходимо разработать PHP-скрипт, использующий возможности графической библиотеки PHP GD.

Вариант 1:

Форма для задания в аналитическом виде функции одного аргумента, начального и конечного значений аргумента и вывода графика функции (для оперирования аналитическим выражением функции использовать функцию create_function()).

Вариант 2:

Форма для голосования с выводом результатов в виде круговой диаграммы с указанием долей в процентах.

Вариант 3:

Лист календаря с текущим месяцем (с указанием дней недели). Для листа календаря использовать готовые изображения для каждого из месяцев.

Вариант 4:

Форма для голосования с выводом результатов в виде столбчатой 3d диаграммы с указанием долей в процентах.

Вариант 5:

Логотип PHP.

Вариант 6:

Форма для голосования с выводом результатов в виде столбчатой гистограммы с указанием долей в процентах.

Вариант 7:

Галерея изображений. В каталоге хранится набор файлов с полноразмерными изображениями любого формата, сформировать документ с миниатюрами-гиперссылками изображений (preview) формата jpeg.

Вариант 8:

Графический счетчик посещения сайта. Фон для значения счетчика выбирается случайным образом из набора графических файлов.

Вариант 9:

Аналоговые часы с фиксированным текущим значением временем.

Вариант 10:

Форма для голосования с выводом результатов в виде кольцевой диаграммы с указанием долей в процентах.

Порядок выполнения лабораторной работы:

1. Для выполнения лабораторной работы установить программный комплекс Denwer.
2. Написать PHP-скрипт в соответствии с заданным вариантом.
3. Протестировать созданный PHP-скрипт.

Содержание отчета (отчет в электронном виде):

- отчет сохранить в файле с именем АВТ-000 Иванов (лр7).doc;
- титульный лист (образец можно скачать по адресу http://ermak.cs.nstu.ru/webprog/wp_labwork_title_page.doc);
- цель работы;
- задание;
- порядок выполнения лабораторной работы
- исходный код скрипта;
- скриншот изображения, сформированного PHP-скриптом;
- выводы по работе.

Теоретические сведения

Графическая библиотека PHP GD

Дополнительные возможности PHP приобретает за счет использования расширений, позволяющих решать стоящие перед web-разработчиком задачи. Одним из таких расширений является графическая библиотека GD, предназначенная для динамической работы с растровыми изображениями.

Библиотека поддерживает практически все существующие форматы графики для использования в WWW: PNG, JPEG, GIF, ICO и различные методы работы с графическими файлами (применение фильтров, текст, изменение размера и прочее), позволяет создавать изображения, состоящие из линий, дуг, текста (включая программный выбор шрифтов) и других изображений, а также использовать различные цвета.

Лабораторная работа №8. Технология AJAX
(4 часа)

Цель работы:

Получить практические навыки в использовании технологии AJAX.

Задание (типовое):

Во всех вариантах заданий необходимо разработать скрипт, использующий возможности технологии AJAX.

Создать документ с двумя раскрывающимися списками. Перечень вариантов в первом раскрывающемся списке разметить с помощью языка разметки гипертекста HTML. Выбор варианта в первом раскрывающемся списке определяет тот или иной набор вариантов для выбора во втором раскрывающемся списке. Варианты для выбора во втором раскрывающемся списке должны быть сформированы с использованием технологии AJAX. Итоговые результаты выбора в обоих раскрывающихся списках обрабатываются серверным скриптом и отображаются на клиентской стороне.

При выполнении лабораторной работы возможно применение функций библиотеки jQuery для использования возможностей технологии AJAX.

Порядок выполнения лабораторной работы:

1. Разметить html-документ.
2. Написать скрипт с использованием технологии AJAX.
3. Протестировать созданный скрипт.

Содержание отчета (отчет в электронном виде):

- отчет сохранить в файле с именем АВТ-000 Иванов (лр8).doc;
- титульный лист (образец можно скачать по адресу http://ermak.cs.nstu.ru/webprog/wp_labwork_title_page.doc);
- цель работы;
- задание;
- порядок выполнения лабораторной работы
- исходный код скрипта;
- выводы по работе.

Теоретические сведения

AJAX (Asynchronous JavaScript and XML) — новый подход к построению интерактивных пользовательских интерфейсов web-приложений, заключающийся в «фоновом» обмене данными браузера с web-сервером

В результате при обновлении данных документ не перезагружается полностью и web-приложения могут быть сделаны более быстрыми и удобными.

AJAX не самостоятельная технология, а концепция использования нескольких смежных технологий: технологии динамического обращения к серверу «на лету», без перезагрузки всей страницы полностью и DHTML для динамического изменения содержимого документа.

AJAX — аббревиатура, обозначающая подход к созданию web-приложений с помощью следующих технологий: стандартизированное представление силами HTML и CSS, динамическое отображение и взаимодействие с пользователем с помощью DOM, обмен и обработка данных в виде XML, асинхронные запросы с помощью объекта XMLHttpRequest, использование JavaScript.

Впервые термин AJAX появился в феврале 2005 года, когда пришлось как-то назвать новый набор технологий, предлагаемый клиенту.

Плюсы технологии: экономия трафика (использование AJAX позволяет значительно сократить трафик при работе с web-приложением, благодаря тому, что часто вместо загрузки всей страницы достаточно загрузить только небольшую изменившуюся часть), уменьшение нагрузки на сервер (AJAX позволяет несколько снизить нагрузку на сервер, к примеру, в Gmail, когда помечаются прочитанные письма, серверу достаточно внести изменения в базу данных и отправить клиентскому скрипту сообщение об успешном выполнении операции, вместо необходимости повторно создавать страницу и отсылать ее клиенту), увеличение реакции интерфейса (поскольку нужно загрузить только

изменившуюся часть, то пользователь видит результат своих действий быстрее).

Минусы технологии: интеграция со стандартными инструментами браузера (динамически создаваемые страницы не регистрируются браузером в истории посещения страниц, поэтому не работает кнопка «Назад», предоставляющая пользователям возможность вернуться к просмотренным ранее страницам), другой недостаток изменения контента страницы при постоянном URL заключается в невозможности сохранения закладки на желаемый материал. Частично решить эти проблемы можно с помощью динамического изменения идентификатора фрагмента (части URL после #), что позволяют многие браузеры, динамически загружаемое содержание недоступно поисковикам (поисковые машины не могут выполнять JavaScript, поэтому разработчики должны позаботиться об альтернативных способах доступа к содержимому сайта), старые методы учета статистики сайтов становятся неактуальными (многие сервисы статистики ведут учет просмотров новых страниц сайта, для сайтов страницы которых широко используют AJAX, такая статистика теряет актуальность).

В стандартном web-приложении обработкой всей информации занимается сервер, тогда как браузер отвечает только за взаимодействие с пользователем, передачу запросов и вывод поступившего HTML

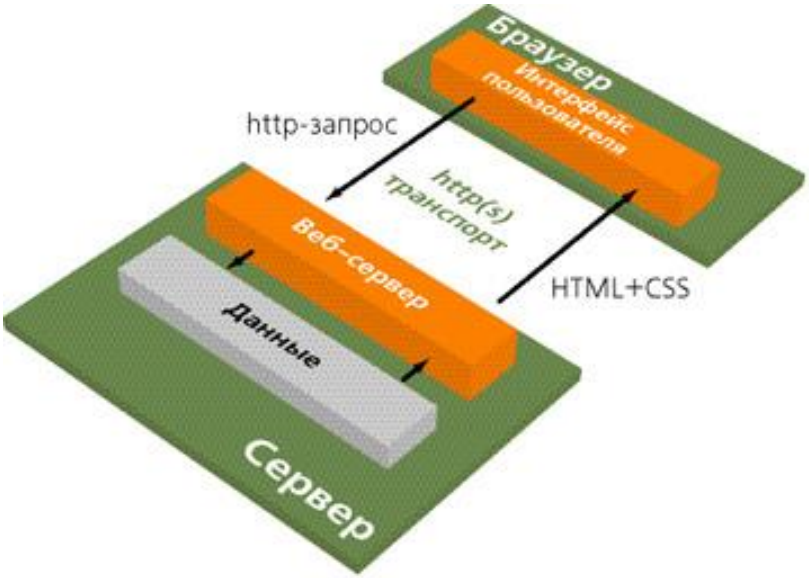
В AJAX-приложении между пользователем и сервером появляется еще один посредник — движок AJAX. Он определяет, какие запросы можно обработать «на месте», а за какими необходимо обращаться на сервер.

Если раньше на каждый запрос сервер выдавал новую страницу, то теперь он отсылает лишь те данные, которые нужны клиенту, а HTML-разметку из них прямо в браузере формирует движок AJAX.

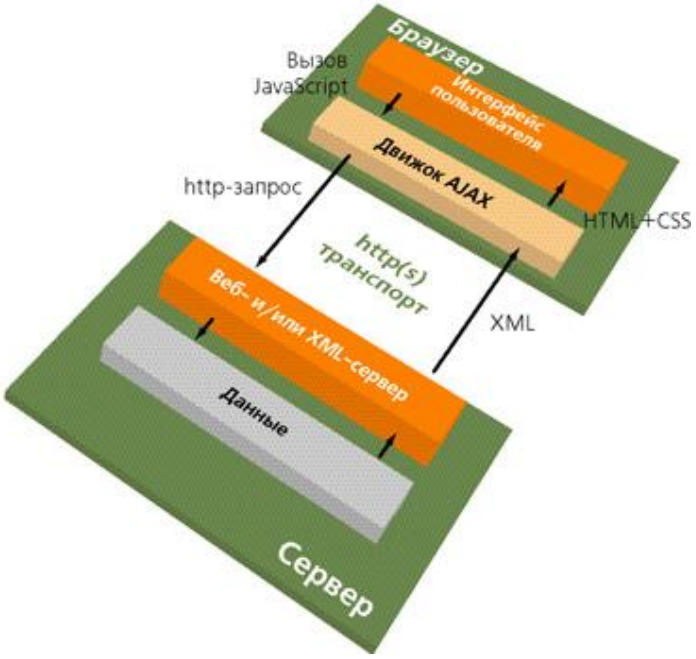
Асинхронность проявляется в том, что далеко не каждый щелчок пользователя доходит до сервера, причем обратное тоже справедливо — далеко не каждая реакция сервера обусловлена запросом пользователя.

Большую часть запросов формирует движок AJAX, причем его можно написать так, что он будет загружать информацию превентивно, предугадывая действия пользователя.

Качественная нагрузка на сервер меняется — если раньше запросов было мало, но каждый из них требовал значительных ресурсов (серверу нужно получить информацию из БД, сформировать из нее web-страницу и отдать браузеру), то теперь задача сервера упрощается (формировать web-страницы не нужно, объем передаваемых данных меньше), но запросов обрабатывать приходится больше.



Классическое web-приложение



Web-приложение, использующее технологию AJAX