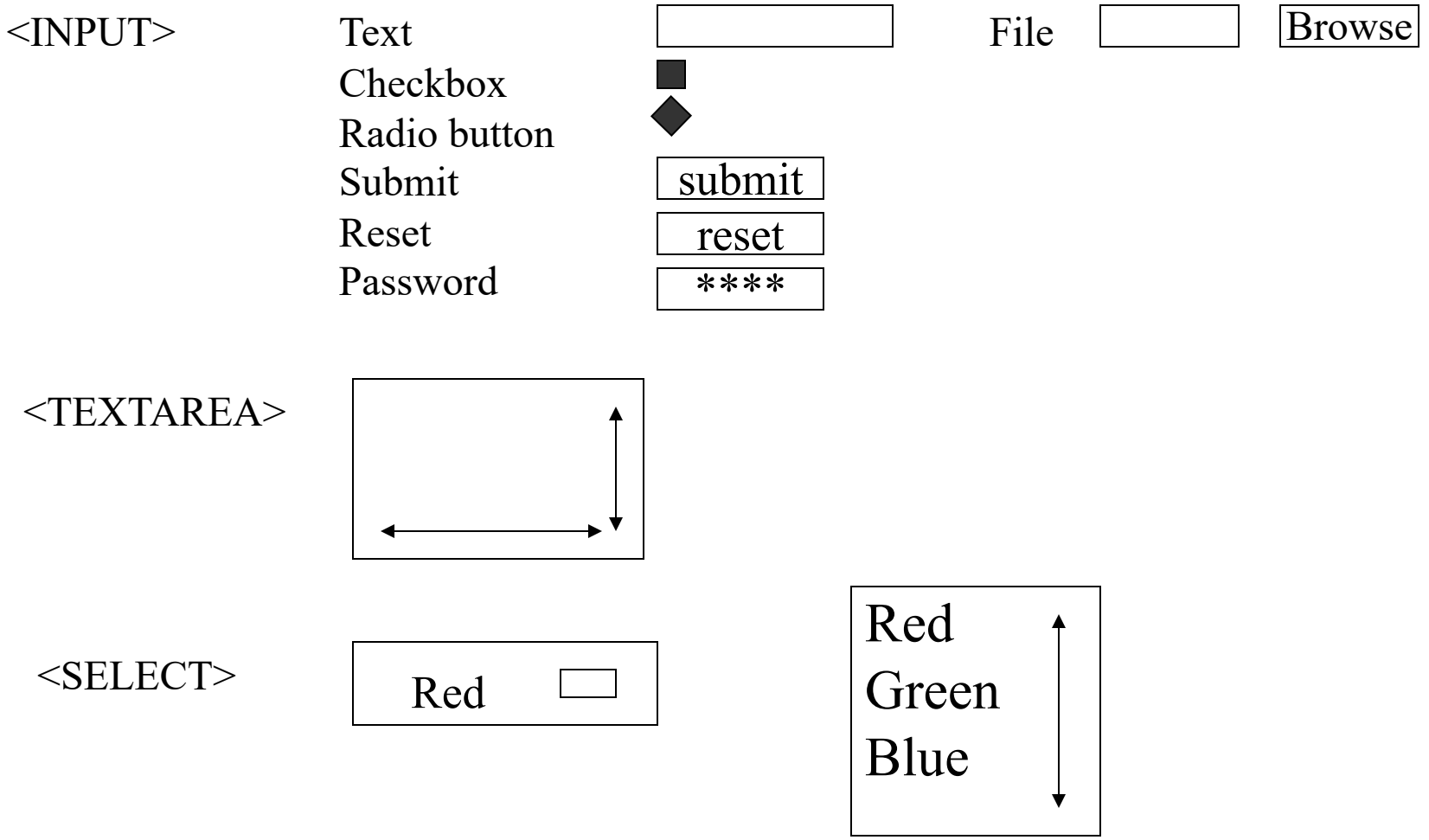
Lecture

Forms and Common Gateway Interface Mechanism

Forms

- Used to create a set of pages that contain fields in which the viewer can select and supply information
  - Introduced very early in HTML 2.0
  - Allows WWW users to perform data entry
  - Permit direct interaction with customers for inquiries, registration, sales of products, and services
  - To create a capability requires two steps:
    - Use HTML form elements to create the pages that contain the form
    - Write a server-side script to process form data; this program must be placed so the WWW server can execute it

The Original Set of User Interface Elements



FORM Element and Some Attributes

- **Syntax <FORM>...</FORM>**
- **Attribute Specifications**
  - ACTION=*URI* (form handler)
  - METHOD=[*get* | post] (HTTP method for submitting form)
    - GET is the default; form contents are appended to the URL
    - POST form contents to be sent as payload
  - ENCTYPE=*ContentType* (content type to submit form as)
    - **Defaults** to application/x-www-urlencoded which returns name/value pairs, separated by &, spaces replaced by + and reserved characters (like #) replaced by %HH, H a hex digit
  - ACCEPT-CHARSET=*Charsets* (supported character encodings)
  - TARGET=*FrameTarget* (frame to render form result in, in HTML4)

    (a browsing context name or keyword, in HTML5, such as _self, _blank, _parent, _top, iframename)
  - ONSUBMIT=*Script* (form was submitted)
  - ONRESET=*Script* (form was reset)
  - AUTOCOMPLETE (**HTML5 ONLY**) values completed by browser

https://developer.mozilla.org/en-US/docs/Web/HTML/Element/form

<INPUT> Tag

- Used inside <FORM> tag to specify a data-entry object
- Has 19 attributes, here are a few
  - TYPE: User input type (default is TEXT)
  - NAME: Name of data entry object whose value the user will supply
  - VALUE: Required for radio and checkboxes
  - CHECKED: For radio buttons and checkboxes
  - SIZE: Specific to each type of field
  - MAXLENGTH: Limit on accepted characters
  - SRC: Image file used as a graphical submit button when TYPE=IMAGE
  - DISABLED unavailable in this context
  - READONLY for text and passwords
- **HTML5 adds several new attributes for validation**
- See http://www.w3schools.com/tags/tag_input.asp

<INPUT> Element, Type Options(cont'd)

- TYPE:[CHECKBOX|FILE|HIDDEN|IMAGE|PASSWORD|
  RADIO|RESET|SUBMIT|TEXT]

[HTML5 adds 13 new input types. See later slides]

- **CHECKBOX:** A single value, on/off; each generates name/value pair

  <INPUT TYPE=CHECKBOX CHECKED NAME="MARRIED" VALUE="yes">

- **FILE:** Users attach a file to the form contents; a text field holds the file name and a button permits browsing

  <INPUT TYPE=FILE NAME="fname">

- **HIDDEN:** The field is not rendered, so servers can maintain state information

  <INPUT TYPE=HIDDEN NAME="BANKACCT" VALUE="A057-23-789">

<INPUT> Element, Type Options(cont'd)

- **RESET:** Defines a button that users click to reset fields to their initial state

```
<INPUT TYPE=RESET VALUE="CLEAR">
```

- **SUBMIT:** Defines a button that users click to submit the form's contents to the server

```
<INPUT TYPE=SUBMIT VALUE="submit data">
```

- **TEXT:** An input field of a single line where users can enter data

```
<INPUT TYPE=TEXT SIZE=20 NAME="lastname"
VALUE="type your last name">
```

<INPUT> Element, Type Options(cont'd)

- **IMAGE:** Used for graphical submit buttons

```
<INPUT TYPE=IMAGE SRC="banner.gif“ VALUE="gohome">
```

- **PASSWORD:** Just like TYPE=TEXT, but the input is echoed with *

```
<INPUT TYPE=PASSWORD SIZE=10 NAME="pw">
```

- **RADIO:** Used for attributes that take a single value from a set of alternatives; all buttons have same name and explicit value

```
<INPUT TYPE=RADIO NAME="AGE" VALUE="0-20">
<INPUT TYPE=RADIO NAME="AGE" VALUE="21-50">
<INPUT TYPE=RADIO NAME="AGE" VALUE="51-100"
  CHECKED>
```

<INPUT> Element, Type Options

- TYPE:[COLOR|DATE|DATETIME|DATETIME-LOCAL|EMAIL|MONTH|NUMBER|RANGE|SEARCH|TEL|TIME|URL|WEEK]
  - **COLOR:** Used for input fields that should contain a color

    Select color: <INPUT TYPE="COLOR" name=”favcolor">
  - **DATE:** Allows the user to select a date

    Birthday: <INPUT TYPE="DATE" NAME="bday">
  - **DATETIME:** Allows the user to select a date and time (with time zone)

    Birthday: <INPUT TYPE="DATETIME" NAME="BDAYTIME”
  - **EMAIL:** Allows the user to enter an e-mail address

    E-Mail: <INPUT TYPE="EMAIL" NAME="email”
  - **MONTH:** Allows the user to select month/year

    Birthday (M/Y): <INPUT TYPE="MONTH" NAME="bdaymonth"

<INPUT> Element, Type Options (cont'd)

- **NUMBER:** Used to enter a numeric value

```
Quantity (1-5): <INPUT TYPE="NUMBER"
name="quantity" min="1" max="5">
```

- **RANGE:** Used to enter a value from a range of numbers

```
<INPUT TYPE="RANGE" NAME="points">
```

- **SEARCH:** Used for search fields (behaves like regular TEXT)

```
Search Google: <INPUT TYPE="SEARCH"
NAME="GOOGLESEARCH”>
```

- **TEL:** Allows the user to enter a telephone num.

```
Telephone: <INPUT TYPE="TEL" NAME="ustel"
```

- **TIME:** Allows the user to select a time

```
Time: <INPUT TYPE="TIME" NAME="ustime"
```

<INPUT> Element, Type Options (cont'd)

- **URL:** Used to enter a URL address

```
Add Homepage: <INPUT TYPE="URL" name="homepage"
```

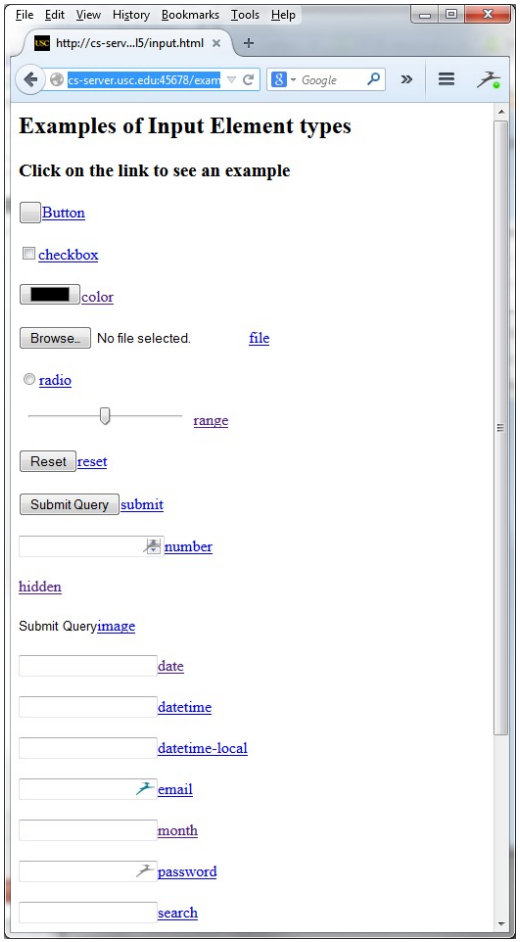
- **WEEK:** Used to select a week and year

```
Select a week: <INPUT TYPE="WEEK" NAME="week_YR">
```

- **DATETIME-LOCAL:** Used selct date and time(no time zone)

```
Birthday: <INPUT TYPE="DATETIME-LOCAL"
NAME="bday">
```

<INPUT> Element, Type Options (cont'd)



Examples of all of the <input> element types, including the most recent in HTML5
provided by w3schools

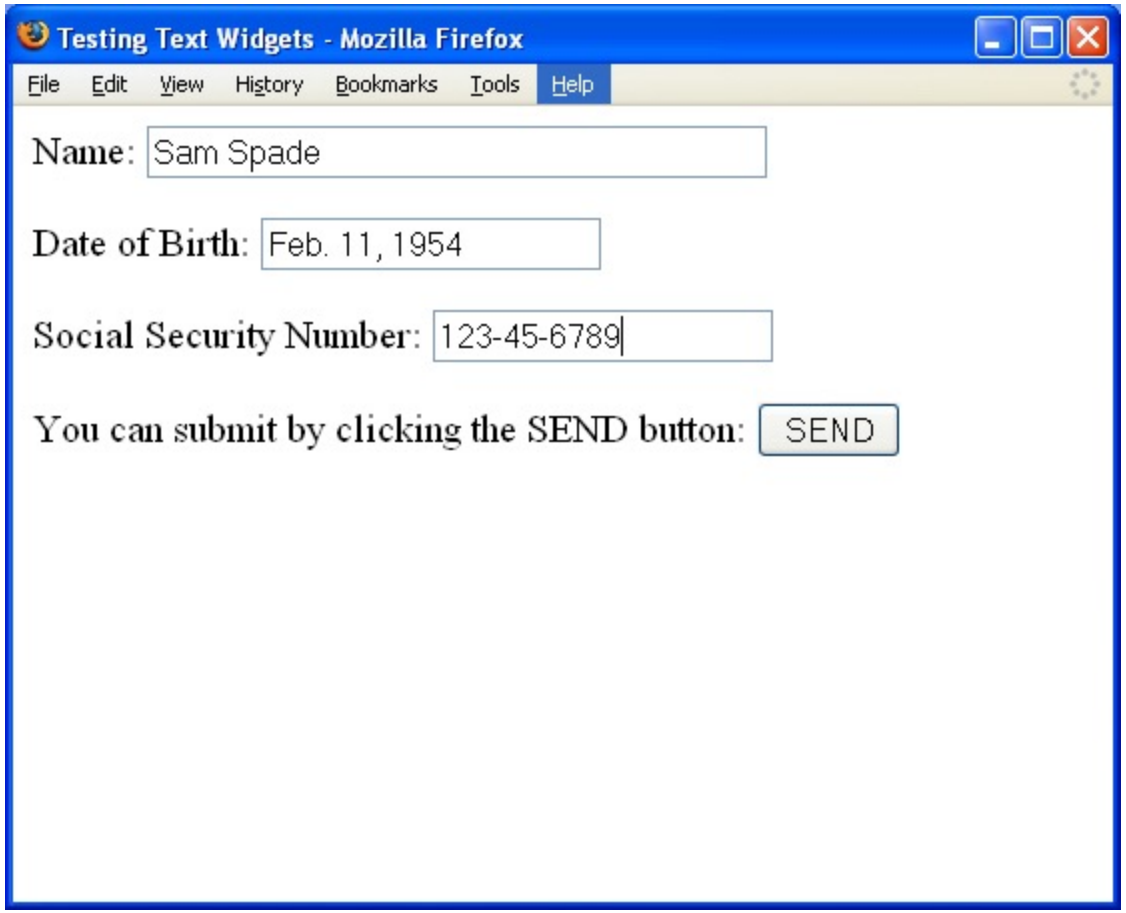
http://csci571.com/examples/html5/input.html

Example of <FORM> With Text Widgets

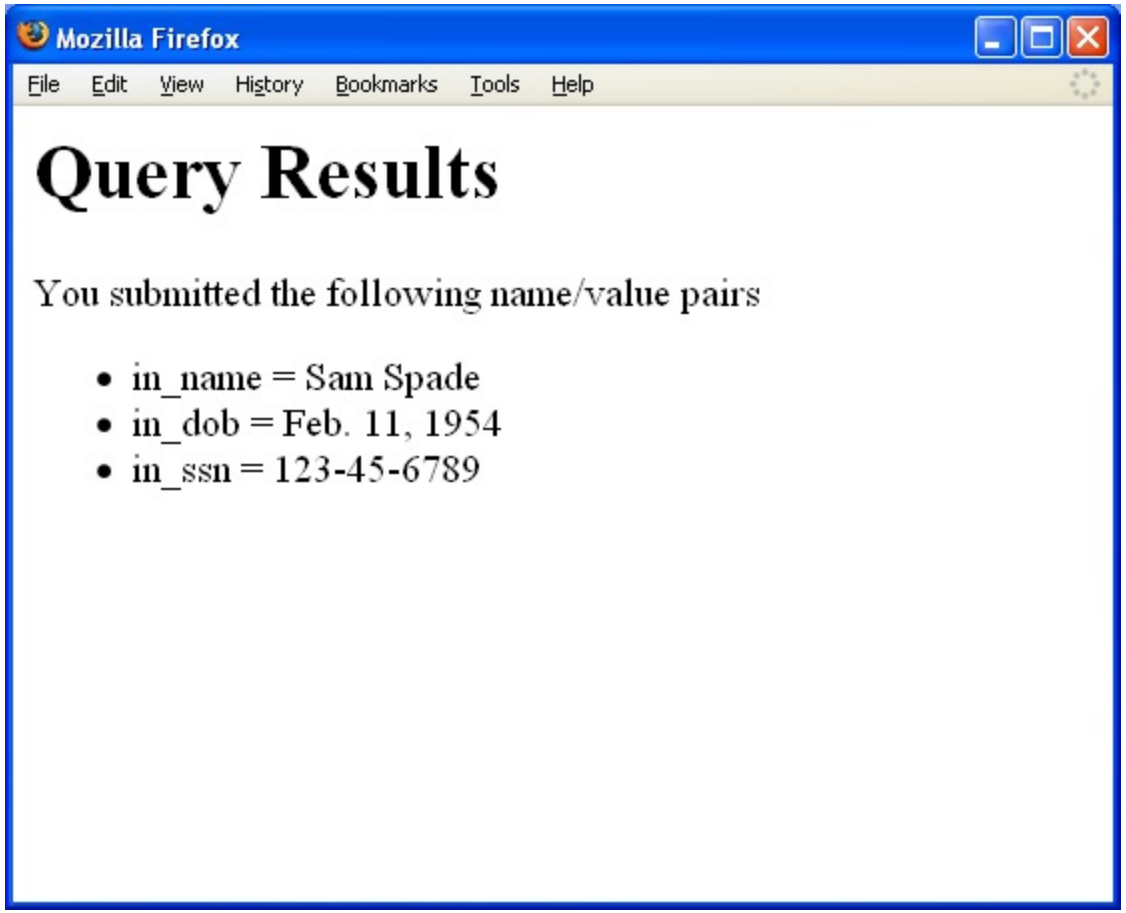
```
<HTML><HEAD><TITLE>Testing Text Widgets</TITLE>
  </HEAD>
<BODY>
<FORM METHOD="POST"
ACTION="/cgi-bin/post-query">
Name: <INPUT NAME="in_name" TYPE="text" SIZE=40><P>
Date of Birth: <INPUT TYPE="text" NAME="in_dob"><P>
Social Security Number: <INPUT TYPE="text"
  NAME="in_ssn"><P>
You can submit by clicking the SEND button:
  <INPUT TYPE="submit" VALUE="SEND">
</FORM></BODY></HTML>
```

*Note:**post-query** is a standard Apache CGI program distributed by web servers and used to check that form elements are being properly sent to the server*

Browser Output of Text Widgets Example



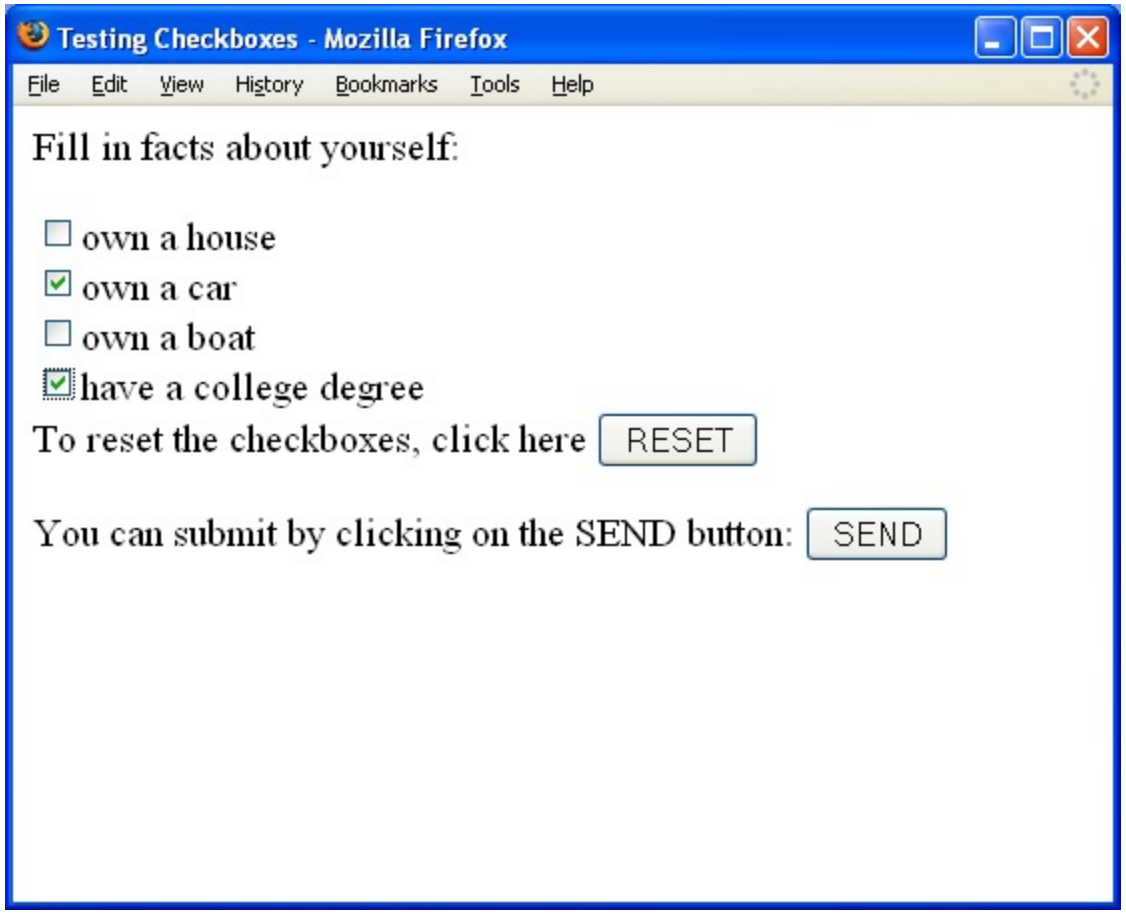
Query Results for Text Widget Example



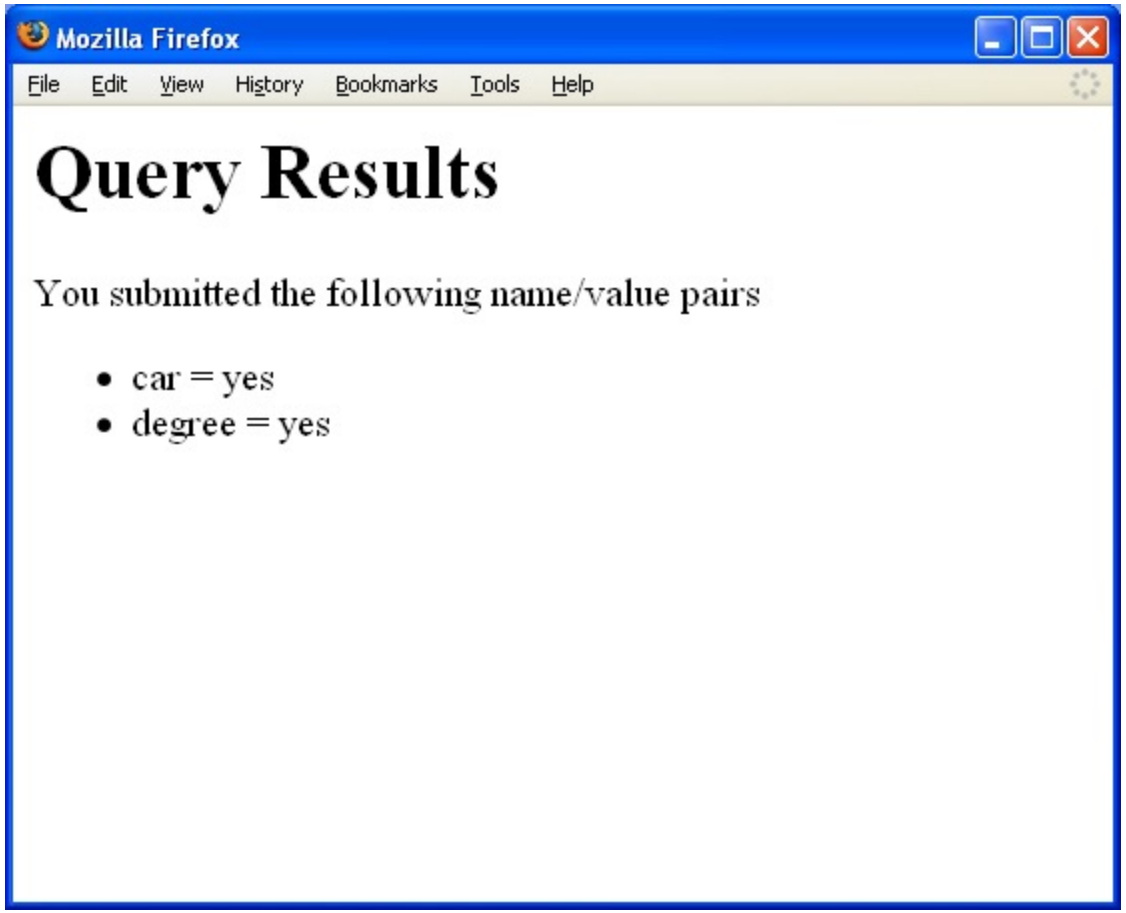
Example of <FORM> With Checkboxes

```
<HTML><HEAD><TITLE>Testing Checkboxes</TITLE></HEAD>
<BODY>
<FORM METHOD="POST" ACTION="/cgi-bin/post-query">
Fill in facts about yourself:<P>
<INPUT TYPE="checkbox" NAME="house" VALUE="yes">own a
house<BR>
<INPUT TYPE="checkbox" NAME="car" VALUE="yes">own a
car<BR>
<INPUT TYPE="checkbox" NAME="boat" VALUE="yes">own a
boat<BR>
<INPUT TYPE="checkbox" NAME="degree" VALUE="yes">have a
college degree<BR>
To reset the checkboxes, click here
<INPUT TYPE=reset VALUE="RESET"><P>
You can submit by clicking on the SEND button:
<INPUT TYPE=submit VALUE="SEND"><P>
</FORM></BODY></HTML>
```

Browser Output of Checkbox Example



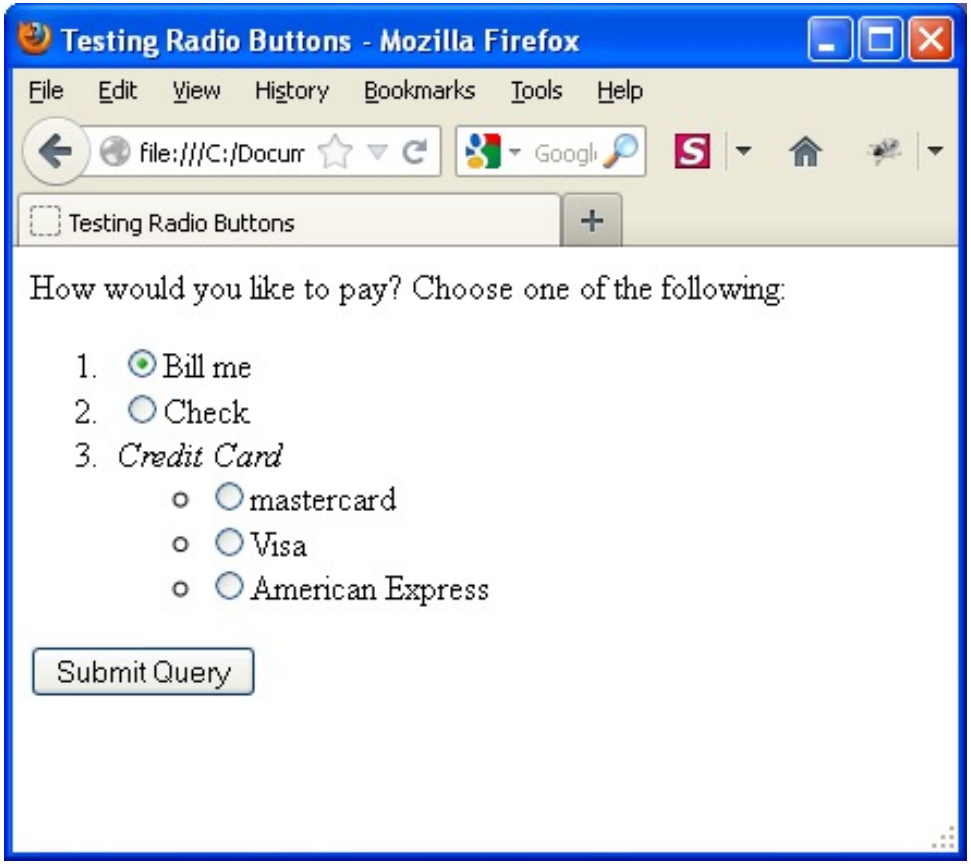
Query Results of Checkbox Example



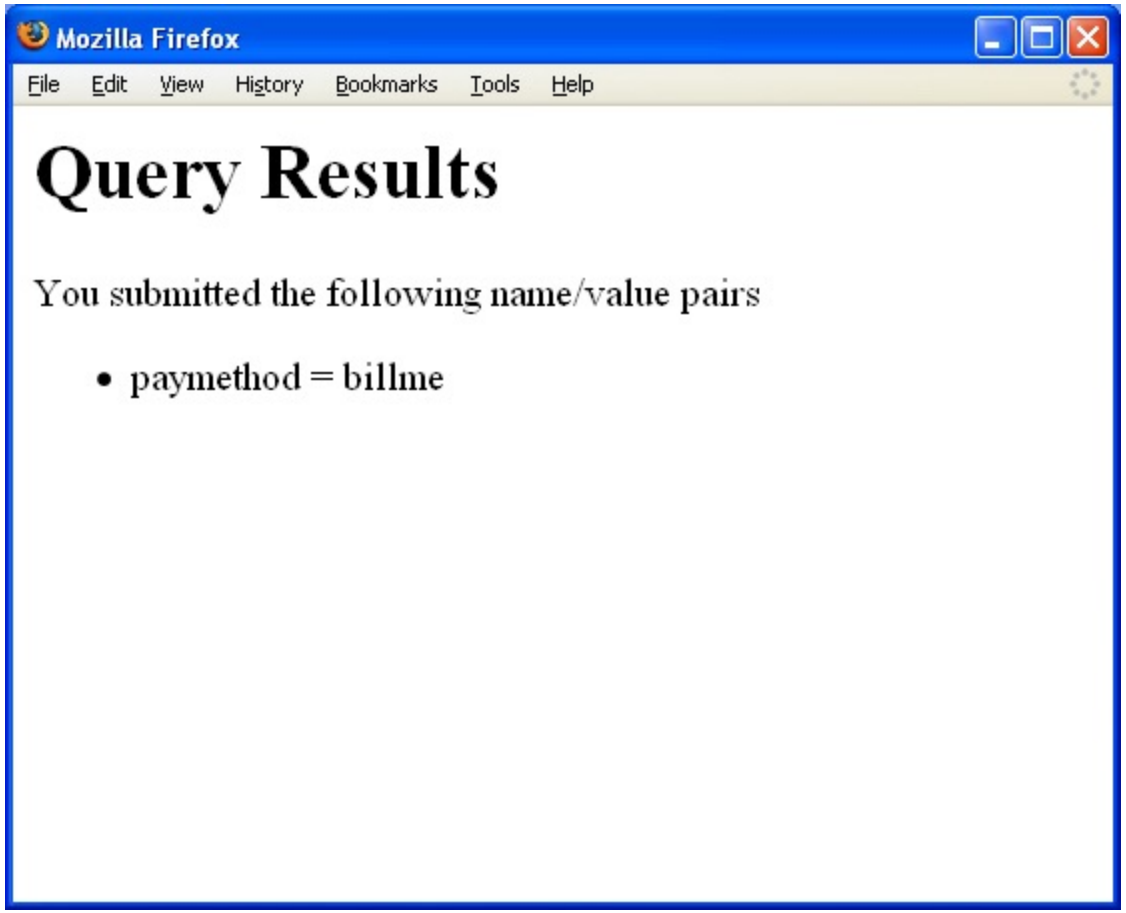
Example of <FORM> With Radio Buttons

```
<HTML><HEAD><TITLE>Testing Radio Buttons</TITLE></HEAD>
<BODY><FORM METHOD="POST" ACTION="/cgi-bin/post-query">
How would you like to pay? Choose one of the following:<P>
<OL><LI><INPUT TYPE="radio" Name="paymethod" VALUE="billme"
CHECKED>Bill me<BR>
<LI><INPUT TYPE="radio" Name="paymethod"
VALUE="check">Check<BR>
<LI><I> Credit Card </I>
<UL><LI><INPUT TYPE="radio" Name="paymethod"
VALUE="mastercard">mastercard<BR>
<LI><INPUT TYPE="radio" Name="paymethod"
VALUE="visa">Visa<BR>
<LI><INPUT TYPE="radio" Name="paymethod"
VALUE="amer">American Express<BR>
</UL></OL><INPUT TYPE="submit" VALUE="Submit Query">
</FORM></BODY></HTML>
```

Browser Output of Radio Buttons



Query Results for Radio Buttons Example



<TEXTAREA> Tag

- specifies a large rectangular text-entry object with multi-line input and scroll bars
- Attributes:

  NAME=name specifies a name for the data entry object to be sent to the server-side script

  COLS=num

  - Width (in characters) of a text-entry region on the screen
  - If user types more than COLS characters, field is scrolled

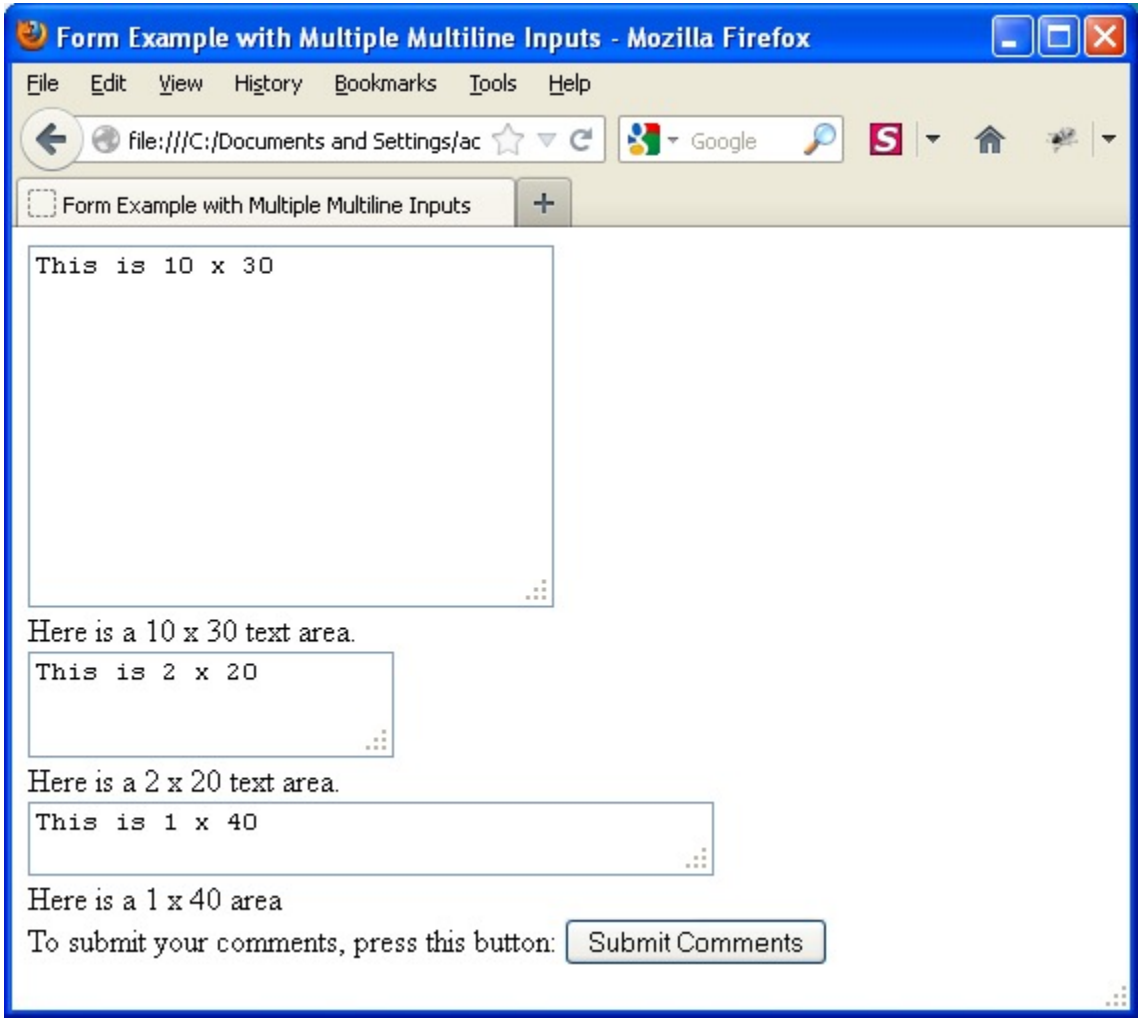
  ROWS=num

  - Height (in characters) of a text-entry region on the screen
  - If user types more than ROWS lines, field is scrolled

Example of Multiline Input Areas

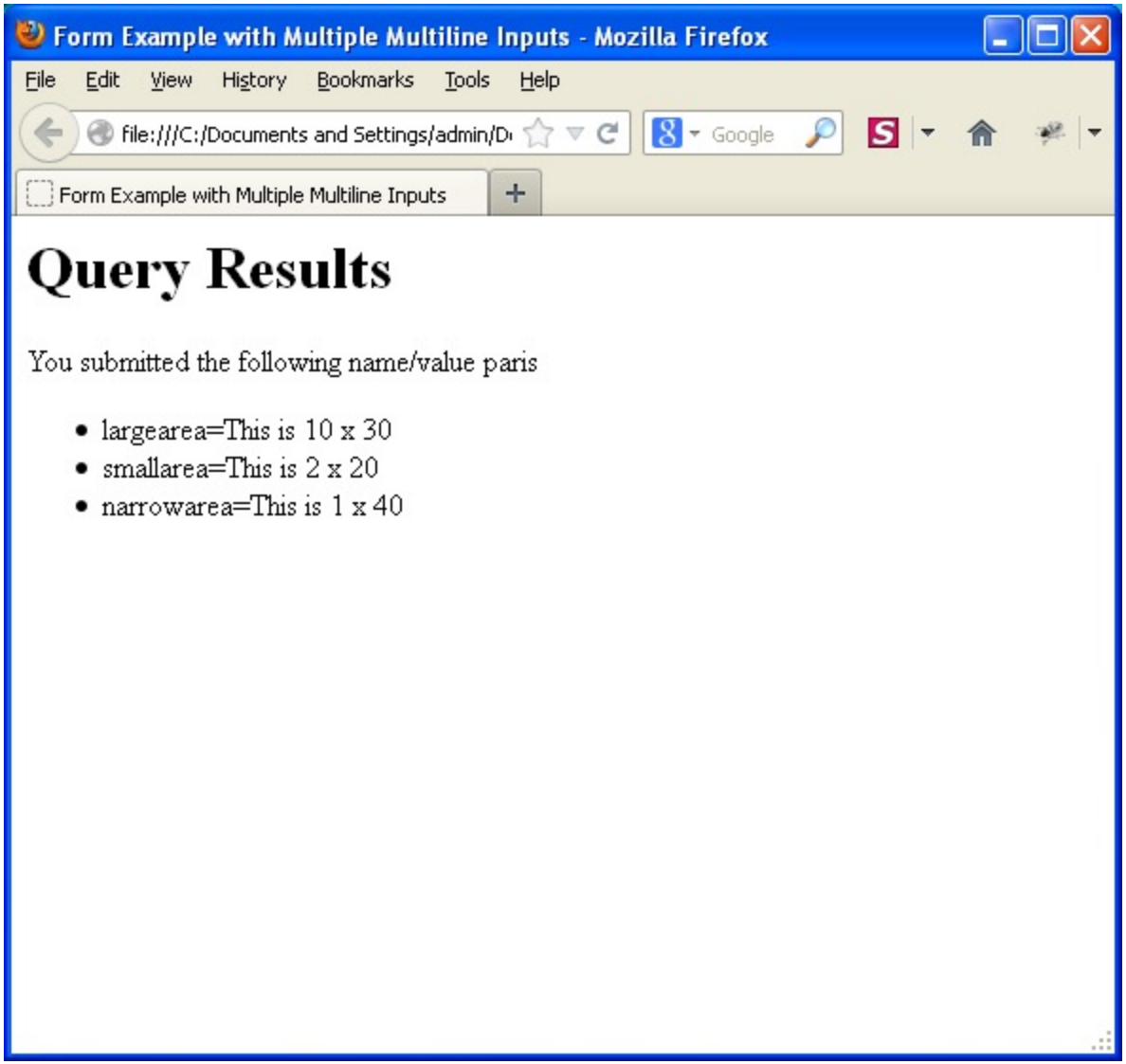
```
<HTML><HEAD><TITLE>Form Example with Multiple
Multiline Inputs</TITLE></HEAD> <body>
<form method="POST" action="/cgi-bin/postquery">
<TEXTAREA NAME="largearea" ROWS=10 COLS=30>This is
10 x 30</TEXTAREA><BR>
Here is a 10 x 30 text area.<BR>
<TEXTAREA NAME="smallarea" ROWS=2 COLS=20>This is 2
x 20</TEXTAREA><BR>
Here is a 2 x 20 text area.<BR>
<TEXTAREA NAME="narrowarea" ROWS=1 COLS=40>This is 1
x 40</TEXTAREA><BR>
Here is a 1 x 40 area <BR>
To submit your comments, press this button:
<INPUT TYPE="submit" VALUE="Submit Comments"><BR>
</FORM></BODY></HTML>
```

Browser Output of Multiline Input Areas



Initial Screen

Query Results of Textarea Example



<SELECT> Tag

- Used inside the <FORM> element to specify a selection list object (a list of items or a pop-down menu that the user can select from)
- Attributes:
  - NAME=name
    - Specifies a name for the data entry object to be passed to the server-side script
  - SIZE=num
    - Number of lines of the list to display at a time
    - If SIZE is 1 or unspecified, browser will display as a drop-down list box
    - If SIZE is greater than 1, browser will display as a scrollable list with only SIZE options visible at a time

<SELECT> Tag Attributes

- MULTIPLE
  - Specifies that multiple list items may be selected (whereas normally only 1 item can be selected)
  - All selected values are sent to server-side script as separate name/value pairs
- HTML5 adds more attributes:
  - **AUTOFOCUS:** drop-down list should automatically get focus
  - **FORM:** defines one of more forms the select fields belongs to
  - **REQUIRED:** user is required to select a value before submitting the form

<OPTION> Tag

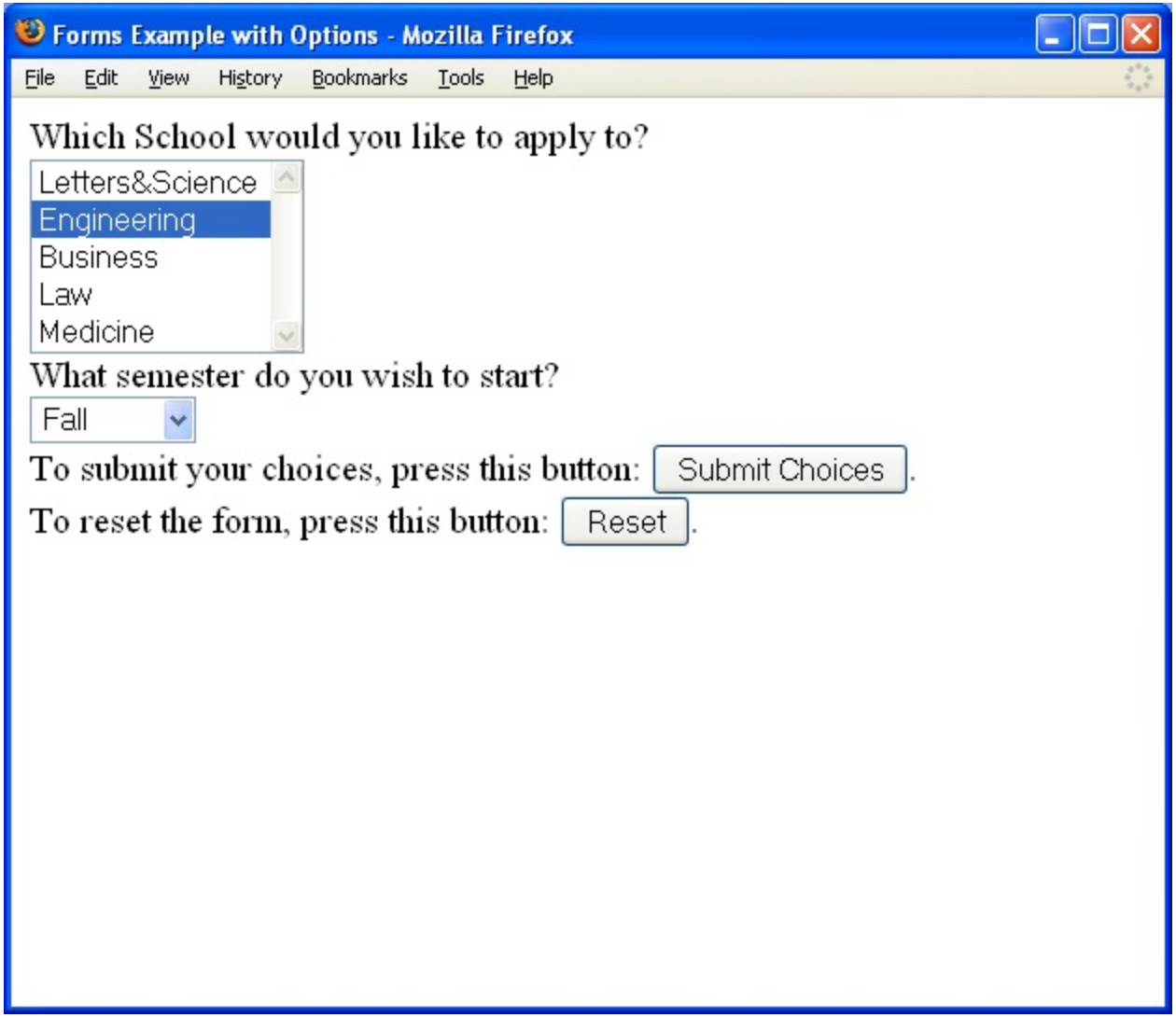
- Used inside the <SELECT> tag to specify the start of a new menu item in the selection list
- Syntax as follows:

  <OPTION attributes> Text
- Attributes:
  - SELECTED
    - Menu item is pre-selected in the list
  - VALUE="text"
    - Text specifies the value to be sent to the script if the option is selected
    - By default, the text following the OPTION element is sent
  - DISABLED
    - Specifies a "grayed", non-selectable item
  - **HTML5 adds the REQUIRED attribute**

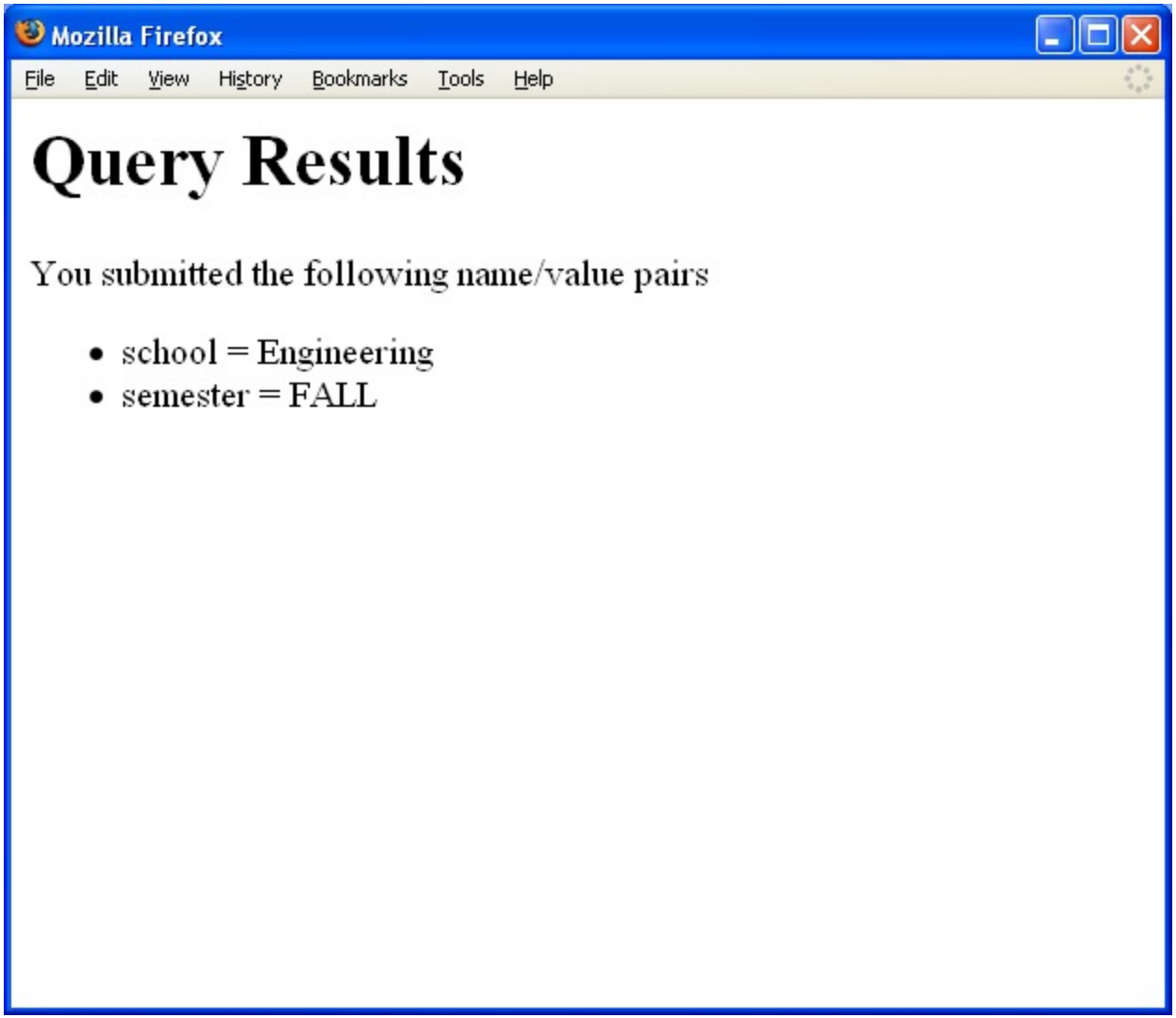
Example of <SELECT>, <OPTION> Tags

```
<HTML><HEAD><TITLE>Forms Example with
Options</TITLE></HEAD><BODY>
<FORM METHOD="POST“ ACTION="/cgi-bin/post-query">
Which School would you like to apply to? <BR>
<SELECT NAME="school" SIZE=5>
  <OPTION> Letters&Science</OPTION>
  <OPTION SELECTED> Engineering</OPTION>
  <OPTION> Business</OPTION>
<OPTION>Law</OPTION><OPTION> Medicine</OPTION></SELECT><BR>
What semester do you wish to start? <BR>
<SELECT NAME="semester">
  <OPTION SELECTED> Fall</OPTION>
  <OPTION> Spring</OPTION>
  <OPTION>Summer</OPTION></SELECT><BR>
To submit your choices, press this button: <INPUT
TYPE="submit" VALUE="Submit Choices">. <BR >
To reset the form, press this button: <INPUT TYPE="reset"
VALUE="Reset">.
</FORM></BODY></HTML>
```

Browser Output of <SELECT>, <OPTION> Example



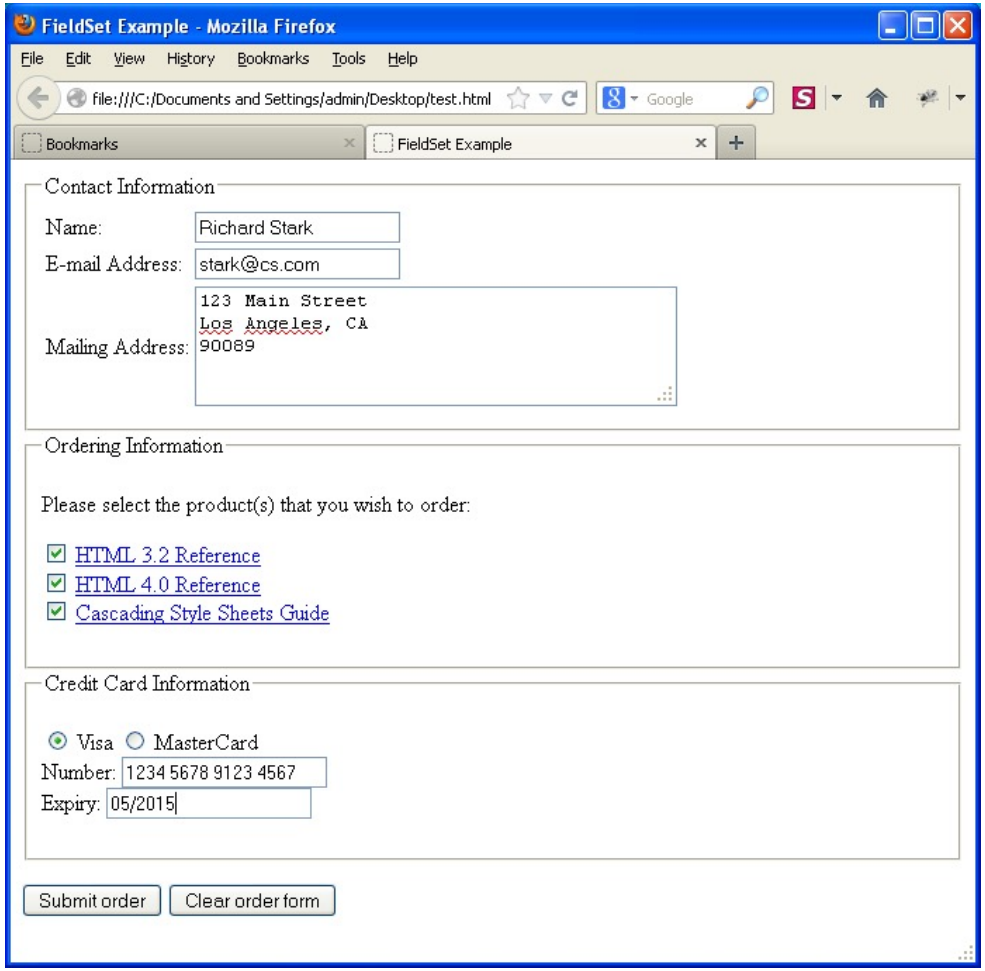
Query Results for <SELECT> Example



FIELDSET – Form Control Group

- The **FIELDSET** element defines a *form control group*.
  - By grouping related form controls, authors can divide a form into smaller, more manageable parts, improving the usability problem that can strike when confronting users with too many form controls.
  - The grouping provided by **FIELDSET** also helps the accessibility of forms to those using aural browsers by allowing these users to more easily orient themselves when filling in a large form.
- The content of a **FIELDSET** element must begin with a **LEGEND** to provide a caption for the group of controls. Following the **LEGEND**, **FIELDSET** may contain any HTML element, including another **FIELDSET**.

Browser Output
3 Fieldsets Grouping form elements



Run through Tab order:
Name
E-mail
Mailing Address
HTML 3.2
HTML 4.0
.
.
Etc

To test ACCESSKEY in Chrome use
ALT + ACCESSKEY (I, O, C)
To test ACCESSKEY in Firefox use
ALT + SHIFT + ACCESSKEY

http://csci571.com/examples/html5/fieldsettest.html

Fieldset Example (see next slide)

```
<FORM METHOD=post ACTION="/cgi-bin/order.cgi">

<FIELDSET> <LEGEND ACCESSKEY=I>Contact Information</LEGEND>
<TABLE> <TR> <TD> <LABEL FOR=name ACCESSKEY=N>Name:</LABEL> </TD>
            <TD> <INPUT TYPE=text NAME=name ID=name> </TD> </TR>
<TR> <TD> <LABEL FOR=email ACCESSKEY=E>E-mail Address:</LABEL> </TD>
<TD> <INPUT TYPE=text NAME=email ID=email> </TD> </TR>
<TR> <TD> <LABEL FOR=addr ACCESSKEY=A>Mailing Address:</LABEL> </TD>
<TD> <TEXTAREA NAME=address ID=addr ROWS=4 COLS=40></TEXTAREA> </TD> </TR> </TABLE> </FIELDSET>

<FIELDSET> <LEGEND ACCESSKEY=O>Ordering Information</LEGEND>
<P>Please select the product(s) that you wish to order:</P>
<P> <LABEL ACCESSKEY=3>
<INPUT TYPE=checkbox NAME=products VALUE="HTML 3.2 Reference">
<A HREF="/reference/wilbur/">HTML 3.2 Reference</A> </LABEL> <BR> <LABEL ACCESSKEY=4> <INPUT TYPE=checkbox
NAME=products VALUE="HTML 4.0 Reference">
<A HREF="/reference/html40/">HTML 4.0 Reference</A> </LABEL> <BR> <LABEL ACCESSKEY=S> <INPUT TYPE=checkbox
NAME=products VALUE="CSS Guide"> <A HREF="/reference/css/">Cascading Style Sheets Guide</A> </LABEL> </P>
</FIELDSET>

<FIELDSET> <LEGEND ACCESSKEY=C>Credit Card Information</LEGEND> <P> <LABEL ACCESSKEY=V> <INPUT TYPE=radio
NAME=card VALUE=visa> Visa </LABEL> <LABEL ACCESSKEY=M>
<INPUT TYPE=radio NAME=card VALUE=mc> MasterCard </LABEL> <BR>
<LABEL ACCESSKEY=u> Number: <INPUT TYPE=text NAME=number> </LABEL> <BR>
<LABEL ACCESSKEY=E> Expiry: <INPUT TYPE=text NAME=expiry> </LABEL> </P> </FIELDSET>

<P> <INPUT TYPE=submit VALUE="Submit order"> <INPUT TYPE=reset VALUE="Clear order form">
</P> </FORM>
```

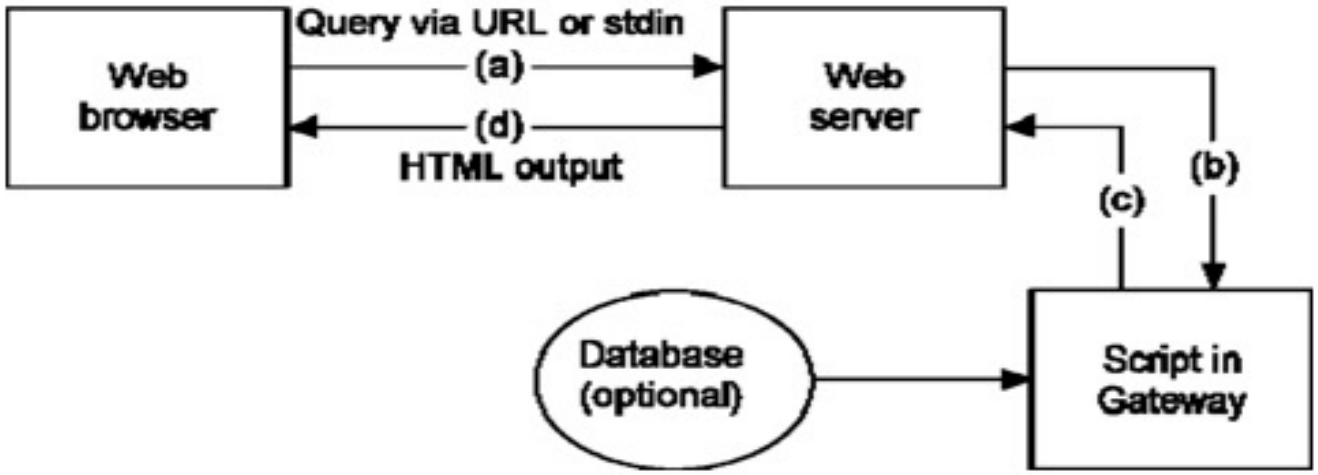
3 fieldsets with legends
ACCESSKEY specifies a single character for giving focus

Purpose of the CGI

- Common Gateway Interface (CGI) is a mechanism by which programs, called *scripts*, can be used to create dynamic Web documents
  - Scripts are placed in a server directory often named cgi-bin
  - Scripts can deliver information that is not directly readable by clients
  - Scripts dynamically convert data from a non-Web source (e.g., DBMS) into a Web-compatible document
- Current version of CGI is 1.1
- The reason for the term "common gateway" is these programs act as gateways between the WWW and any other type of data or service
- See http://www.w3.org/CGI/

Basic Operation

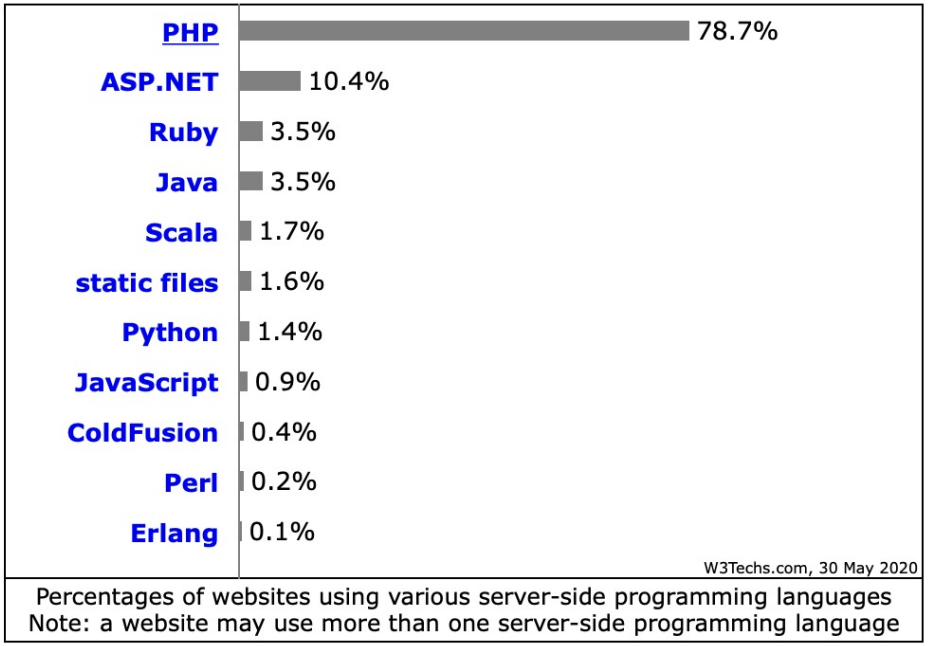
- An executable program that can be run without being directly invoked by users



The browser issues a query, (a), which is sent to the server; the server interprets it and invokes the proper CGI script, passing it the input data, (b); output from the script is returned, (c), via the server, to the browser, (d); output may be HTML, but it may instead be a URL, which is fetched by the server

Languages to Write Gateway Programs

- Any language that can produce an executable file
- Some typical ones are:
  - Traditional compiled languages such as C/C++
  - Or interpreted languages such as:
    - PHP, JavaScript, Python or Java
- Interpreted languages are often preferred as they are
  - Easy to write and portable, and speed is usually not a factor



http://w3techs.com/technologies/overview/programming_language/all

Anchors Are Used to Invoke CGI Scripts

- A hypertext reference can refer to:
  - A remote file

```
<A HREF="http://domain_name/path/myfile.html">
```

  - An executable script in the cgi-bin directory

```
<A HREF="http://domain_name/cgi-bin/scriptname">
```

  - An executable script with arguments

```
<A HREF="http://domain_name/cgi-bin/scriptname?arg1+arg2">
```

- URLs produced by the query "bicycle tours":

```
http://search.yahoo.com/bin/search?p=bicycle+tours
http://search.msn.com/results.asp?RS=CHECKED&FORM=M
SNH&v=1&q=bicycle+tours&zip=90211
```

the current versions are somewhat different

CGI Script Environment Variables

- Environment variables
  - are a set of pre-defined dynamic values that can affect a running program
  - they are generally part of the operating environment in which a program runs;
  - UNIX (its variants) and Windows all use these as a means of passing information about the environment of a process
  - CGI environment variables are created by the web server and set immediately before the web server executes a gateway script
  - the CGI script can retrieve the values and use the data they send
  - CGI environment variables are defined in
    http://tools.ietf.org/html/rfc3875

CGI Environment Variables

- Can be classified into two major categories:
  - 1. Non-request specific
  - 2. Request specific
- **Non-request-specific** environment variables are the same for all requests:
  - SERVER_SOFTWARE, the name and version of the information server software answering the request

    e.g., SERVER_SOFTWARE = Apache/1.3.15
  - SERVER_NAME, server's hostname, DNS alias, or IP address, e.g., SERVER_NAME = nunki.usc.edu
  - GATEWAY_INTERFACE, the revision of the CGI specification with which this server complies
  - SERVER_PROTOCOL, the name and revision of the information protocol with which this request came in

    e.g., SERVER_PROTOCOL = HTTP/1.0
  - SERVER_PORT, the port number to which the request was sent

    e.g., SERVER_PORT = 8088

CGI Environment Variables (cont'd)

- **Request-specific** environment variables
  - These variables are set depending on each request
    - REQUEST_METHOD, the method with which the request was made; e.g., (GET, POST)
    - PATH_INFO, the extra path information as given by the client; e.g.,

      given http://nunki.usc.edu:8080/cgi-bin/test.cgi/extra/path

      then PATH_INFO = /extra/path
    - PATH_TRANSLATED, the PATH_INFO path translated into an absolute document path on the local system

      PATH_TRANSLATED = /auto/home-scf-03/csci571/WebServer/apache_1.2.5/htdocs/extra/path
    - SCRIPT_NAME, the path and name of the script being accessed as referenced in the URL

      SCRIPT_NAME = /cgi-bin/test.cgi
    - QUERY_STRING, the information that follows the ? in the URL that referenced this script

CGI Environment Variables (cont'd)

- REMOTE_HOST, Internet domain name of the host making the request
- REMOTE_ADDR, the IP address of the remote host making the request
- AUTH_TYPE, the authentication method required to authenticate a user who wants access
- REMOTE_USER, user name that server and script have authenticated
- REMOTE_IDENT, the remote user name retrieved by the server using inetd identification (RFC 1413)
- CONTENT_TYPE, for queries that have attached information, such as POST method, this is the MIME content type of the data
- CONTENT_LENGTH, the length of the content as given by the client

CGI Environment Variables (cont'd)

- Also, every item of information in an HTTP request header is stored in an environment variable
  - Capitalize the name in the request header field
  - Convert dashes to underscores
  - Add the prefix HTTP_
- For example:
  - HTTP_USER_AGENT contains the request header User_Agent field data

  e.g. HTTP_USER_AGENT = Mozilla/4.7 [en]C-DIAL (WinNT; U)
  - HTTP_ACCEPT contains the request header Accept field, of the form type/subtype
  - HTTP_REFERER contains the URL of the document that generated this request

CGI Script Output

- The script sends its output to stdout; the server adds appropriate headers and returns this output to the client
- Output from a script to the server could be:
  - A document generated by a script
  - The type of document could be: HTML, plain text, image, video or audio clip, and many other types
  - Instructions to the server for retrieving the desired output elsewhere
  - an error indicator

Server Directives

- The output of scripts begins with a small header consisting of text lines containing server directives
  - This must be followed by a blank line
- Any headers that are not server directives are sent directly back to the client
- Server directives are used by CGI scripts to inform the server about the type of output
- The current CGI specification defines three server directives:
  - Content-type
  - Location
  - Status

Server Directives (cont'd)

- 1. **Content-type: type/subtype**
  - The MIME type of the document being returned
  - For example,

  content-type: text/html (HTML document)

  content-type: text/plain (plain-text document)
- 2. **Location**
  - Alerts the server that the script is returning a reference to a document, not an actual document
  - If the argument is a URL, the server will issue a redirect to the client; for example,

  location: http://www.ncsa.uiuc.edu/
  - If the argument is a path, the document specified will be retrieved by the server, starting at the document root; for example,

  location: /path/doc.txt

Server Directives (cont'd)

- 3. **Status**
  - This is used to give the server an HTTP/1.1 status line to send to the client
  - The format is nnn xxxx, where
    - nnn is the three-digit status code
    - xxxx is the informative message

  E.g., 403 Forbidden

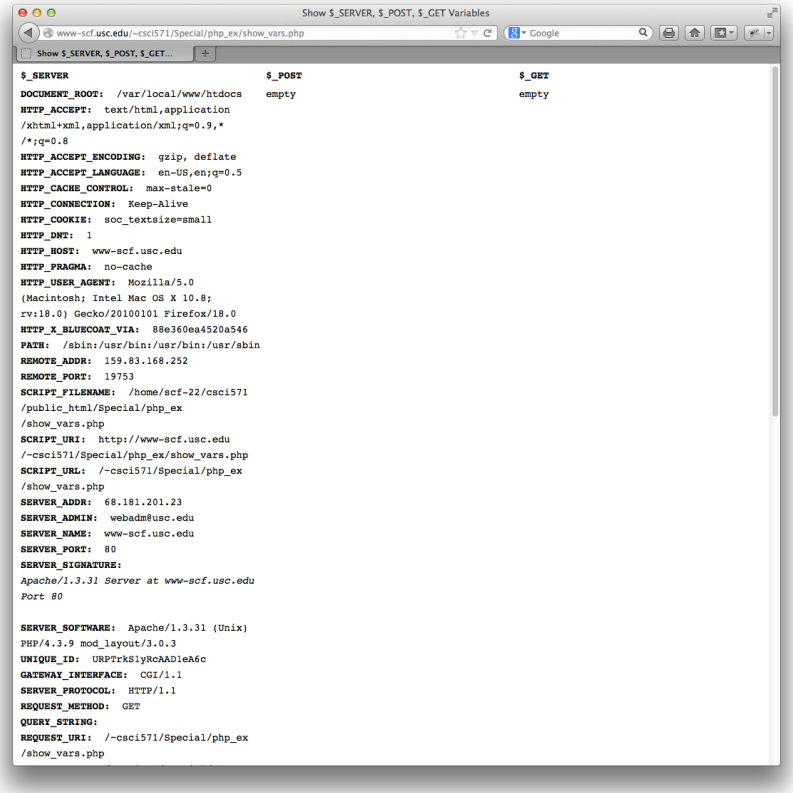
Things to Check Before Running CGI Scripts

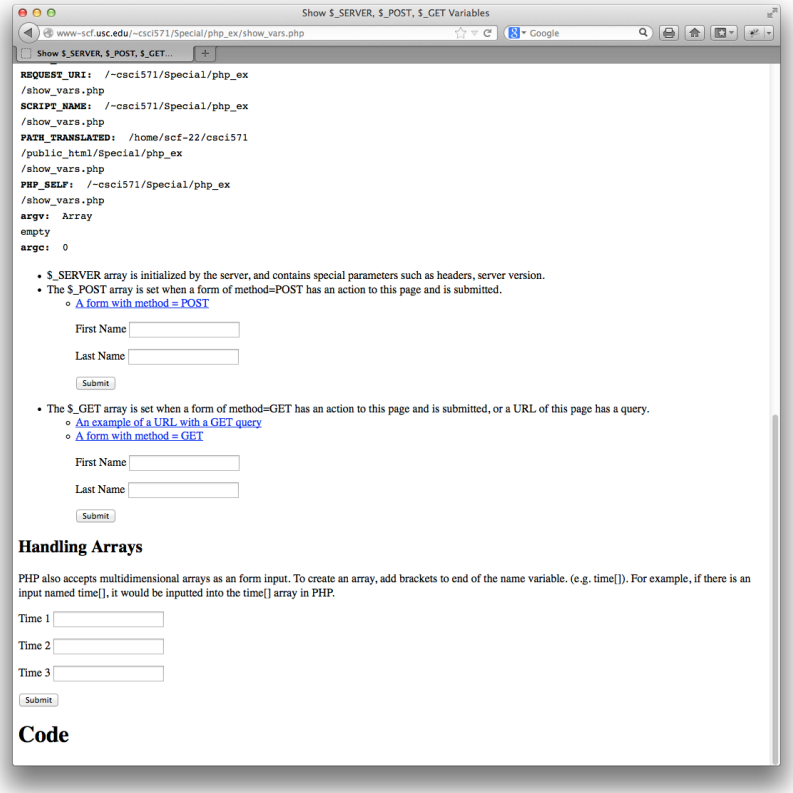
- The following need to be readable and executable by the server
  - CGI scripts
  - Other programs that the scripts call
  - The directory in which the scripts reside
- In UNIX, check the read/write/execute permissions of the files and directories
- In Windows, check the web server settings of the script directories

show_vars.php

- PHP is a language with built-in ability to access environment variables
- show_vars.php is a program that prints environment variables
- The code is available at:

  http://csci571.com/examples/php/show_vars.php
- Below is some sample output





show_vars.php – output tabs & arrays

```
<!doctype html><html>
<head><title>Show $_SERVER, $_POST, $_GET Variables</title></head>
<body>
<?php
   function print_tabs($tabs) {
        for($i = 0; $i < $tabs; $i++) {
                echo "     ";
        }   }
   function print_array($arr, $tabs = 0) {
        if(!empty($arr)) {
                foreach($arr as $k=> $v) {
                        print_tabs($tabs);
                        echo "<b>" . $k . "</b>:  " . $v .
   "<br/>";
                        if(is_array($v)) {
                                print_array($v, $tabs+1);
                        }
                }
        } else {
                echo "empty<br/>";
        }   }
?>
```

show_vars.php – $_SERVER, $_POST, $_GET

```
<table>
   <tr><th width="33%">$_SERVER</th><th
   width="34%">$_POST</th><th
   width="33%">$_GET</th></tr>
   <tr>
<td valign="top">
<?php print_array($_SERVER); ?>
</td>
<td valign="top">
<?php print_array($_POST); ?>
</td>
<td valign="top">
<?php print_array($_GET); ?>
</td>
   </tr>
</table>
```

show_vars.php – POST

```
<ul>
   <li>$_SERVER array is initialized by the server, and contains
   special parameters such as headers, server version.</li>
   <li>The $_POST array is set when a form of method=POST has an
   action to this page and is submitted.
        <ul>
        <li>
<a href="?fname=Hello&lname=World">A form with method = POST</a>
                <div>
                <form method="POST" action="">
                <p>
                <label for="fname">First Name</label>
                <input type="text" value="" name="fname"></p>
                <p>
                <label for="lname">Last Name</label>
                <input type="text" value="" name="lname"></p>
                       <p>
       <input type="submit" value="Submit" name="submit"></p>
                </form></div></li></ul>
```

Discussion session

- Firefox developer Tools
  - Built-in, no download required
  - Similar tools as Firebug, but sometimes named differently
  - Provides additional tools not available in Firebug (now retired)