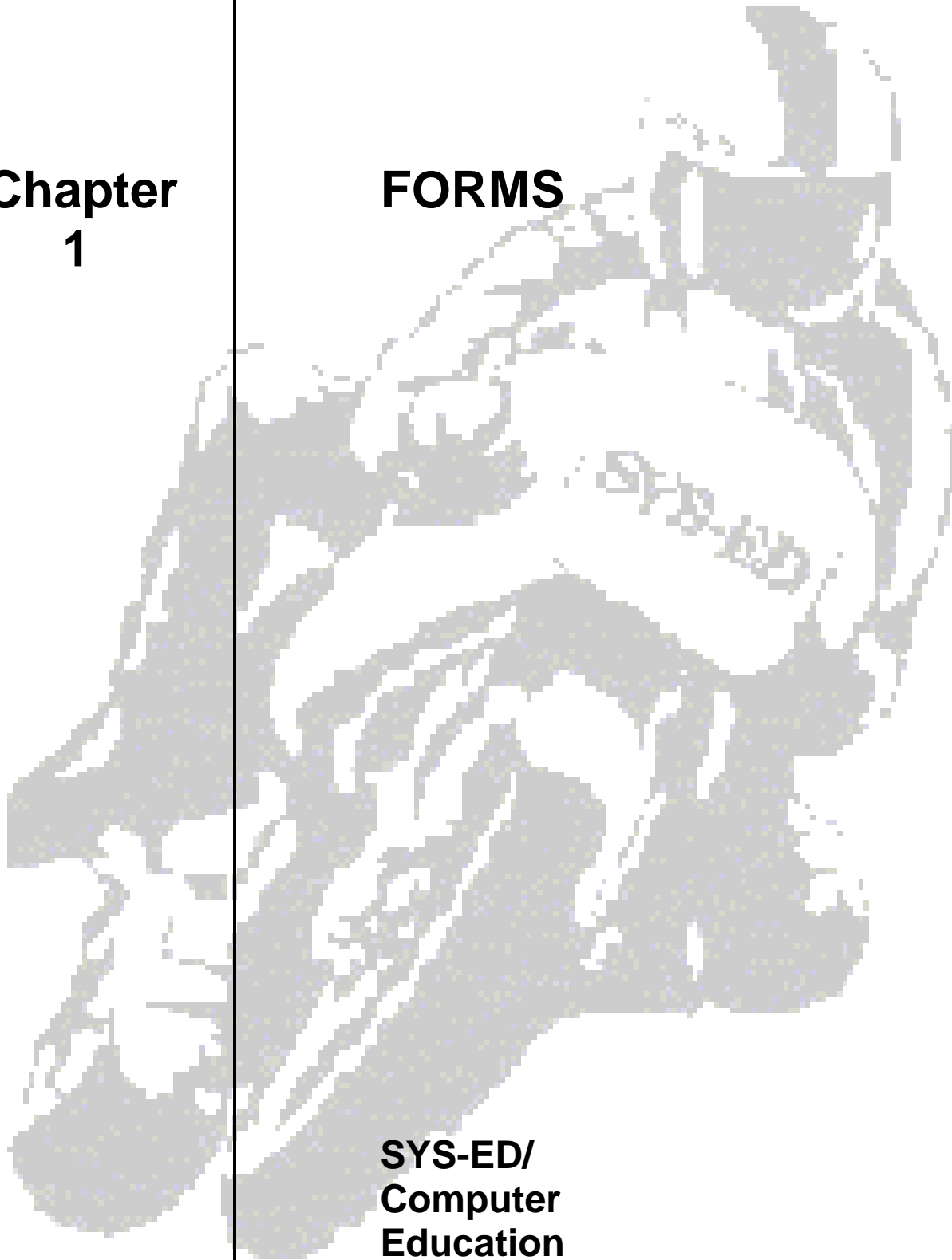


**Chapter
1**

FORMS

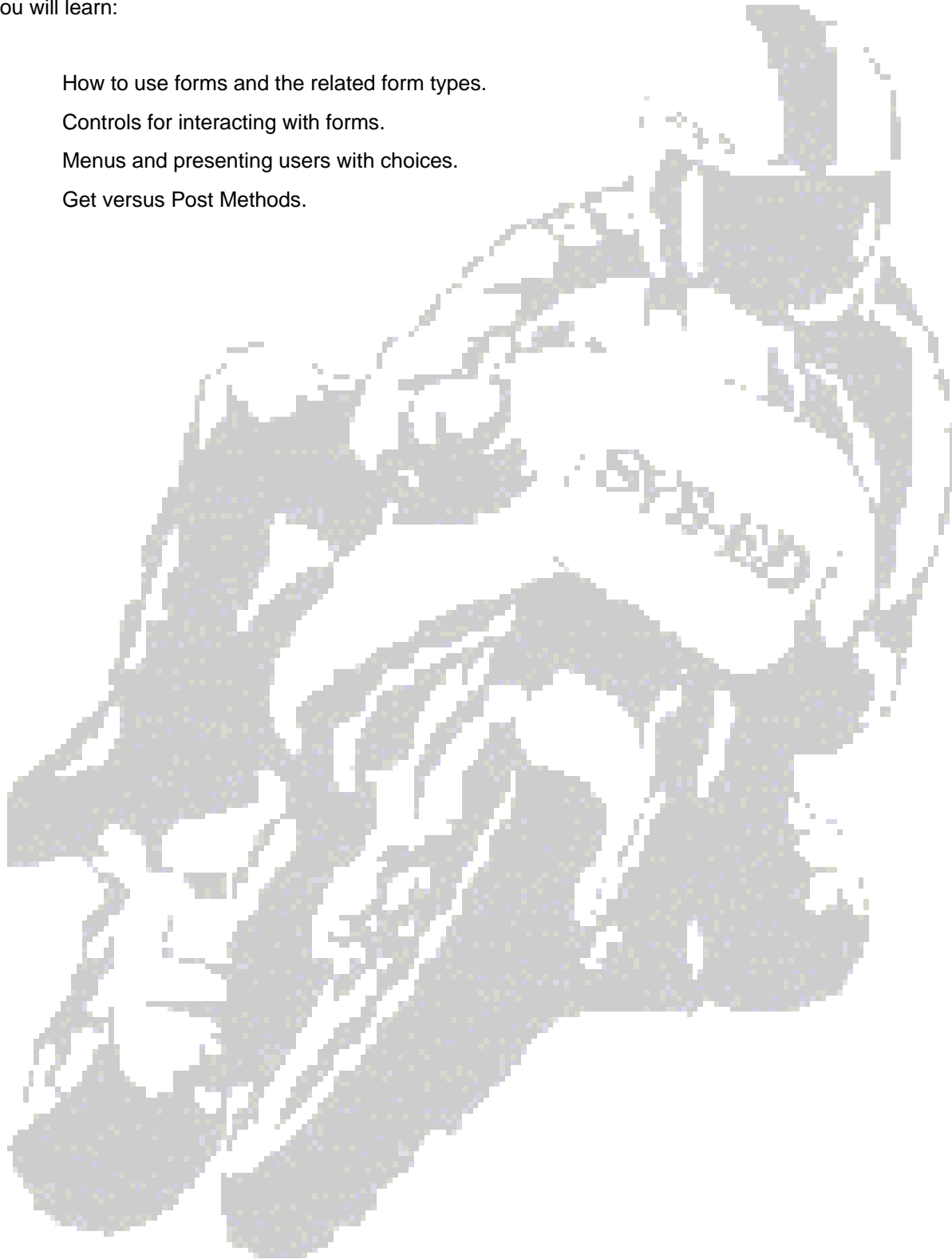


**SYS-ED/
Computer
Education
Techniques, Inc.**

Objectives

You will learn:

- How to use forms and the related form types.
- Controls for interacting with forms.
- Menus and presenting users with choices.
- Get versus Post Methods.



1 Purpose and Features

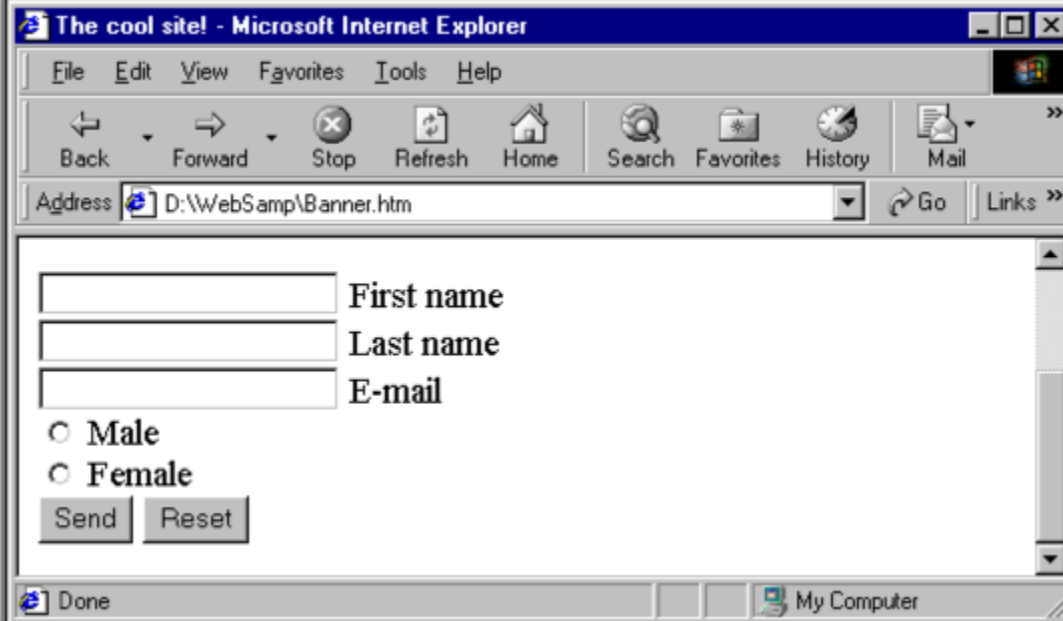
An HTML form is a section of a document containing:

- normal content.
- markup.
- special elements called controls which contain checkboxes, radio buttons, menus, and labels.

Users will generally complete a form by entering text, selecting menu items, etc., before submitting the form to an agent for processing. The agent can be a web server, mail server or script.

Example:

This simple form includes labels, radio buttons, and push buttons.



The screenshot shows a Microsoft Internet Explorer browser window titled "The cool site! - Microsoft Internet Explorer". The address bar displays "D:\WebSamp\Banner.htm". The form content is as follows:

<input type="text"/>	First name
<input type="text"/>	Last name
<input type="text"/>	E-mail

Male
 Female

The form can be reset or submitted.

1.1 <FORM> Element

The <FORM> element acts as a container for controls.

It specifies:

- The layout of the form given by the contents of the element.
- The program that will handle the completed and submitted form; which is known as the action attribute.
- The receiving program must be able to parse name/value pairs in order to utilize them.
- The method by which user data will be sent to the server; which is known as the method attribute.
- A character encoding that must be accepted by the server in order to handle this form; which is known as the accept-charset attribute.
- A form can contain text and markup items such as paragraphs, lists, etc. in addition to form controls.

Syntax:

```
<FORM
ACTION=url
METHOD=get-post
TARGET=window>
```

Attribute	Purpose
ACTION=url	Specifies the address to be used to carry out the action of the form. If none is specified, the base URL of the document is used.
METHOD=get-post	Indicates how the form data should be sent to the server. The form data can be one of two values: GET Appends the arguments to the action URL and opens it as if it were an anchor. POST Sends the data via an HTTP post transaction.

Attribute	Purpose								
TARGET=window	<p data-bbox="560 241 1364 304">Specifies to load the results of the form submission into the targeted window.</p> <p data-bbox="560 346 1364 409">The window must begin with an alpha-numeric character to be valid; except for the following four target windows:</p> <table data-bbox="576 451 1404 714"><tbody><tr><td data-bbox="576 451 755 514">_blank</td><td data-bbox="771 451 1404 514">Specifies to load the link into a new blank window. This window is not named.</td></tr><tr><td data-bbox="576 514 755 577">_parent</td><td data-bbox="771 514 1404 577">Specifies to load the link into the immediate parent of the document the link is in.</td></tr><tr><td data-bbox="576 577 755 640">_self</td><td data-bbox="771 577 1404 640">Specifies to load the link into the same window the link was clicked in.</td></tr><tr><td data-bbox="576 640 755 714">_top</td><td data-bbox="771 640 1404 714">Specifies to load the link into the full body of the window.</td></tr></tbody></table>	_blank	Specifies to load the link into a new blank window. This window is not named.	_parent	Specifies to load the link into the immediate parent of the document the link is in.	_self	Specifies to load the link into the same window the link was clicked in.	_top	Specifies to load the link into the full body of the window.
_blank	Specifies to load the link into a new blank window. This window is not named.								
_parent	Specifies to load the link into the immediate parent of the document the link is in.								
_self	Specifies to load the link into the same window the link was clicked in.								
_top	Specifies to load the link into the full body of the window.								

Example:

```
<FORM TARGET="viewer" ACTION="http://www.sample.com/bin/search">  
  ...  
</FORM>
```

1.2 Successful Controls

A successful control is valid for submission. Every successful control has its control name paired with its current value as part of the submitted form data set.

A successful control must be defined within a FORM element and must have a control name.

However:

- Controls that are disabled cannot be successful.
- If a form contains more than one submit button; only the activated submit button is successful.
- All "on" checkboxes may be successful.
- For radio buttons that share the same value of the name attribute, only the "on" radio button may be successful.
- For menus, the control name is provided by a SELECT element and values are provided by OPTION elements.
- Only selected options may be successful. When no options are selected, the control is not successful and neither the name nor any values are submitted to the server when the form is submitted.

2 Controls

Users interact with forms through named controls. A control's "control name" is given by its name attribute. The scope of the name attribute for a control within a FORM element is the FORM element.

Each control has an initial and a current value; both of which are character strings. A control's "initial value" may be specified with the control element's value attribute. However, the initial value of a TEXTAREA element is given by its contents, and the initial value of an OBJECT element in a form is determined by the object implementation.

The control's "current value" is first set to the initial value. Thereafter, the control's current value may be modified through user interaction and scripts. A control's initial value does not change. Thus, when a form is reset, each control's current value is reset to its initial value. If a control does not have an initial value, the effect of a form reset on that control is undefined.

When a form is submitted for processing, some controls have their name paired with their current value and these pairs are submitted with the form. Those controls for which name/value pairs are submitted are called successful controls.

2.1 Control Types

HTML defines the following control types:

Control Type	Description
Buttons	<p>Authors may create three types of buttons:</p> <ul style="list-style-type: none"> submit When activated, a submit button submits a form. A form may contain more than one submit button. reset When activated, a reset button resets all controls to their initial values. push Push buttons have no default behavior. Each push button may have client-side scripts associated with the element's event attributes.
Checkboxes	These are on/off switches that may be toggled by the user. A switch is "on" when the control element's checked attribute is set. When a form is submitted, only "on" checkbox controls can become successful.
Radio buttons	Radio buttons are like checkboxes except that when several share the same control name, they are mutually exclusive: when one is switched "on", all others with the same name are switched "off".
Menus	Menus offer users options from which to choose.
Text input	Authors may create two types of controls that allow users to input text: single-line input control and multi-line input control.

Control Type	Description
File select	Allows the user to select files in order that their contents may be submitted with a form.
Hidden controls	Authors may create controls that are not rendered; but whose values are submitted with a form.

3 <INPUT> Element

The <INPUT> element specifies a form control.

Syntax:

```
<INPUT  
ALIGN=align-type  
[ CHECKED | ]  
MAXLENGTH=length  
NAME=name  
SIZE=size  
SRC=address  
TYPE=type  
VALUE=value>
```

Attribute	Purpose
ALIGN=align-type	Specifies how the next line of text will be aligned with the image. The align-type can be TOP, MIDDLE, or BOTTOM.
CHECKED	Include CHECKED to set a checkbox or radio button to "selected" when the form first loads.
MAXLENGTH=length	Indicates the maximum number of characters that can be entered into a text control.
NAME=name	Specifies the name of the control.
SIZE=size	Specifies the size of the control (in characters). For TEXTAREA-type controls, both height and width can be specified, using this format: "width,height".
SRC=address	Used when TYPE=IMAGE. Specifies the address of the image to be used.
VALUE=value	For textual/numerical controls, specifies the default value of the control. For Boolean controls, specifies the value to be returned when the control is turned on.

Example:

```
<FORM ACTION="http://intranet/survey" METHOD=POST>
<P>Name
<BR><INPUT NAME="CONTROL1" TYPE=TEXT VALUE="Your Name">
<P>Password
<BR><INPUT TYPE="PASSWORD" NAME="CONTROL2">
<P>Color
<BR><INPUT TYPE="RADIO" NAME="CONTROL3" VALUE="0" CHECKED>Red
<INPUT TYPE="RADIO" NAME="CONTROL3" VALUE="1">Green
<INPUT TYPE="RADIO" NAME="CONTROL3" VALUE="2">Blue
<P>Comments
<BR><INPUT TYPE="TEXTAREA" NAME="CONTROL4" SIZE="20,5" MAXLENGTH="250">
<P><INPUT NAME="CONTROL5" TYPE=CHECKBOX CHECKED>Send receipt
<P><INPUT TYPE="SUBMIT" VALUE="OK"><INPUT TYPE="RESET" VALUE="Reset">
</FORM>
```

4 Choices

The <SELECT> element creates a menu. Each choice offered by the menu is represented by an OPTION element.

A SELECT element must contain at least one OPTION element.

4.1 <SELECT> Element

The <SELECT> element denotes a list box or dropdown list.

Syntax:

```
<SELECT  
  MULTIPLE  
  NAME=name  
  SIZE=n>
```

Attribute	Purpose
MULTIPLE	Indicates that multiple items can be selected.
NAME=name	Specifies a name for the list.
SIZE=n	Specifies the height of the list control.

4.2 <OPTION> Element

The <OPTION> element denotes one choice in a list box.

In a SELECT block, the <OPTION> element denotes one of the choices that will appear in the list.

Syntax:

```
<SELECT  
  SELECTED  
  VALUE=value>
```

Attribute	Purpose
SELECTED	Indicates that this item is the default. If not present, item #1 becomes the default.
VALUE=value	Indicates the value that will be returned if this item is chosen.

4.3 Pre-selected Options

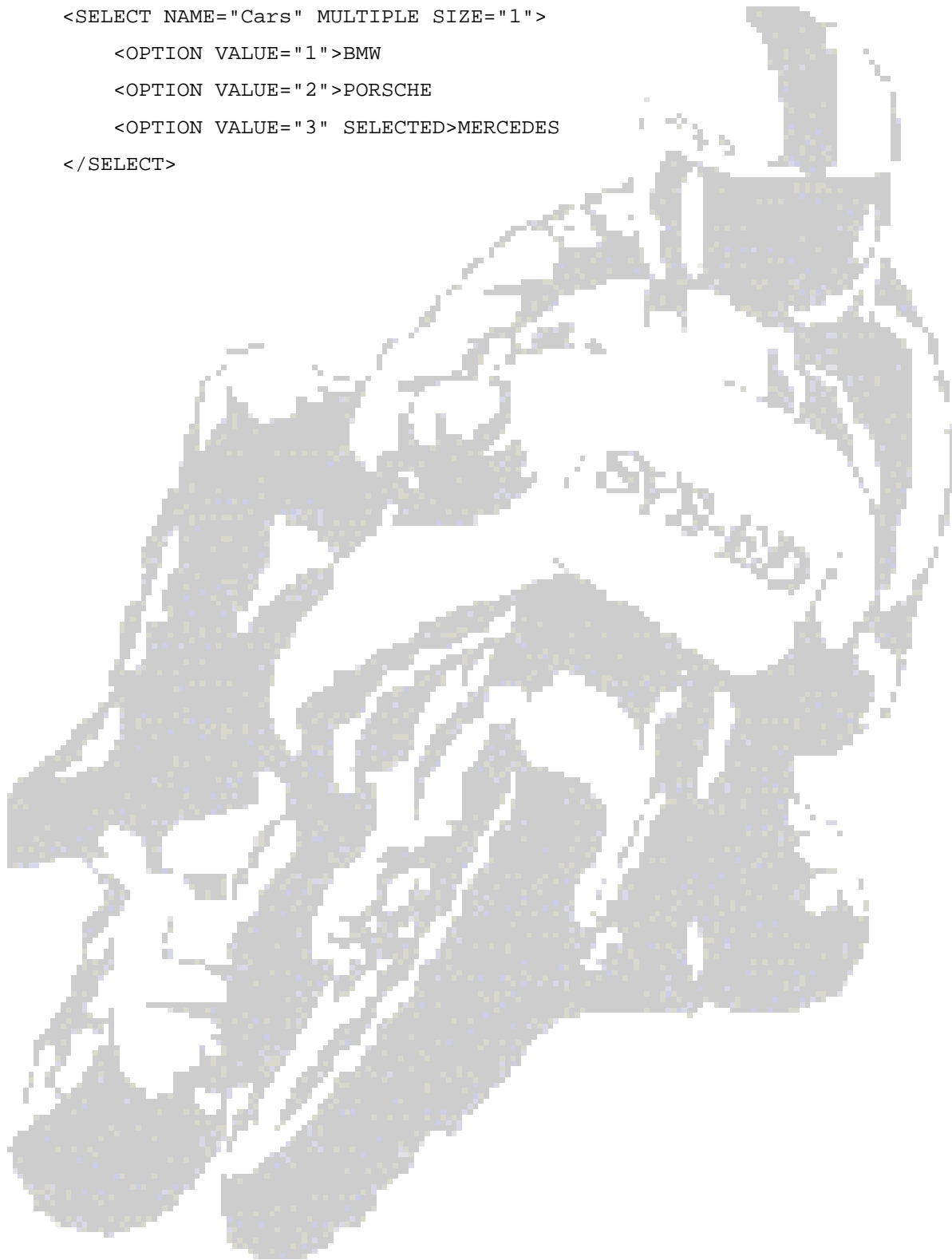
Zero or more choices may be pre-selected for the user.

The following criteria should be used for determining which choices are to be pre-selected.

- If no OPTION element has the selected attribute set, the user agent behavior for choosing which option is initially selected is undefined.
- Since user agent behavior differs, authors should ensure that each menu includes a default pre-selected OPTION.
- If one OPTION element has the selected attribute set, it should be pre-selected.
- If the SELECT element has the multiple attribute set and more than one OPTION element has the selected attribute set, they should all be pre-selected.
- It is considered an error if more than one OPTION element has the selected attribute set and the SELECT element does not have the multiple attribute set.
- User agents may vary in how they handle this error; but should not pre-select more than one choice.

Example:

```
<SELECT NAME="Cars" MULTIPLE SIZE="1">  
  <OPTION VALUE="1">BMW  
  <OPTION VALUE="2">PORSCH  
  <OPTION VALUE="3" SELECTED>MERCEDES  
</SELECT>
```



5 <TEXTAREA> Element

The <TEXTAREA> element creates a multiple-line text entry control in which the user can enter and edit text.

Syntax:

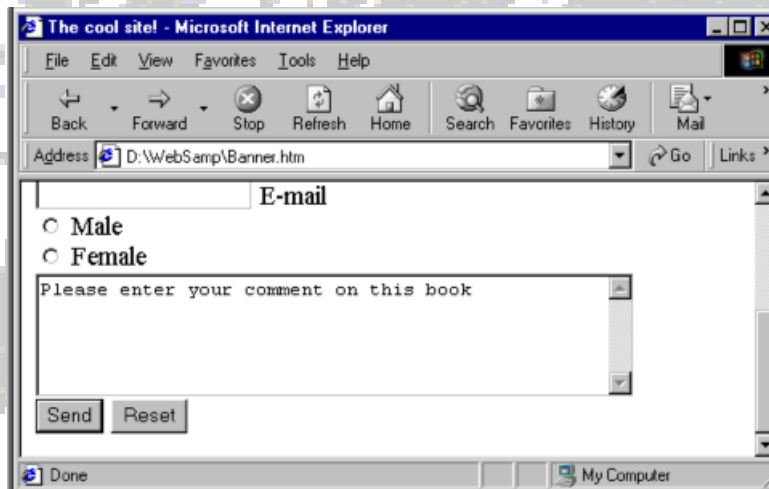
```
<TEXTAREA
COLS=n
NAME=name
ROWS=n>
```

Attribute	Purpose
COLS=n	Sets the width, in characters, of the text area.
NAME=n	Sets the name of the text area. This name is used when the element is used within a FORM element.
ROWS=n	Sets the height, in characters, of the text area.

The end element is required. Any text between the start and end elements is used as the initial value for the control.

Example:

```
<TEXTAREA rows=5 cols=50 name="Desc">
Please enter your comment on this book
</TEXTAREA><BR>
```



6 Get versus Post Methods

The action attribute of the form sends the user information to a script, which will then read it. This can be accomplished through "get" or "post."

6.1 Get Method

With the Get method, the values of the form elements are carried to the querystring in "name=value" pairs. Spaces are substituted with %20, and additional form elements are separated by ampersands. Once the information is passed to the querystring, a CGI script, JavaScript, JSP, or ASP script can use the information.

Get is commonly used when users bookmark pages.

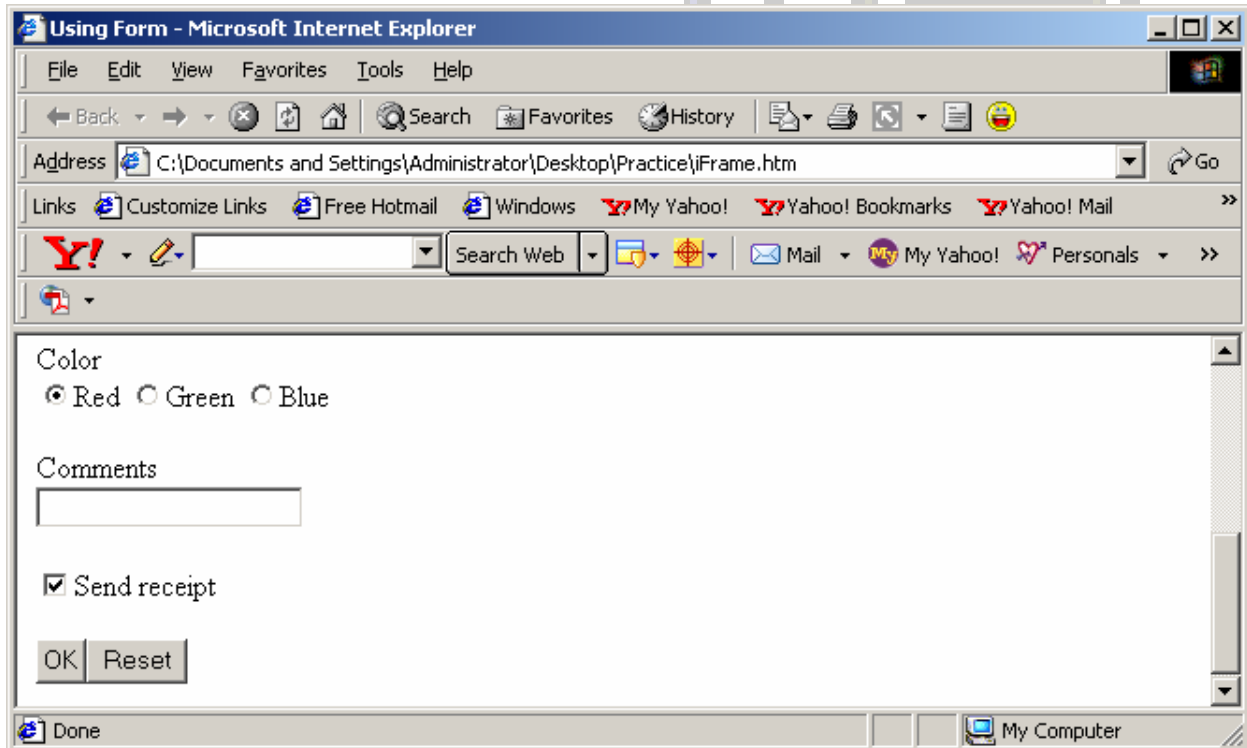
Example:

Create an HTML page with the following content:

```
<html>
<head>
  <title>Using Form</title>
</head>
<body>
<h1>Hello Student</h1>
<hr>

  <FORM ACTION="http://intranet/survey" METHOD=POST>
  <P>Name
  <BR><INPUT NAME="CONTROL1" TYPE=TEXT VALUE="Your Name">
  <P>Password
  <BR><INPUT TYPE="PASSWORD" NAME="CONTROL2">
  <P>Color
  <BR><INPUT TYPE="RADIO" NAME="CONTROL3" VALUE="0" CHECKED>Red
  <INPUT TYPE="RADIO" NAME="CONTROL3" VALUE="1">Green
  <INPUT TYPE="RADIO" NAME="CONTROL3" VALUE="2">Blue
  <P>Comments
  <BR><INPUT TYPE="TEXTAREA" NAME="CONTROL4"
    SIZE="20,5" MAXLENGTH="250">
```

```
<P><INPUT NAME="CONTROL5" TYPE="CHECKBOX" CHECKED>Send receipt  
<P><INPUT TYPE="SUBMIT" VALUE="OK"><INPUT TYPE="RESET" VALUE="Reset">  
</FORM>  
  
</body>  
  
</html>
```



Click OK and check the URL in browser's URL bar.

6.2 Post Method

There are potential problems with the Get Method. In some cases, it will not be a good practice to have the user see the values passed along by the form.

In these circumstances, the Post method should be used. With post, the form information can still be read by a script, but it is not visible to the user and cannot be bookmarked.