FORMS in HTML

Bob Singh

Introduction
Creating Forms
    Start and end of the form
Attributes
    ACTION
    ENCTYPE
    METHOD
        GET method
        POST method
    TARGET
    ID, TITLE, NAME
    CLASS, STYLE, LANG, DIR
    Event-related attributes
Named Input Fields
    Text Box and Password Box
    Check Boxes
    Radio Buttons
    Menus
    Action Buttons
    Text Window

Few websites to refer to for more information on Forms in HTML

http://www.utoronto.ca/webdocs/HTMLdocs/NewHTML/
http://www.webcom.com/html/tutor/forms/start.shtml
http://www.cwru.edu/help/introHTML/toc.html
http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimerAll.html

For different tags used in HTML, this is a good website to refer to:

http://werbach.com/barebones/barebones.html

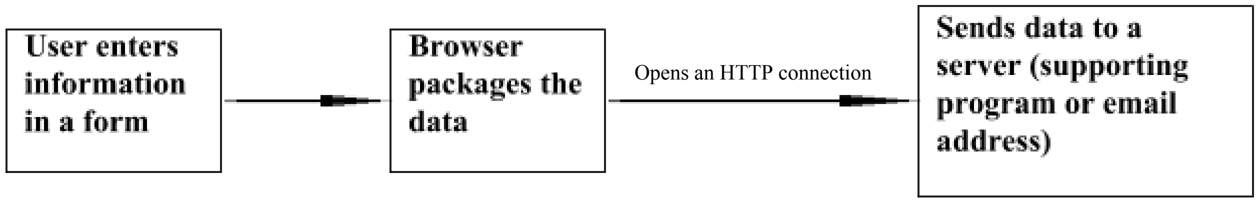
Introduction

An HTML form is a document comprised of form fields. Form fields are objects that allow a user to input information like text input boxes, radio buttons, pull-down menus, checkboxes etc. Forms make HTML interactive, by providing an automated way to interact with the HTML documents.

Forms and Applets provide user-interaction capability, so how are forms different from applets?

Forms and applets both provide user-interaction capabilities, but applets are difficult to write and slow to execute compared to forms. Applets are not standardized for all the browsers. On the other hand, almost every browser, supports forms. You need to know "Swing" in order to implement applets.

What happens when a user clicks on a Submit button in a Form?


User enters information in a form
Browser packages the data
Opens an HTTP connection
Sends data to a server (supporting program or email address)

User enters information into form fields or controls. The browser then packages the data, opens an HTTP connection and sends the data to a server or an email address.

Web servers are programs, which know how to distribute Web pages. They are not programmed to process data from every form. So, they pass the data to an application through CGI(Common Gateway Interface), which processes the information and creates a reply in HTML. Reply could be a simple *"Thank You"* or asking the user to supply missing fields in the form.

When data is sent to an email address, it merely goes to someone's mailbox.

Creating Forms

Start and end of the form

Form is placed inside the body of an HTML document, using `<form>` tag and its end tag `</form>`. If you have more than one form, closing `</form>` tag is important for distinguishing between the multiple forms.

```
<form method=POST action="[name of program]">
</form>
```

`<form>` tag takes different attributes.

Attributes

Forms have different attributes:

ACCEPT-CHARSET	ONKEYPRESS
ACTION	ONKEYUP
CLASS	ONMOUSEDOWN
DIR	ONMOUSEMOVE
ENCTYPE	ONMOUSEOUT
ID	ONMOUSEOVER
LANG	ONMOUSEUP
METHOD	ONRESET
NAME	ONSUBMIT
ONCLICK	STYLE
ONDBLCLICK	TARGET
ONKEYDOWN	TITLE

ACTION

This attribute specifies the URL of the application or the processing script, which is supposed to receive and process the form's data. Without it, the browser would not know, where to send the form data. An action URL should be in this form:

```
protocol://server/path/script_file
```

So, a typical `<form>` tag with the action attribute would look like this:

```
<form action = http://www.xyz.com/cgi-bin/update>
…..
</form>
```

In the above example, URL tells the browser to contact the web server named `www` in the `xyz.com` domain and pass along the user's form values to the application named `update` located in `cgi-bin` directory. This attribute is mandatory in the form.

ENCTYPE

`ENCTYPE` is an attribute, which was introduced by Netscape, for the purpose of providing a file name to be uploaded as form input. You set `ENCTYPE` equal to the MIME type expected for the file being uploaded. This is how this attribute is set:

```
<form method=GET
          enctype="image/gif"
          action="load_upload.cgi">
```

```
</form>
```

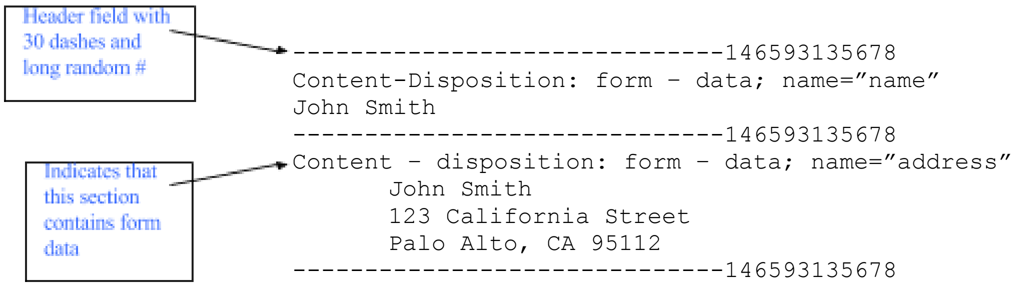
`ENCTYPE` does not create an input field for the filename, rather it tells the browser how the form data is being encoded. There are three types of encoding:

i. **application/x-www-form-urlencoded**
It is the standard encoding, where spaces are converted in the form values to a plus (+) sign, non-alphanumeric characters into a percent sign (%) followed by two hexadecimal digits that are the ASCII code of the character and the line breaks in multiline data into %0D%0A. This also includes a name for each field in the form. As an example, name and address fields are sent from the browser to the server like this:

```
Name=John+Smith%0D%0Aaddress=123+California+Street%0D%0APalo+Alto,
+CA+95112

Output would look like this:
John Smith
123 California Street
Palo Alto, CA 95112
```

ii. The multipart/form-data encoding
This encoding format can be used only when the method attribute of the <form> tag is set to POST. This encoding is required for those forms that contain file selection fields for upload by the user. Each field has its own section. Within each section, one or more header lines define the name of the field followed by one or more lines containing the value of the field. Taking the above example,


Header field with 30 dashes and long random #
Indicates that this section contains form data

```
------------------------------146593135678
Content-Disposition: form - data; name="name"
John Smith
------------------------------146593135678
Content - disposition: form - data; name="address"
      John Smith
      123 California Street
      Palo Alto, CA 95112
------------------------------146593135678
```

iii. The text/plain encoding:
This encoding is used, when you don't have access to a form-processing server and form data needs to be sent by email (the form's "action" attribute is set to "mailto URL"). In this encoding, each element in the form is placed on a single line, with the name and value separated by an equal to sign. The above example, will look like this:

```
name=John Smith
address=123 California Street%0D%0APalo Alto, CA 95112
```

METHOD
Method specifies to HTTP, which method to use, when passing the data to the server for processing. It can be set to values of `GET` and `POST`.

`GET` method contacts the form-processing server and send the form data in a single transmission. The data is appended to the end of the URL of the application or processing script, separated by a question mark character. As an example, this is how you can code for `GET` method:

```
<form method=GET
  action=http://www.xyz.com/cgi-bin/update>
</form>
```

`POST` method sends the form data to the server in a separate HTTP transaction. The browser sends the data in two steps: the browser first contacts the form processing server specified in the action attribute and once contact is made, it sends the data to the server in a separate transmission.

Which method to use?

GET method	**POST method**
i. Should be used to send forms with a few short fields. ii. Not a very secure method of form data transmission iii. Easy to build a `CGI` script to support `GET` because you don't need extra steps of decoding parameters like in `POST` iv. Lets you include form-like parameters as part of the `URL`	i. Used to send forms with many long text fields. ii. More secure method of form data transmission. iii. Not easy to build a `CGI` script to support `POST` because you need extra steps of decoding `POST` parameters. iv. POST-style applications, on the other hand, require an extra transmission from the browser, after the URL.

If there is a CGI script, which takes a number as an input and performs different actions based on that number; then you can explicitly code the value in the anchor tag that invoked the script, rather than having user fill in the form. As an example,

```
<a href=//www.cgiserver.com/cgi-bin/myscript?num=4>
```

This code would invoke the script "`myscript`" on the server `www.cgiserver.com`, passing a value of 4 for the variable num. You need to use "GET" method for this. This works fine, if you have only one variable. If you have two or more variables, you need to precede each variable name, after the first with an ampersand (&). The same ampersand may be considered as a character insertion character within href. So, you must use &#38 or &amp; as a substitute for literal ampersand.

```
<a href=//www.cgiserver.com/cgi-bin/myscript?num=4&num2=6>
```

needs to be replaced with

```
<a href=//www.cgiserver.com/cgi-bin/update?num=4&amp;num2=6>
```

TARGET

`TARGET` can be used to redirect the results of a form to another window or form or frame. You can add the target attribute to the `<form>` tag and provide the name of the window or frame to receive results.

```
<form method=GET
      action="http://www.xyz.com/cgi-bin/update "
      target="main">
</form>
```

ID, TITLE, NAME

The `id` attribute gives a unique string label to the form for reference by programs and hyperlinks. The `title` attribute defines a quote-enclosed string value to label the form. It gives this value only to the form-segment, which cannot be used by other programs or hyperlinks.

CLASS, STYLE, LANG, DIR attributes

The `class` attribute lets you format the content according to a predefined class of the `<form>` tag. The `style` attribute creates a style for the elements enclosed by the form, overriding any other style in effect. This attribute affects the body content, text in general. The `lang` attribute lets you define the language used within the form. The `dir` attribute tells the browser, which direction to display the list contents – from left to right(`dir=ltr`) or right to left(`dir=rtl`).

Event-related attributes

Forms have two main event-related attributes: `onSubmit` and `onReset`. With `onSubmit`, the browser executes these commands before it actually submits the form's data to the server or sends it to an email address. This event may be used for scanning the form's data and prompting the user to complete one or more missing elements. It may also be used for letting the users know, when form's data is sent by email. The `onReset` attribute is similar to the `onSubmit` attribute except that it is executed, when the user presses "Reset" button. The value of these event attributes is enclosed in quotation marks.

Using Email to collect data from a form

There are many users, who don't have access to a web server; so they cannot create or manage CGI programs. You can use email to collect data from the form, by using mailto URL for the form's action attribute. For example, the syntax would something like this:

```
<form method=POST action="mailto://sukhjit@singh.com"
OnSubmit="window.aler('This form is being sent by email')">
```

```
</form>
```

There are some issues with Email forms.

- If you use mailto or form-to-email facility, it might not work on all the browsers because all the browsers don't support it.
- There is no confirmation page to assure the user that their form has been processed.
- Your data may arrive in a form that is difficult to read, unless you use a readable `enctype` such as `text/plain`.

Named Input Fields

The named input fields compose the bulk of a form. The fields appear as GUI controls such as text boxes, check boxes, radio buttons, drop-down menus, Each control is given a specific name, which becomes a variable name in the processing script.

`<input>` tag handles most of the named input fields. `<input>` can place most of the fields on your form. `<input>` tag requires the following attributes: `type and name.` `Name` attribute should not have any alphanumeric characters. This tag can be used to define the following controls:

- Text Boxes
- Multiple-choice list
- Clickable images
- Action Buttons
- Radio Buttons
- Check Boxes

Each control is given a specific name, which becomes a variable name in the processing script.

Text Box and Password Box

These controls take text. The only difference between the two is that the text typed in the password box appears on the screen as asteriks.

```
<input type=text|password name=name [value="default_text"]
[size=width] [maxlength=max_width]>
```

Here, the "`name`" attribute is mandatory because it provides a unique name to the data entered in the field. "`value`" attribute enables you to place some default text in the field rather than having it blank.

Text Box
You can ask the browser to display a text input box of certain length by using `size`, restrict the number of inputted characters by using `maxlength.`

```
<input type=text name=name size=30 maxlength=256>
```

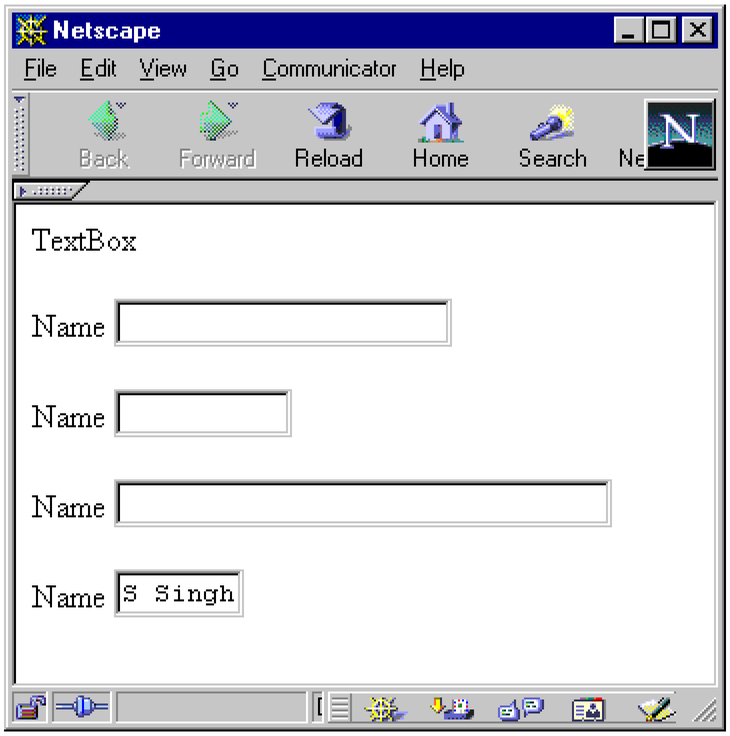
Here, you have a text input box 30 characters wide into which the user may type 256 characters. The browser automatically scrolls text inside the text box.

```
<input type=text name=name size=3 maxlength=3 value=”abc”>
```

In this example, you have a text input box 3 characters wide into which the user may type 3 characters, with an initial value of “`abc`”.

```
<form>
     TextBox
    <p>
      Name <input type=text name=name >
    <br>
    <br>
       Name <input type=text name=name size = 10 maxlength=10>
    <br>
    <br>
       Name <input type=text name=name size = 30 maxlength=256>
    <br>
    <br>
       Name <input type=text name=name size = 7 maxlength=7 value=”S Singh”>
</form>
```

Output of the above form will look like this:


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Ne

TextBox

Name

Name

Name

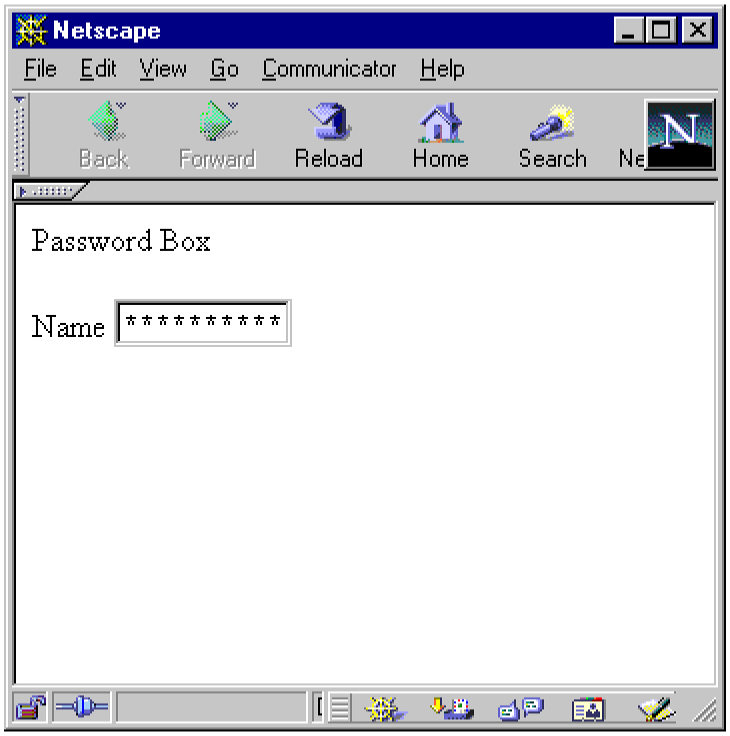
Name S Singh

For the first line, the browser's default width is given to the textbox. In the second line of the code, a textbox with a width of 10 is created and it accepts a maximum of 10 characters. In the third line, it shows a textbox of size 30 with a maximum number of accepted characters being 256. In fourth line, it shows a textbox with an initial value of "S Singh" with the control being seven characters wide and lets the user type only seven characters.

Password Box
It behaves like a textbox except that the user-typed characters don't appear on the screen. This text control is not secure because the browser transmits text as unencrypted, when the form is submitted to the server. To create a password box, set the value of the type attribute to password. All other attributes are the same. For example,

```
<form>
          Password Box
           <br>
           <br>
           Name <input type=password name=name size = 10 maxlength=10>
</form>
```

Output looks like this:


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Ne

Password Box

Name **********

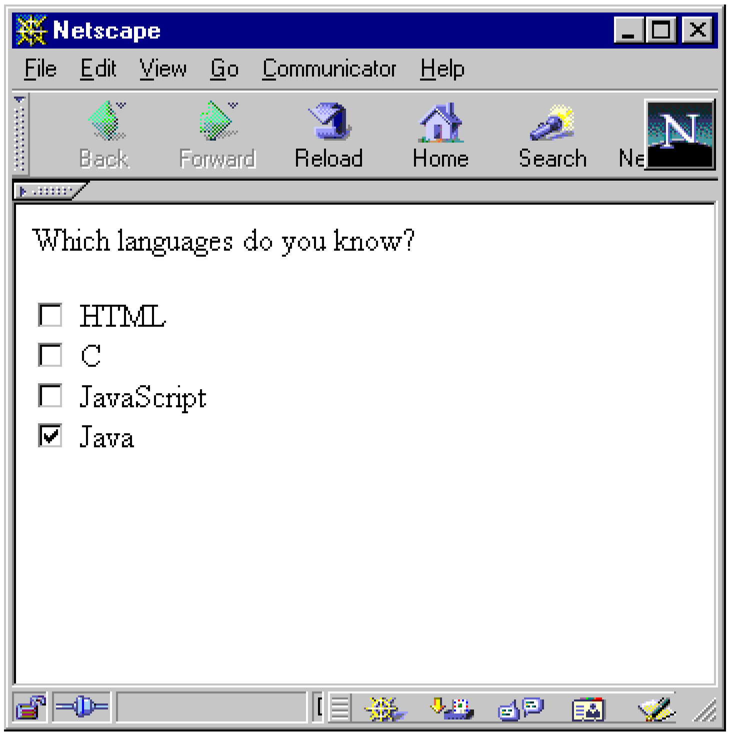
Check Boxes
Check Boxes provide users with several choices, from which they can select as many choices as they wish to. Each check box should have its own unique name.

```
<input type=checkbox name=name value="value" [checked]>
```

You set the `type` attribute for each `<input>` tag to `checkbox`. The value attribute specifies which data is being sent to the server, if the corresponding check box is selected. The optional checked attribute tells the browser to display a checked box by default, unless the user deselects it.

```
<form>
   Which languages do you know?
    <p>
         <input type=checkbox name=language value="HTML"> HTML
    <br>
         <input type=checkbox name=language value="C"> C
    <br>
         <input type=checkbox name=language value="javascript"> JavaScript
    <br>
         <input type=checkbox name=language checked value="java"> Java
</form>
```

Output will look like this in a browser window:


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Ne

Which languages do you know?

☐ HTML
☐ C
☐ JavaScript
☑ Java

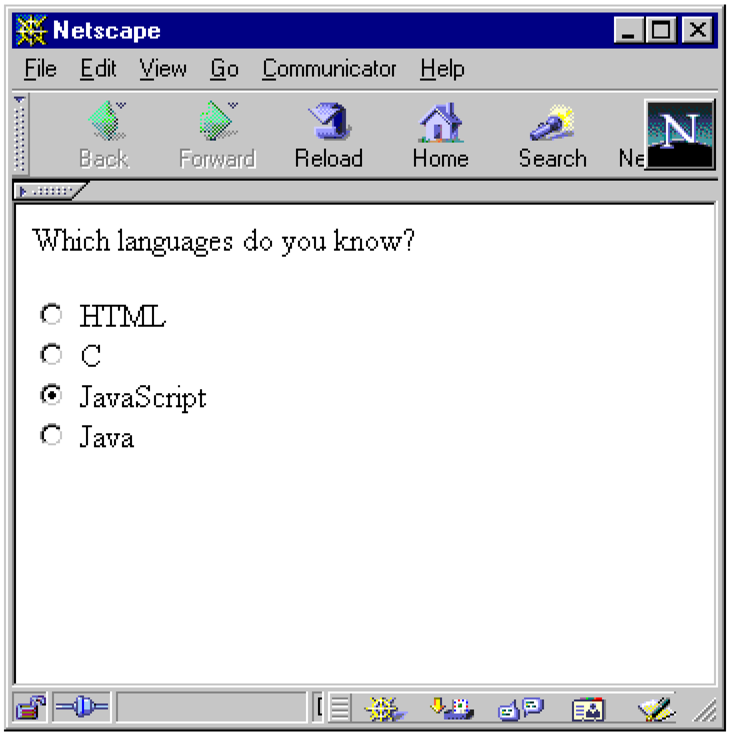
Here because of the "checked" attribute, particular checkbox is checked by default. It can be unchecked, if required.

Radio Buttons

Radio buttons are similar to checkboxes, except they provide user with a set of choices, from which the user can choose only one. You can create a radio button by setting `type` attribute of the `<input> to radio`. Radio buttons also require a name and value attribute. Radio buttons with the same name are a member of one group. You can have one of the choices selected as a default by including `checked` attribute with the element. Using above example,

```
<form>
   Which languages do you know?
    <p>
          <input type=radio name=language value="HTML"> HTML
    <br>
          <input type=radio name=language value="C"> C
    <br>
          <input type=radio checked name=language value="javascript">
JavaScript
     <br>
          <input type=radio name=language value="java"> Java
</form>
```

Here, `"JavaScript"` will be checked by default. It can be deselected if the user selects some other option. Output of the above form will be something like this with three radio buttons:


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Ne

Which languages do you know?

○ HTML
○ C
● JavaScript
○ Java

Menus

In order to create pull-down or scrollable option menus, you can use `<select>` and `</select>` container tags. The HTML code used to create a general menu is:

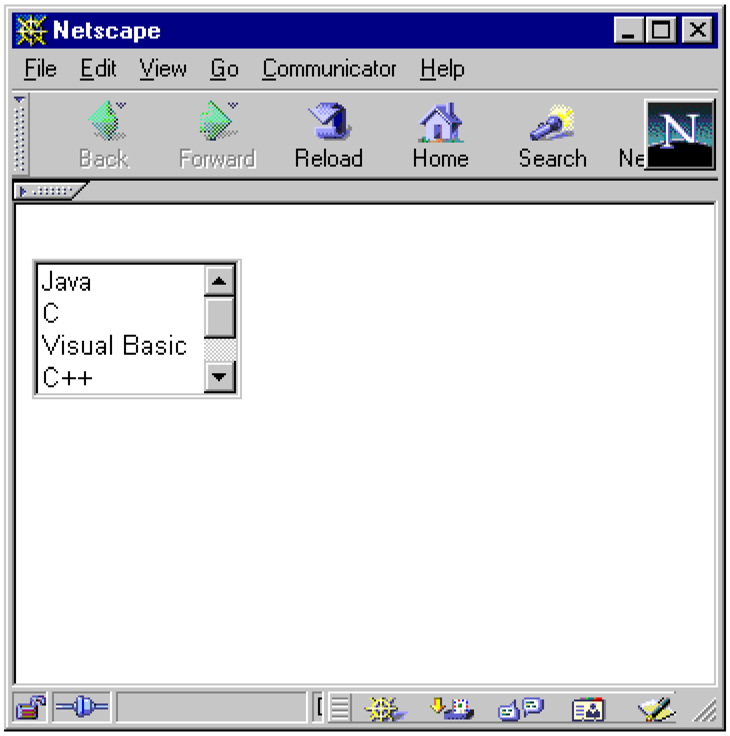
```
<form>
<select name=name [size="size"] [multiple]>
```

```
<option [selected]>option 1 </option>
<option [selected]> option 2 </option>
…..
</select>
…..
</form>
```

In the `<select>` tag, name attribute gives a unique identifier to the input field. The optional `size` attribute enables you to specify how many options should be displayed, in the browser window. The default `size` is 1. In order to select multiple options at a time, you have to add `multiple` attribute to the `<select>` tag. By default, all options within a multiple-choice `<select>` tag are unselected. You can include the `selected` attribute in the <option> tag to pre-select one or more options.

```
<form>
       <br>
       <select name=language multiple size=4>
       <option> Java
       <option> C
       <option value=VB> Visual Basic
       <option value=C++>C++
       <option value=corba>Corba
       <option value=xml>XML
       </select>
</form>
```

Here, four options will be displayed in the browser window and the user will be able to select multiple choices by holding the Ctrl key down (because of the use of "`multiple`" attribute)


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Ne
Java
C
Visual Basic
C++

```
<option value = VB> Visual Basic
<option> Visual Basic
```

In the above example, both will display “Visual Basic” – the only difference is that the first is set within `<option>` tag and the second one defaults to the content of the `<option>` tag itself.
You can use `<optgroup>` tag to logically group different options like this:

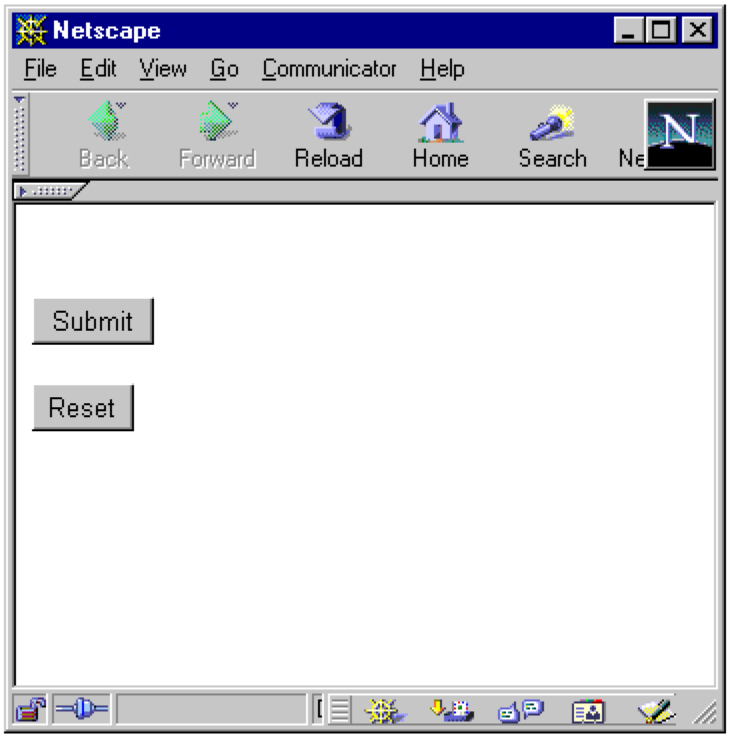
```
<form>
       <select name=languages multiple size=6>
       <optgroup label=Compiled languages>
              <option> C
              <option> C++
       </optgroup>
       <optgroup label =Interpreted languages>
              <option>Shell Scripting
              <option>Perl
              <option>Java
       </optgoup>
       </select>
</form>
```

Currently, **no browser supports `<optgroup>` tag**.

Action Buttons

Action Buttons can be of two types: **`Submit`** and **`Reset`**.
Submit button: Clicking the `Submit` button instructs the browser to package the form data and send it to the server. **Reset button:** Clicking the `Reset` button clears out any data entered in the form and sets all the named input fields to their default values. You can use `<input>` tag with `type` attribute to be `submit` or `reset`. You can also create a custom image to be a `Submit` button for your forms.

```
<form>
       <br>
       <br>
       <input type=submit value=”Submit Data”>
       <br>
       <br>
       <input type=reset value=”Reset Data”>
</form>
```


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Ne
Submit
Reset

You should set the `value` to a text string that clearly describes the function of the button. If `value` is not specified, the button text is "`Submit`" and "`Reset`".

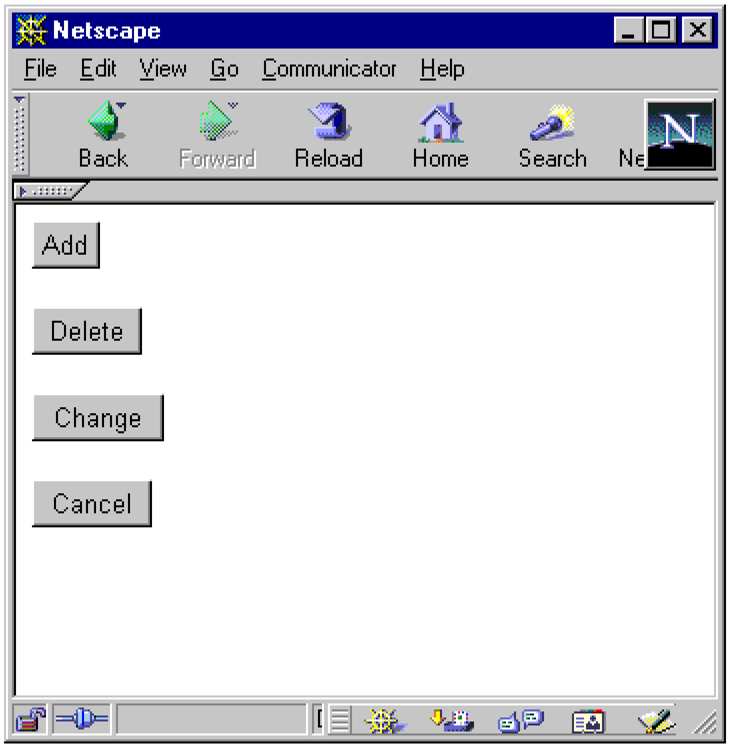
In order to **create custom image button**, you use the `<input>` tag with type as `image`. Image buttons require a `src` attribute with the URL of the image file. You can also use `align` and `border` attributes to control image alignment.

```
<form>
       <input type=submit src="pictures/abc.gif" name=map align top>
</form>
```

Multiple buttons in a single form
You can have several buttons of the same or different types in a single form. You can make the name of each button different, but it is easier to name all similarly acting buttons the same.

```
<form>
       <input type=submit name=action value=Add>
       <br>
       <input type=submit name=action value=Delete>
       <br>
       <input type=submit name=action value=Change>
       <br>
       <input type=submit name=action value=Cancel>
</form>
```

Output will look like this:


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search
Add
Delete
Change
Cancel

Push buttons
Using the `<button>` tag, you can create push buttons, which merely give you control over how the element gets displayed by the browser. This tag provides a greater variety and richer content over <input> tag. The `value` attribute can be used to set the label on the button, the `name` attribute if specified, will cause the supplied value to be passed to the form processing script. The HTML 4.0 standard is not clear as to what display enhancements to a form should the control `<button>` provide, other than that it provides 3D appearance to the button and visually it appears to in and back out when pressed. No browser yet supports `<button>`. You can use type attribute to define button's action. Its value can be set to submit, reset or button. Like `<input>,` a `<button type=submit>,` when selected by the user tells the browser to package and send the contents of the form to the form-processing server or email. When `type=reset` is used, it creates a reset button.

```
<button type = submit>
                   Send it <img src="sendit.gif" align=middle>
</button>
```

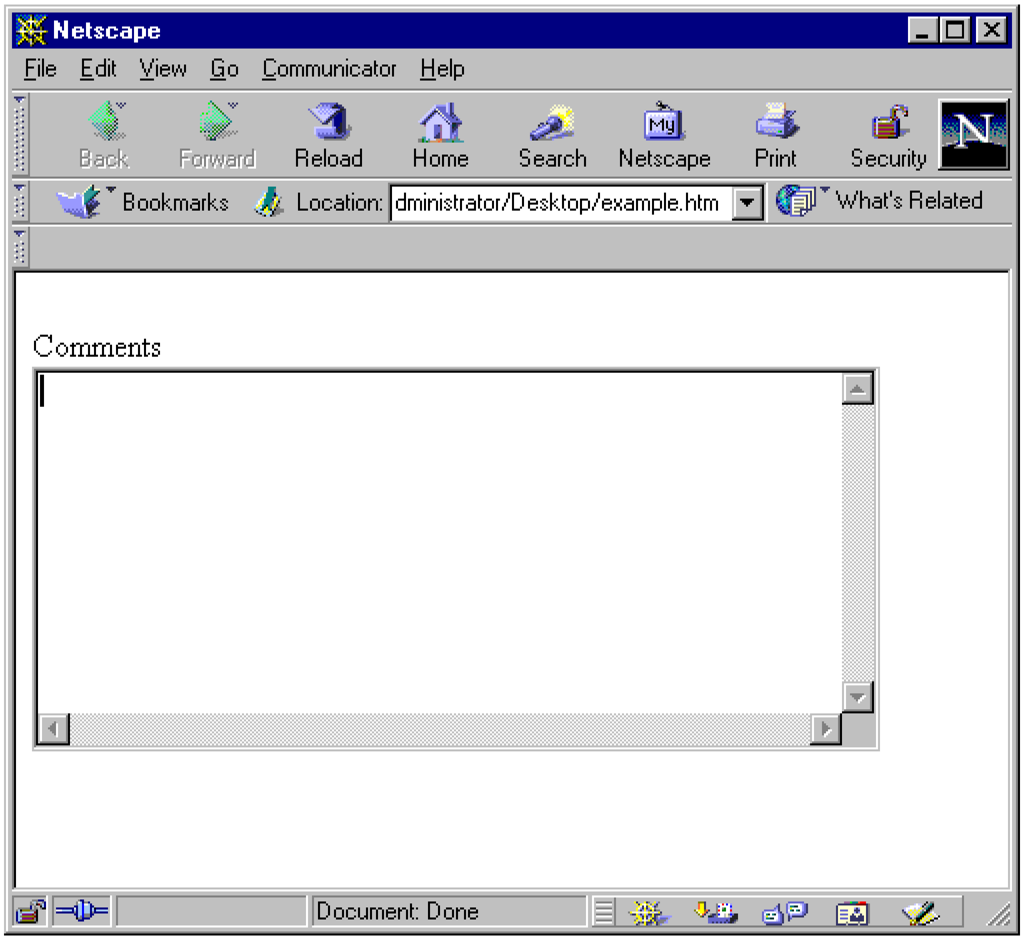
Text Window
Text boxes and password boxes are used for simple one-line input fields. You can create multiple line text windows using `<textarea>` and `</textarea>` container tags. The HTML syntax for that is:

```
<textarea name=name [rows="rows"] [cols="columns"]>
Default window_text
```

```
</textarea>
```

The name identifier gives the text window a unique name, the optional `rows` and `cols` identifier attributes give the dimensions of the text window in the browser. The default number of rows and columns vary from browser to browser. You can include word wrap within the text area, otherwise text typed in the text area is transmitted to the server exactly as typed. With the wrap attribute set to virtual, the text is wrapped within the text area for the presentation to the user, but it is transmitted to the server (just like it was typed).

```
<form>
       <br>
       Comments
       <textarea name=comments cols=50 rows=10>
       </textarea>
</form>
```


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security
Bookmarks Location: dministrator/Desktop/example.htm What's Related
Comments
Document: Done