*Session 3 - Forms and frames HTML forms provide a simple and reliable user interface to collect data from the user and transmit the data to a servlet or other server-side program for processing.

A form will take input from the site visitor and then will post your back-end application such as CGI, ASP Script or PHP script etc. Then your back-end application will do required processing on that data.

Elements like text fields, textarea fields, drop-down menus, radio buttons, checkboxes, etc. which are used to take information from the user.

How HTML Forms Transmit Data

HTML forms let you create a variety of user interface controls to collect input in a Web page. Each of the controls typically has a name and a value, where the name is specified in the HTML and the value comes either from user input or from a default value in the HTML. The entire form is associated with the URL of a program that will process the data, and when the user submits the form (usually by pressing a button), the names and values of the controls are sent to the designated URL as a string of the form .

This string can be sent to the designated program in one of two ways: GET or POST. The first method, an HTTP GET request, appends the form data to the end of the specified URL after a question mark. The second method, HTTP POST, sends the data after the HTTP request headers and a blank line.

A simple syntax of using $< form >$ is as follows:

```
<form action = "back-end script" method = "posting method">
    form elements like input, textarea etc.
</form>
```

Most frequently used form attributes are:

- **name:** This is the name of the form.
- **action:** Here you will specify any script URL which will receive uploaded data.
- **method:** Here you will specify method to be used to upload data. It can take various values but most frequently used are GET and POST.
- **target:** It specifies the target page where the result of the script will be displayed. It takes values like _blank, _self, _parent etc.

There are different types of form controls that you can use to collect data:

COPYRIGHT ©TALENTSPRINT, 2015. ALL RIGHTS RESERVED.

- Text input controls
- Buttons
- Checkboxes and radio buttons
- Select boxes
- File select boxes
- Hidden controls
- Submit and reset button

Text input controls

HTML supports three types of text-input elements: textfields, password fields, and text areas. Each is given a name, and the value is taken from the content of the control. The name and value are sent to the server when the form is submitted, which is typically done by means of a submit button.

There are actually three types of text input used on forms:

- **Single-line text input controls:** Used for items that require only one line of user input, such as search boxes or names. They are created using the $< input >$ element.
- **Password input controls:** Single-line text input that mask the characters a user enters.
- **Multi-line text input controls:** Used when the user is required to give details that may be longer than a single sentence. Multi-line input controls are created with the $< textarea >$ element.

Following is the list of attributes for $< input >$ tag.

- **type:** Indicates the type of input control you want to create. This element is also used to create other form controls such as radio buttons and checkboxes.
- **name:** Used to give the name part of the name/value pair that is sent to the server, representing each form control and the value the user entered.
- **value:** Provides an initial value for the text input control that the user will see when the form loads.

- **size:** Allows you to specify the width of the text-input control in terms of characters.
- **maxlength:** Allows you to specify the maximum number of characters a user can enter into the text box.

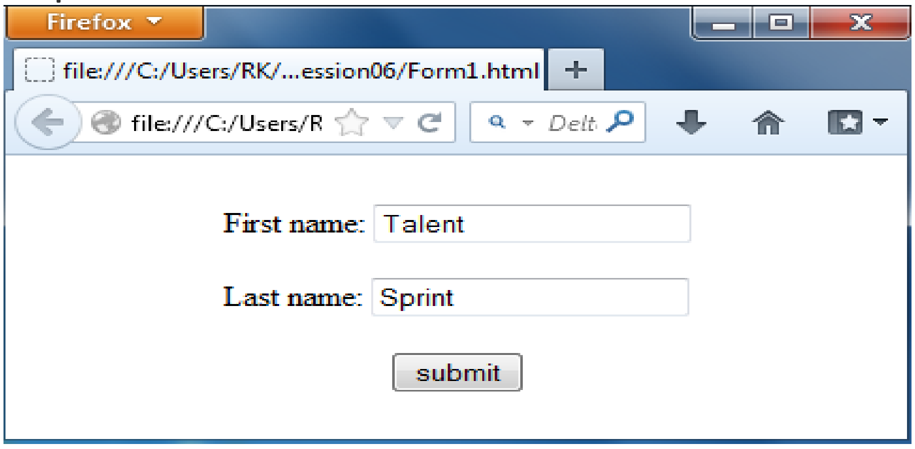
Single-line text input controls

Single-line text input controls are created using an $< input >$ element whose type attribute has a value of text.

Example:

```
<form>
    First name:
        <input type = "text" name = "first_name" value = "Talent"/> <br>
    Last name:
        <input type = "text" name = "last_name" value = "Sprint"/> <br>
        <input type = "submit" value = "submit" />
</form>
```

Output:



First name: Talent

Last name: Sprint

submit

Password input controls

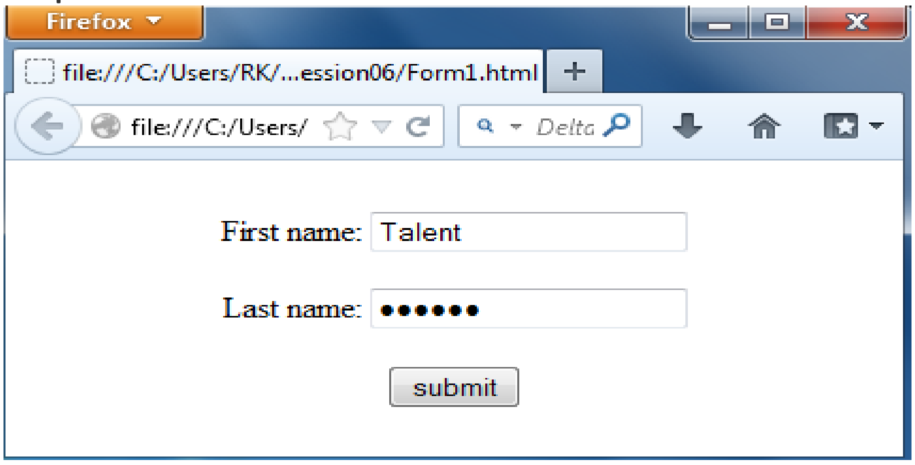
This is also a form of single-line text input controls are created using an $< input >$ element whose type attribute has a value of password.

Example:

```
<form>
```

```
        Login :
        <input type = "text" name = "login" value = "Talent"/> <br>
        Password:
        <input type = "password" name = "password" value = "Sprint"/> <br>
        <input type = "submit" value = "submit" />
    </form>
```

Output:



First name: Talent

Last name: ●●●●●●

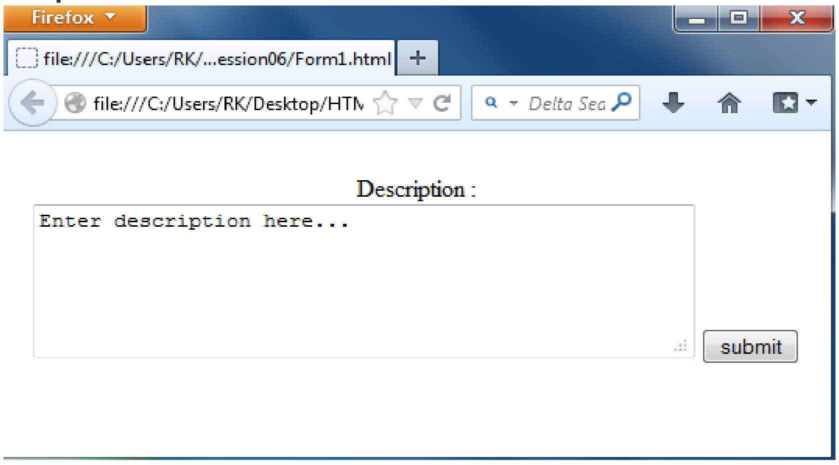
submit

Multiple-Line Text Input Controls

If you want to allow a visitor to your site to enter more than one line of text, you should create a multiple-line text input control using the $< textarea >$ element.

```
    <form>
        Description : <br />
            <textarea rows = "5" cols = "50" name = "description">
                 Enter description here...
            </textarea>
            <input type = "submit" value = "submit" />
    </form>
```

Output:


Firefox
file:///C:/Users/RK/...ession06/Form1.html
file:///C:/Users/RK/Desktop/HTM
Delta Sea
Description :
Enter description here...
submit

Following is the detail of above used attributes for $< textarea >$ tag.

- **name:** The name of the control. This is used in the name/value pair that is sent to the server.
- **rows:** Indicates the number of rows of text area box.
- **cols:** Indicates the number of columns of text area box.

Creating Button

The buttons are used for two main purposes in HTML forms, to submit forms and to reset the controls to the values specified in the original HTML. Browsers that use JavaScript can also use buttons for a third purpose, to trigger arbitrary JavaScript code.

Traditionally, buttons have been created by the INPUT element used with a TYPE attribute of SUBMIT, RESET, or BUTTON.

When you use the $< input >$ element to create a button, the type of button you create is specified using the type attribute.

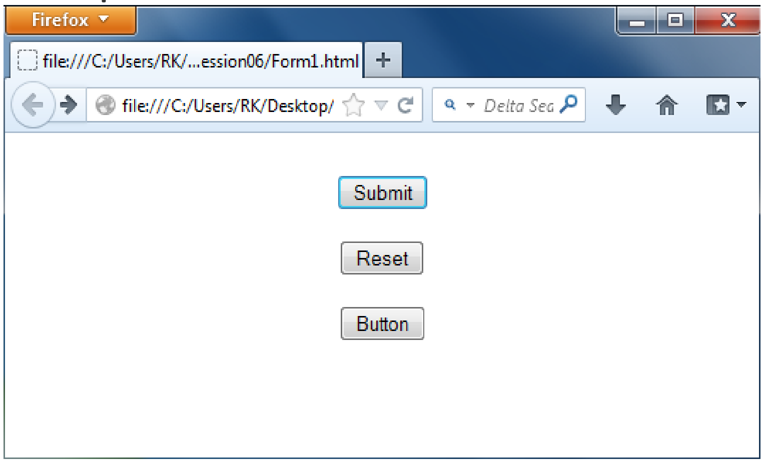
The type attribute can take the following values:

- **submit:** This creates a button that automatically submits a form.
- **reset:** This creates a button that automatically resets form controls to their initial values.
- **button:** This creates a button that is used to trigger a client-side script when the user clicks that button.

Example:

```
<form>
    <input type = "submit" name = "Submit" value = "Submit" />
    <br /><br />
    <input type = "reset" value = "Reset" />
    <input type = "button" value = "Button"  />
</form>
```

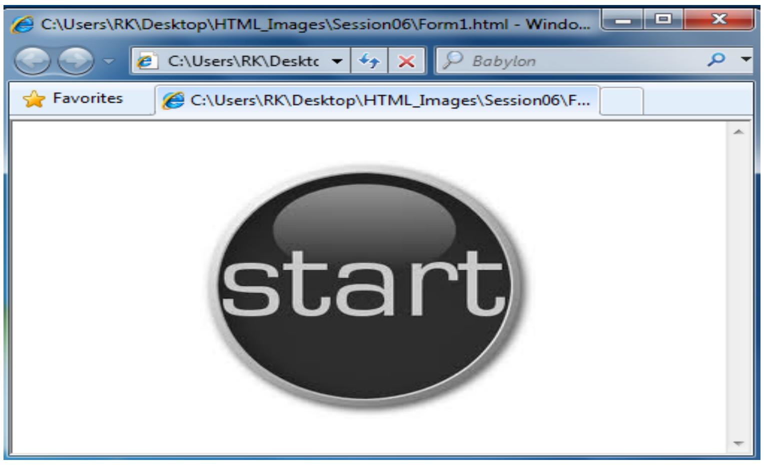
Output:


Firefox
file:///C:/Users/RK/...ession06/Form1.html
file:///C:/Users/RK/Desktop/
Delta Sea
Submit
Reset
Button

Example: Using an image to create a button.

```
<form >
    <input type = "image" name = "imagebutton" src = "URL" />
</form>
```

Output:

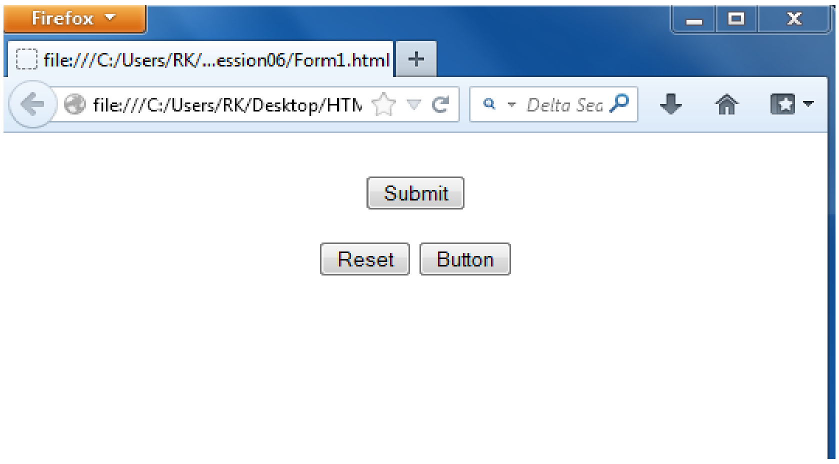

C:\Users\RK\Desktop\HTML_Images\Session06\Form1.html - Windo...
Favorites
start

Note:Here src attribiute specifies a location of the image on your webserver.

Example: Using $< button >$ element to create various buttons

```
<form>
    <button type = "submit">Submit</button>
    <br /><br />
    <button type = "reset"> Reset </button>
    <button type = "button"> Button </button>
</form>
```

Output:



Checkboxes Control

Checkbox buttons allows the user to select one or more options among a set of predefined choices. They are created using $< input >$ tag.

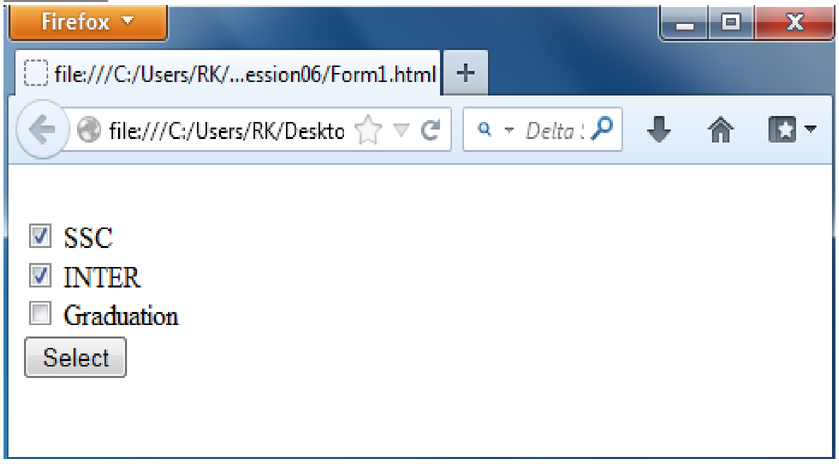
Following is the list of important checkbox attributes:

- **type:** Indicates that you want to create a checkbox.
- **name:** Name of the control.
- **value:** The value that will be used if the checkbox is selected. More than one checkbox should share the same name only if you want to allow users to select several items from the same list.
- **checked**: Indicates that when the page loads, the checkbox should be selected.

Example:

```
<form>
    <input type = "checkbox" name = "SSS" value = "on"> SSC
    <input type = "checkbox" name = "Inter" value = "on"> INTER
    <input type = "checkbox" name = "Graduation" value = "on"> Graduation
    <input type = "submit" value = "Select" />
</form>
```

Output:


Firefox
file:///C:/Users/RK/...ession06/Form1.html
file:///C:/Users/RK/Deskto
Delta

☑ SSC
☑ INTER
☐ Graduation
Select

Raido box Control

Radio Buttons are used when only one option is required to be selected. radio buttons can be grouped so that only a single member of the group can be selected at a time.

Radio buttons differ from check boxes in that only a single radio button in a given group can be selected at any one time. You indicate a group of radio buttons by providing all of them with the same NAME. Only one button in a group can be depressed at a time, selecting a new button when one is already selected results in the previous choice becoming deselected. The value of the one selected is sent when the form is submitted.

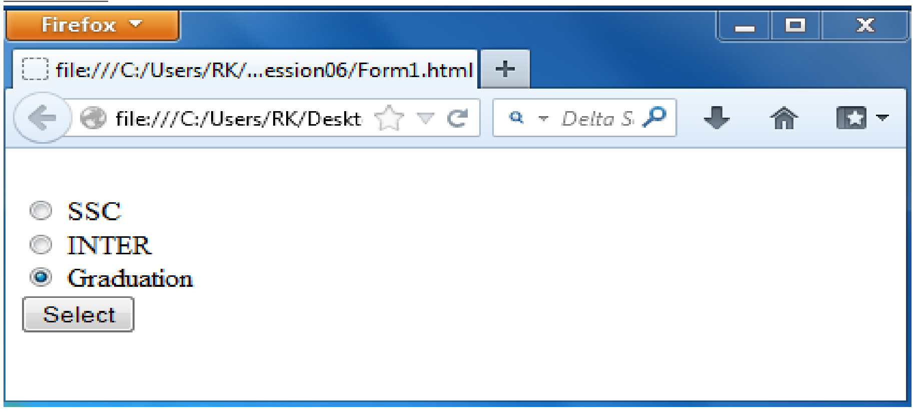
Following is the list of important radiobox attributes:

- **type:** Indicates that you want to create a radiobox.
- **name:** Name of the control.
- **value:** Used to indicate the value that will be sent to the server if this option is selected.
- **checked:** Indicates that this option should be selected by default when the page loads.

Example:

```
    <form>
        <input type = "radio" name = "Qualification" value = "SSC"> SSC
        <input type = "radio" name = "Qualification" value = "Inter"> INTER
        <input type = "radio" name = "Qualification" value = "Graduation">
                                                          Graduation
        <input type = "submit" value = "Select" />
    </form>
```

Output:


Firefox
file:///C:/Users/RK/...ession06/Form1.html
file:///C:/Users/RK/Deskt
Delta S
SSC
INTER
Graduation
Select

Select box Control

A SELECT element presents a set of options to the user. You specify each choice with an OPTION element enclosed between $< SELECT... >$ and $< /SELECT... >$.

Following is the list of important attributes of $< select >$:

- **name:** This is the name for the control.
- **size:**This can be used to present a scrolling list box.
- **multiple:** If set to "multiple" then allows a user to select multiple items from the menu.

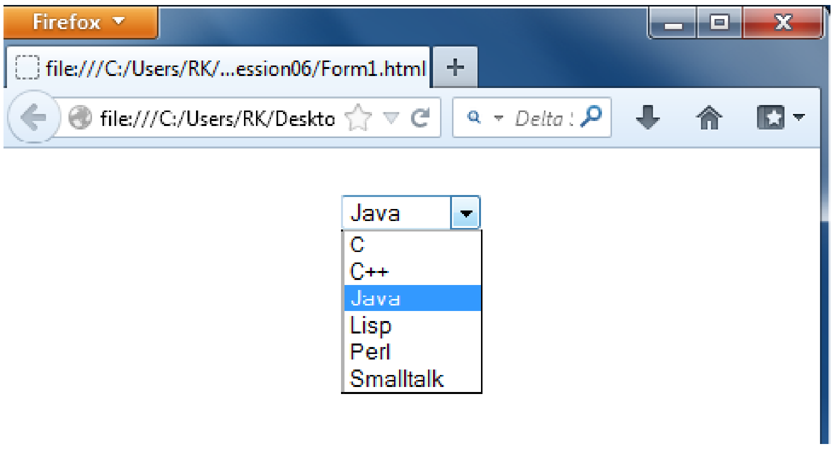
Following is the list of important attributes of $< option >$:

- **value:** The value that is sent to the server if this option is selected.
- **selected:** Specifies that this option should be the initially selected value when the page loads.
- **label:** An alternative way of labeling options.

Example: of a SELECT menu

```
<SELECT NAME = "language">
    <OPTION VALUE = "c">C
    <OPTION VALUE = "c++">C++
    <OPTION VALUE = "java" SELECTED>Java
    <OPTION VALUE = "lisp">Lisp
    <OPTION VALUE = "perl">Perl
    <OPTION VALUE = "smalltalk">Smalltalk
</SELECT>
```

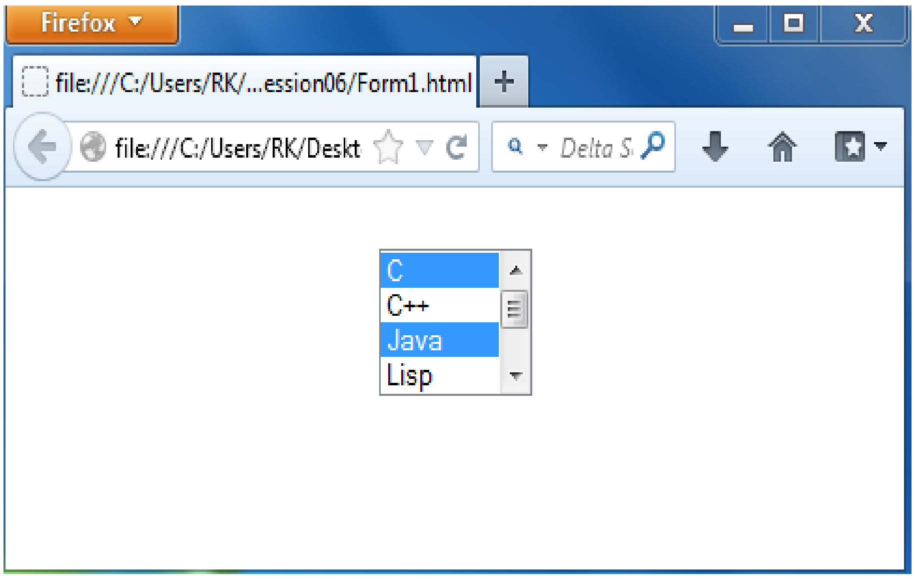
Output:


Firefox
file:///C:/Users/RK/...ession06/Form1.html
file:///C:/Users/RK/Deskto
Delta
Java
C
C++
Java
Lisp
Perl
Smalltalk

Example: of a SELECT menu that permits selection of multiple options

```
Languages you know:<BR>
<SELECT NAME = "language" MULTIPLE>
    <OPTION VALUE = "c">C
    <OPTION VALUE = "c++">C++
    <OPTION VALUE = "java" SELECTED>Java
    <OPTION VALUE = "lisp">Lisp
    <OPTION VALUE = "perl" SELECTED>Perl
    <OPTION VALUE = "smalltalk">Smalltalk
</SELECT>
```

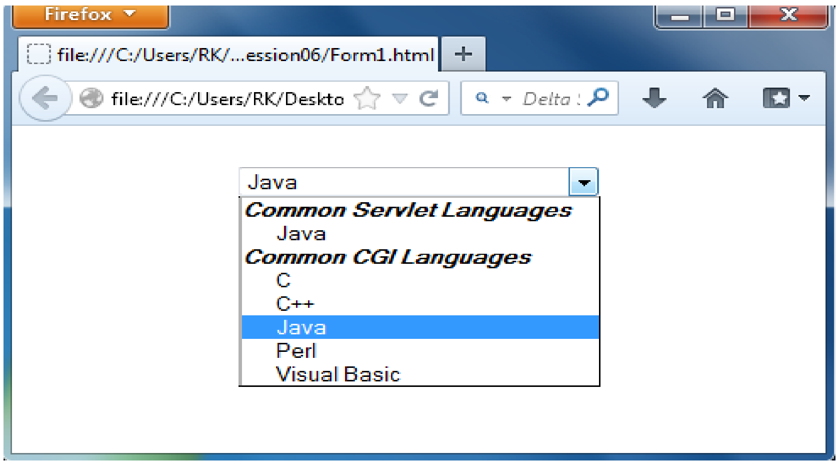
Output:


Firefox
file:///C:/Users/RK/...ession06/Form1.html
file:///C:/Users/RK/Deskt
Delta S
C
C++
Java
Lisp

Example: SELECT menu categorized into two groups with the OPTGROUP element

```
Server-side Languages:
<SELECT NAME = "language">
    <OPTGROUP LABEL = "Common Servlet Languages">
        <OPTION VALUE = "java1">Java
    </OPTGROUP>
    <OPTGROUP LABEL = "Common CGI Languages">
        <OPTION VALUE = "c">C
        <OPTION VALUE = "c++">C++
        <OPTION VALUE = "java2">Java
        <OPTION VALUE = "perl">Perl
        <OPTION VALUE = "vb">Visual Basic
    </OPTGROUP>
</SELECT>
```

Output:


Firefox
file:///C:/Users/RK/...ession06/Form1.html
file:///C:/Users/RK/Deskto
Delta

Java

Common Servlet Languages
Java
Common CGI Languages
C
C++
Java
Perl
Visual Basic

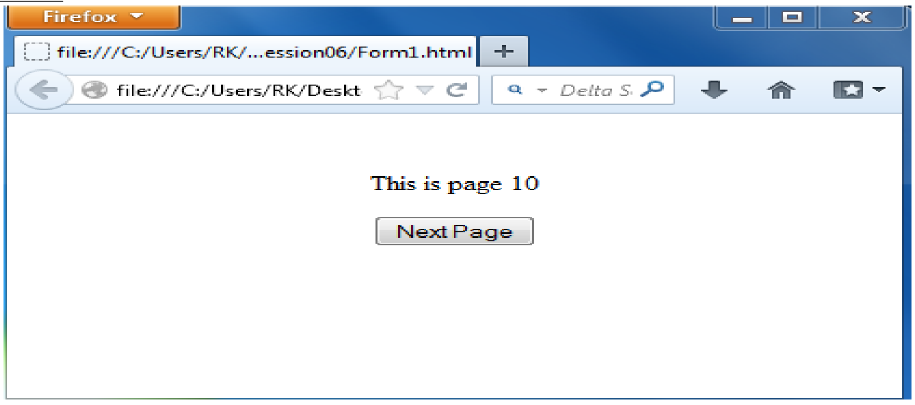
Hidden Controls

If you will want to pass information between pages without the user seeing it. Hidden form controls remain part of any form, but the user cannot see them in the Web browser. They should not be used for any sensitive information you do not want the user to see because the user could see this data if she looked in the source of the page.

Example:

```
<form method = "get" name = "pages">
    <p>This is page 10</p>
    <input type = "hidden" name = "pagenumber" value = "10" />
    <input type = "submit" value = "Next Page" />
</form>
```

Output:


Firefox
file:///C:/Users/RK/...ession06/Form1.html
file:///C:/Users/RK/Deskt
Delta S
Next Page

This is page 10

HTML5 New Input Types

HTML5 has several new input types for forms. These new features allow better input control and validation.

Tag	Description
$< datalist >$	Specifies a list of pre-defined options for an $< input >$ element
$< keygen >$	Specifies a key-pair generator field in a form
$< output >$	Represents the result of a calculation
$< color >$	Used for input fields that should contain a color.
$< date >$	Allows the user to select a date.
$< datetime >$	Allows the user to select a date and time (with time zone).
$< datetime - local >$	Allows the user to select a date and time (no time zone).
$< email >$	Defines an input fields that should contain an e-mail address.
$< month >$	Allows the user to select a month and year.
$< number >$	Defines an input fields that should contain a numeric value.
$< range >$	Defines an input fields that should contain a value from a range of numbers.
$< search >$	Used for search fields (a search field behaves like a regular text field).
$< tel >$	Defines an input fields that should contain a telephone number.
$< time >$	Defines a control for entering a time (no time zone)
$< url >$	Defines a field for entering a URL
$< week >$	Defines a week and year control (no time zone)

HTML5 New Attributes

HTML5 has provided new attributes for $< form >$ and $< input >$.

New $< form >$ Attributes

autocomplete - The autocomplete attribute specifies whether a form or input field should have autocomplete-on or off. When autocomplete is on, the browser automatically complete values based on values that the user has entered before. *Note: The autocomplete attribute works with* $< form >$ *and the following* $< input >$ *types: text, search, url, tel, email, password, datepickers, range, and color.*

novalidate -The novalidate attribute is a boolean attribute. When present, it specifies that the form-data (input) should not be validated when submitted.

New $< input >$ **Attributes**

autofocus - The autofocus attribute is a boolean attribute. When present, it specifies that an $< input >$ element should automatically get focus when the page loads.

form - The form attribute specifies one or more forms an $< input >$ element belongs to.

formmethod - The formmethod attribute overrides the method attribute of the $< form >$ element.

min and max - The min and max attributes specify the minimum and maximum value for an $< input >$ element.

placeholder - The placeholder attribute specifies a short hint that describes the expected value of an input field (e.g. a sample value or a short description of the expected format).

required - It specifies that an input field must be filled out before submitting the form.

Examples

Exampe-01 Sample code to play Video Clip

```
<DOCTYPE html>
    <html>
        <head>
            <title> HTML5 Video </title>
        </head>
        <body>
            <center>
                <video width="320" height="240" controls>
                    <source src="./media/movie1.mp4"
                                        type="video/mp4">
                    <source src="./media/movie1.ogg"
```

```
                                                type="video/ogg">
                                Your browser does not support the video tag.
                        </video>
                        <h1> <u>This is a video player.</u></h1>


                </center>
        </body>
    </html>
```

Example-02 Sample code to play Audio Clip

```
<DOCTYPE html>
    <html>
        <head>
            <title> HTML5 Audio</title>
        </head>
        <body>
            <center>

                <audio controls>
                    <source src="./media/audio1.ogg"
                                                type="audio/ogg">
                    <source src="./media/audio1.mp3"
                                                type="audio/mpeg">
                    Your browser does not support the audio element.
                </audio>
                <h1> <u>This is an audio player.</u></h1>

            </center>
        </body>
    </html>
```

New Elements in HTML5

Whats new?

HTML5 was created to make the coding process easier and more logical. The unique and impressive features HTML5 comes with are in the multimedia department. Many of the features it comes with have been created with the consideration that users should be able to run heavy content on low-powered devices. The syntactic features include the new $< video >$, $< audio >$ and $< canvas >$ elements, but also integration of vector graphics content (what we knew before as being the $< object >$ tags). This means that multimedia and graphic content on the web will be handled and executed easier and faster, without the need of plugins or APIs.

New Elements

Tag	Description
$< article >$	Defines an article in the document
$< aside >$	Defines content aside from the page content
$< audio >$	Defines sound, such as music or other audio streams.
$< video >$	Defines video, such as a movie clip or other video streams.
$< source >$	is used to specify multiple media resources for media elements, such as $< video >$ and $< audio >$.
$< bdi >$	Defines a part of text that might be formatted in a different direction from other text outside it
$< details >$	Defines additional details that the user can view or hide
$< dialog >$	Defines a dialog box or window
$< figcaption >$	Defines a caption for a $< figure >$ element
$< figure >$	Defines self-contained content, like illustrations, diagrams, photos, code listings, etc.

COPYRIGHT ©TALENTSPRINT, 2015. ALL RIGHTS RESERVED.

Tag	Description
$< footer >$	Defines a footer for the document or a section
$< header >$	Defines a header for the document or a section
$< main >$	Defines the main content of a document
$< mark >$	Defines marked or highlighted text
$< menuitem >$	Defines a command/menu item that the user can invoke from a popup menu
$< meter >$	Defines a scalar measurement within a known range
$< nav >$	Defines navigation links in the document
$< progress >$	Defines the progress of a task
$< rp >$	Defines what to show in browsers that do not support ruby annotations
$< rt >$	Defines an explanation/pronunciation of characters (for East Asian typography)
$< ruby >$	Defines a ruby annotation (for East Asian typography)
$< section >$	Defines a section in the document
$< summary >$	Defines a visible heading for a $< details >$ element
$< time >$	Defines a date/time
$< wbr >$	Defines a possible line-break

COPYRIGHT ©TALENTSPRINT, 2015. ALL RIGHTS RESERVED.