HTML Forms

HTML Forms are required when you want to collect some data from the site visitor. For example registration information: name, email address, credit card, etc.
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 in whatever way you like.
Form elements are like text fields, textarea fields, drop-down menus, radio buttons, checkboxes, etc. which are used to take information from the user.
A simple syntax of using <form> is as follows:

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

There are different types of form controls that you can use to collect data from a visitor to your site.

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

HTML Forms - Text Input Controls:

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.

Single-line text input controls:

Single-line text input controls are created using an <input> element whose type attribute has a value of text.
Here is a basic example of a single-line text input used to take first name and last name:

<form action="/cgi-bin/hello_get.cgi" method="get">
First name:
<input type="text" name="first_name" />
**
**
Last name:
<input type="text" name="last_name" />
<input type="submit" value="submit" />
</form>

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.

<form action="/cgi-bin/hello_get.cgi" method="get">
Login :
<input type="text" name="login" />
**
**
Password:
<input type="text" name="password" />
<input type="submit" value="submit" />
</form>

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 action="/cgi-bin/hello_get.cgi" method="get">
**Description :
**
<textarea rows="5" cols="50" name="description">
Enter description here...
</textarea>
<input type="submit" value="submit" />
</form>

Attributes are

- **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.

HTML Forms - Creating Button:

There are various ways in HTML to create clickable buttons. A clickable button is created using <input> tag.
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.

Here is the example:

```
<form action="http://www.example.com/test.asp" method="get">
<input type="submit" name="Submit" value="Submit" />
<br /><br />
<input type="reset" value="Reset" />
<input type="button" value="Button" />
</form>
<button type="button"> Button </button>
</form>
```

HTML Forms - Checkboxes Control:

Checkboxes are used when more than one option is required to be selected. They are created using <input> tag as shown below.
Here is example HTML code for a form with two checkboxes

```
<form action="/cgi-bin/checkbox.cgi" method="get">
<input type="checkbox" name="maths" value="on"> Maths
<input type="checkbox" name="physics" value="on"> Physics
<input type="submit" value="Select Subject" />
</form>
```

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.

HTML Forms - Raidobox Control:

Radio Buttons are used when only one option is required to be selected. They are created using <input> tag as shown below:
Here is example HTML code for a form with two radio button:

```
<form action="/cgi-bin/radiobutton.cgi" method="post">
<input type="radio" name="subject" value="maths" /> Maths
<input type="radio" name="subject" value="physics" /> Physics
<input type="submit" value="Select Subject" />
</form>
```

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.

HTML Forms - Select box Control:

Drop Down Box is used when we have many options available to be selected but only one or two will be selected..
Here is example HTML code for a form with one drop down box

```
<form action="/cgi-bin/dropdown.cgi" method="post">
<select name="dropdown">
<option value="Maths" selected>Maths</option>
<option value="Physics">Physics</option>
</select>
<input type="submit" value="Submit" />
```

</form>

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.

HTML Forms - File Select Boxes:

If you want to allow a user to upload a file to your web site from his computer, you will need to use a file upload box, also known as a file select box. This is also created using the <input> element.
Here is example HTML code for a form with one file select box

```
<form action="/cgi-bin/hello_get.cgi" method="post"
name="fileupload" enctype="multipart/form-data">
<input type="file" name="fileupload" accept="image/*" />
</form>
```

HTML Forms - 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.
Following hidden form is being used to keep current page number. When a user will click next page then the value of hidden form will be sent to the back-end application and it will decide which page has be displayed next.

```
<form action="/cgi-bin/hello_get.cgi" method="get" name="pages">
<p>This is page 10</p>
<input type="hidden" name="pgaenumber" value="10" />
<input type="submit" value="Next Page" />
</form>
```

HTML Forms - Submit and Reset Button

These are special buttons which can be created using <input> When submit button is clicked then Forms data is submitted to the back-end application. When reset button is clicked then all the forms control are reset to default state.
You already have seen submit button above, we will give one reset example here:

```
<form action="/cgi-bin/hello_get.cgi" method="get">
First name:
<input type="text" name="first_name" />
<br>
Last name:
<input type="text" name="last_name" />
<input type="submit" value="Submit" />
<input type="reset" value="Reset" />
</form>
```

HTML Forms OBJECT element

The purpose of the <object> element is to support HTML helpers (plug-ins).

HTML Helpers (Plug-ins)

A helper application is a small computer program that extends the standard functionality of the browser. Helper applications are also called plug-ins.

Plug-ins are often used by browsers to play audio and video.Examples of well-known plug-ins are Adobe Flash Player and QuickTime. Plug-ins can be added to Web pages through the <object> tag or the <embed> tag. Most plug-ins allow manual (or programmed) control over settings for volume, rewind, forward, pause, stop, and play.

<OBJECT> may be used when the author wishes to provide an alternative for user agents that don't support a particular media. A simple example of using OBJECT is:

```
<OBJECT data=TheEarth.avi type="application/avi" alt="The Earth">
<img src=TheEarth.gif alt="The Earth">
</OBJECT>
```

Here the user agent would show an animation if it supports the AVI format, otherwise it would show a GIF image. The IMG element is used for the latter as it provides for backwards compatibility with existing browsers. The TYPE attribute allows the user agent to quickly detect that it doesn't support a particular object, and avoid wasting time downloading it. Another motivation for using the TYPE attribute is when the object is loaded off a local drive, as it allows the format to be specified directly rather than being inferred from the file extension. The ALT attribute allows the user agent to provide an alternative to processing the OBJECT resource indicated by the DATA attribute.

The SPAN and DIV HTML Elements

The SPAN and DIV HTML tags are very useful for use with Cascading Style Sheets. Many beginners use the two elements in a similar fashion, but they serve different purposes. Learn how to use these two elements effectively in your web pages.

The DIV Element

The DIV element defines logical divisions on your web page. It acts a lot like a <P> element, by placing newlines or carriage returns before and after the division. A division can have multiple paragraphs in it.

Using the DIV Tag

To use the DIV element, surround the area of your page that you want as a separate division with the and tags:

```
<div>
<p>contents of div</p>
</div>
```

If the area is unique on the page, you can add anid, e.g. or if there are many similar areas onthe page, you can add a class, e.g. <divclass="bigDiv">. Both of these attributes can then be selectted using CSS or modified using JavaScript.

The DIV element allows you to define the style of entire sections of the HTML. You can define a division of your page as a callout and give that area a different style from the surrounding text. That area may have images, paragraphs, headlines, anything you wanted. The DIV element also gives you the ability to identify unique areas of your documents.

The DIV element is different from the HTML5 SECTION element because it does not give the enclosed content any semantic meaning. If you aren't sure whether the block of content should be a DIV or a SECTION, think about what that content's purpose is and why you need the DIV or SECTTION element.

- If you need the element simply to add styles to that area of the page, you should use theDIV element.
- If that area of the page has a specific semantic meaning or the contents can stand on its own like an article or or blog post, then you should use the SECTION element.

One thing to keep in mind when using the DIV element is that it breaks paragraphs. It acts as a paragraph end/beginning, and while you can have paragraphs within a DIV you can't have aDIV inside a paragraph.

The most important attributes of the DIV element are:

- style

- class
- id

Even if you don't use style sheets or DHTML, you should get into the habit of using the DIV element. This will give you more flexibility and future proof your HTML.

Because the CENTER element has been deprecated in HTML 4.0 and is obsolete in HTML5, it is a good idea to start using to center the text inside yourDIV elements instead.

The SPAN Element

The SPAN element has very similar properties to the DIV element, in that it, along with CSS, can change the style of the text it encloses. But without any style attributes, the SPAN element doesn't change the enclosed text at all.

The main difference between the SPAN and DIV elements is that SPAN doesn't do any formatting of it's own. As mentioned above, the DIV element includes a paragraph break. TheSPAN element simply tells the browser to apply the style rules to whatever is within the SPAN.

To use the SPAN element, simply surround the text that you want to add styles to with the and tags:

```
<div id="mydiv">
<p><span>Highlighted text</span> and non-highlighted text.</p>
</div>
```

Then add the class="highlight" or other class to the SPAN element to style the text with CSS. e.g. <style class="highlight">

The SPAN element has no required attributes, but the three that are the most useful are the same as for the DIV element:

- style
- class
- id

Use SPAN when you want to change the style of elements without placing them in a new block-level element in the document. For example, if you had a Level 3 Heading (H3) that you wanted the second word to be red, you could surround that word with 2ndWord and it would still be a part of the H3 tag, just red. For example:

<h3>This is My Awesome Headline</h3>

Embedding media (Video and Audio):

```
<video src="baby.mp4"  width="600" height="600" controls>
   Your browser does not support the <video> element.
</video>
```

```
<audio src="audio.ogg" controls autoplay loop>
<p>Your browser does not support the audio element </p>
</audio>
```

This code example uses attributes of the <audio> element:

- controls : Displays the standard HTML5 controls for the audio on the web page.
- autoplay : Makes the audio play automatically.
- loop : Make the audio repeat (loop) automatically.

```
<audio src="audio.mp3" preload="auto" controls></audio>
```

The preload attribute is used in the audio element for buffering large files. It can take one of 3 values:

- "none" does not buffer the file

- "auto" buffers the media file
- "metadata" buffers only the metadata for the file

Multiple source files can be specified using the <source> element in order to provide video or audio encoded in different formats for different browsers. For instance:

```
<video controls>
  <source src="foo.ogg" type="video/ogg">
  <source src="foo.mp4" type="video/mp4">
  Your browser does not support the <code>video</code> element.
</video>
```

Video Attribute Specification:

The HTML5 video tag can have a number of attributes to control the look and feel and various functionalities of the control:

Attribute	Description
autoplay	This boolean attribute if specified, the video will automatically begin to play back as soon as it can do so without stopping to finish loading the data.
autobuffer	This boolean attribute if specified, the video will automatically begin buffering even if it's not set to automatically play.
controls	If this attribute is present, it will allow the user to control video playback, including volume, seeking, and pause/resume playback.
height	This attribut specifies the height of the video's display area, in CSS pixels.
loop	This boolean attribute if specified, will allow video automatically seek back to the start after reaching at the end.
preload	This attribute specifies that the video will be loaded at page load, and ready to run. Ignored if autoplay is present.
poster	This is a URL of an image to show until the user plays or seeks.
src	The URL of the video to embed. This is optional; you may instead use the <source> element within the video block to specify the video to embed
width	This attribut specifies the width of the video's display area, in CSS pixels.
Attribute	**Description**
autoplay	This boolean attribute if specified, the audio will automatically begin to play back as soon as it can do so without stopping to finish loading the data.
autobuffer	This boolean attribute if specified, the audio will automatically begin buffering even if it's not set to automatically play.

controls	If this attribute is present, it will allow the user to control audio playback, including volume, seeking, and pause/resume playback.
loop	This boolean attribute if specified, will allow audio automatically seek back to the start after reaching at the end.
preload	This attribute specifies that the audio will be loaded at page load, and ready to run. Ignored if autoplay is present.
src	The URL of the audio to embed. This is optional; you may instead use the <source> element within the video block to specify the video to embed

Handling Media Events:

The HTML5 audio and video tag can have a number of attributes to control various functionalities of the control using Javascript:

Event	**Description**
abort	This event is generated when playback is aborted.
canplay	This event is generated when enough data is available that the media can be played.
ended	This event is generated when playback completes.
error	This event is generated when an error occurs.
loadeddata	This event is generated when the first frame of the media has finished loading.
loadstart	This event is generated when loading of the media begins.
pause	This event is generated when playback is paused.
play	This event is generated when playback starts or resumes.
progress	This event is generated periodically to inform the progress of the downloading the media.
ratechange	This event is generated when the playback speed changes.
seeked	This event is generated when a seek operation completes.
seeking	This event is generated when a seek operation begins.

suspend	This event is generated when loading of the media is suspended.
volumechange	This event is generated when the audio volume changes.
waiting	This event is generated when the requested operation (such as playback) is delayed pending the completion of another operation (such as a seek).