

# Basic HTML tags

# Structural Tags

`<HTML>`

These tags enclose the entire Web page document.

`</HTML>`

`<HEAD>`

These tags enclose the Head part of the document

`</HEAD>`

`<TITLE>`

These tags enclose the title of the document. This text appears in the title bar in the browser and on the bookmark list if someone bookmarks your web page.

`</TITLE>`

# Sample Structure of a Web Site

```
<HTML>  
  <HEAD>  
    <TITLE> SVMIT Bharuch</TITLE>  
  </HEAD>  
  
  <BODY>  
    This is SVMIT Webpage!  
  </BODY>  
</HTML>
```

# Header Tags

Header Tags -- Used for marking sections and subsections in a document.

<H1>Header 1 -- Giant-sized and bold </H1>

<H2>Header 2 -- Large and bold </H2>

<H3>Header 3 -- Normal-sized and bold </H3>

<H4>Header 4 -- Small and bold </H4>

<H5>Header 5 -- Very Small and bold </H5>

<H6>Header 6 -- Tiny and bold </H6>

# Header Tags (cont.)

**H1 = Giant-sized and bold**

**H2 = Large and bold**

**H3 = Normal-sized and bold**

**H4 = Small and bold**

**H5 = Very Small and bold**

**H6 = Tiny and bold**

# Breaking Lines and Paragraphs

- `<P> text </P>`
    - Paragraph tag
    - Most browsers render (process) this with blank lines between each paragraph
  - `<BR>`
    - Line break tag
    - Used when the webmaster wants a carriage return but doesn't want a blank line to follow
- 

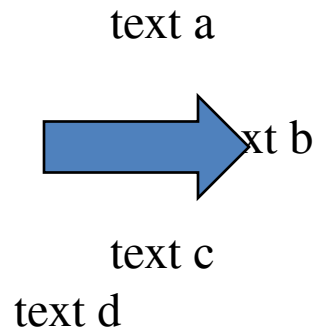
Example:

`<p>text a</p>`

`<p>text b</p>`

`<br>text c`

`<br>text d`



# Horizontal Rule

The <HR> tag puts a graphical line across the page.

Ex:



## *Horizontal Rule Attributes:*

NOSHADE -- A solid line with no shading

WIDTH="xx%/xx" -- Controls the width of the line. You may specify either percentage of the width of a page or actual pixel length

SIZE="xx" -- Controls the height of the line. You need to specify the dimension in pixels.

ALIGN="left/center/right" -- This allows the line to be aligned to the left, right, or center of the page

# Text Formatting Tags

Some basic text formatting styles:

Tag	Result
<I> Italics </I>	<i>Italics</i>
<B> Bold </B>	<b>Bold</b>
<PRE> Preformatted Text </PRE>	Preformatted Text
<STRONG> Strong </STRONG>	<b>Strong</b>
<ADDRESS> Address </ADDRESS>	<i>Address</i>
<CITE> Citations </CITE>	<i>Citations</i>
<CODE> Source Code </CODE>	Source Code



# Font modifications

Web creators can also change the way text looks by using the <FONT> tag

SIZE="number" - changes size of the font; 1=smallest, 7 = largest

<FONT SIZE="7">Big</FONT> <FONT SIZE="1">Small</FONT>

Big Small

---

COLOR="color-name" - changes text color

<FONT COLOR="red">This is red</FONT>

This is red

---

FACE="font-name" - changes font

<FONT FACE="verdana">This is the verdana font;</FONT> <FONT FACE="chicago">this is the chicago font.</FONT>

This is the verdana font; this is chicago font.

# <Font> modifications (cont.)


One can combine font modifications:

<FONT SIZE="7" FACE="courier" COLOR="red">Big, Courier & Red</FONT>

Big, Courier & Red

<FONT SIZE="7"><FONT FACE="courier">Big & Courier</FONT> - Just Big</FONT>

Big & Courier - Just Big



# Lists -- Unordered Lists

## Unordered lists:

<UL>

<LI>Item One

<LI>Item Two

<LI>Item Three

<LI>Item Four

</UL>



• Item One

• Item Two

• Item Three

• Item Four

---

## *Unordered List Attributes:*

type="disc/circle/square"

• Disc (default)     Circle    ■ Square

# Lists -- Ordered Lists

## Ordered (Numbered) Lists:

<OL>

<LI> Item One

<LI> Item Two

<LI> Item Three

<LI> Item Four

</OL>



1. Item One

2. Item Two

3. Item Three

4. Item Four

---

## Ordered List Attributes:

type="i/I/a/A/1"

(default)

i =	i. Item One	I =	I. Item One	a =	a. Item One	A =	A. Item One	1 =	1.Item One
	ii. Item Two		II. Item Two		b. Item Two		B. Item Two		2. Item Two
	iii. Item Three		III. Item Three		c. Item Three		C. Item Three		3. Item Three
	iv. Item Four		IV. Item Four		d. Item Four		D. Item Four		4. Item Four

start="xx"

# Lists -- Definition Lists

## Definition Lists:

```
<DL>  
  <DT>List Name One  
    <DD>This is where information about List Name One would go</DD>  
</DT>  
  <DT>List Name Two  
    <DD>This is where information about List Name Two would go</DD>  
</DT>  
</DL>
```



List Name One

This is where information about List Name One  
would go

List Name Two

This is where information about List Name Two  
would go

# Links

The anchor tag <A> is used to link one document to another or from one part of a document to another part of the same document.

Basic Links:

```
<A HREF="http://www.stanford.edu/">Stanford University</A>
```

Inter-document Links:

```
<A HREF="#spot">Point to 'spot' in this document</A>
```

Defining a point in a document:

```
<A NAME="spot">Spot</A>
```

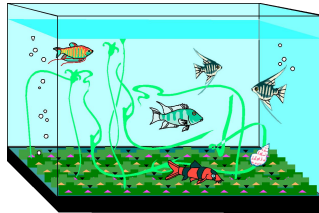
Email links:

```
<A HREF="mailto:someone@somehost.com">Email someone@somehost.com</A>
```

# Graphics

To have a graphic appear on a webpage, web designers must put the <IMG> tag in with the address where the graphic "lives":

```
<IMG SRC="http://www.someplace.com/images/fish.gif">
```



Graphics attributes:

alt="text": insert a description of the graphic for those who are using browsers that cannot process images (e.g., page readers for the blind)

width="xx/xx%": width in pixels/percentage

height="xx/xx%": height in pixels/percentage

border="xx": pixel length of the border surrounding the image.

hspace="xx": places a buffer of space horizontally around the image

vspace="xx": places a buffer of space vertically around the image

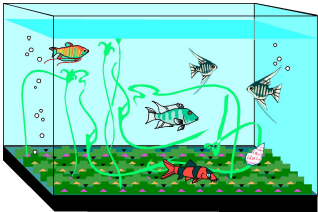
align="top/middle/bottom/right/left": aligns image in relation to the text (see

next

2 slides)

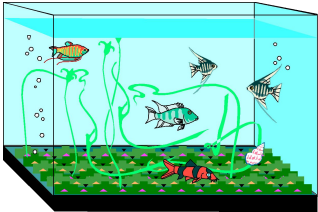
# Graphics (cont.)

``All about Fish



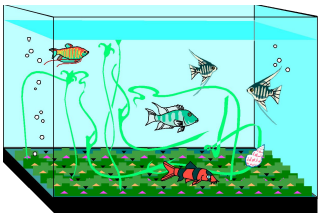
All about Fish

``All about Fish



All about Fish

``All about Fish



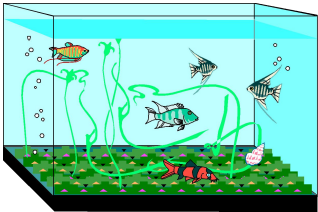
All about Fish



# Graphics (cont.)

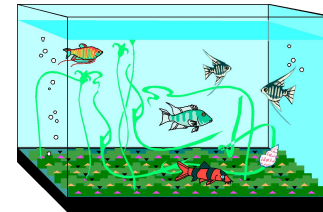
```

```



```

```



# HTML Tables

**<table>**

**<tr>**

**<th>**Company</th>

**<th>**Contact</th>

**<th>**Country</th>

</tr>

**<tr>**

**<td>**Google</td>

**<td>**Sundar Pichai</td>

**<td>**USA</td>

</tr>

**<tr>**

**<td>**Microsoft</td>

**<td>**Satya Nadella</td>

**<td>**USA</td>

</tr>

**</table>**

# Table: border, rowspan, colspan

```
<body>
<table border = "1">
  <tr>
    <th>Column 1</th>
    <th>Column 2</th>
    <th>Column 3</th>
  </tr>
  <tr>
    <td rowspan = "2">Row 1 Cell 1</td>
    <td>Row 1 Cell 2</td>
    <td>Row 1 Cell 3</td>
  </tr>
  <tr>
    <td>Row 2 Cell 2</td>
    <td>Row 2 Cell 3</td>
  </tr>
  <tr>
    <td colspan = "3">Row 3 Cell 1</td>
  </tr>
</table>
</body>
```

Column 1	Column 2	Column 3
Row 1 Cell 1	Row 1 Cell 2	Row 1 Cell 3
	Row 2 Cell 2	Row 2 Cell 3
Row 3 Cell 1		

OUTPUT ?

# ALL HTML Tags

<https://way2tutorial.com/html/tag/index.php>

# HTML Forms

# Parts of a Web Form

- A **Form** is an area that can contain **Form Control/Elements**.
- Each piece of information for a form is stored in a **Field**.
- The value itself is called **Field Value**.

# Parts of a Web Form

- Users enter or select a field using **Form Control/ Elements**.
  - Form/Control elements include: buttons, checkboxes, text fields, radio buttons, drop-down menus, etc
  - A form usually contains a **Submit** button to send the information in the form elements to the server

# Control Elements

- **Input Boxes** – for text and numerical entries
- **Option Buttons**, also called **Radio Buttons** – for selecting a single option from a predefined list
- **Selection Lists** – for long lists of options, usually appearing in a **Drop-Down List Box**
- **Check Boxes** – for specifying yes or no
- **Text Areas** – for extended entries that can include several lines of text



# HTML Forms

- The basic construction of a HTML form is this...
  - <form>** - begin a form
  - <input>** - ask for information in one of several different ways
  - <input>** - there can be as many input areas as you wish
  - </form>** - end a form HTML form

# Forms and Server-Based Programs

- Forms are used to collect information.
- The information is then sent back to the server.
- Information is stored and analyzed using a program on the server.
- By giving users access to programs that react to user input, the Web became a more dynamic environment where companies and users could interact.

# Forms and Server-Based Programs

- Server-Based programs provide:
  - Online databases containing customer information
  - Online catalogs for ordering and purchasing merchandise
  - Dynamic Web sites with content that is constantly modified and updated
  - Message boards for hosting online discussion forums

# Forms and Server-Based Programs

- Because these programs run on Web servers, rather than locally, users might not have permission to create or edit them. Instead, users will receive information about how to interact with the programs on the server.
- Several reason to restrict direct access:
  - When you run a server-based program, you are interacting directly with the server
    - Security risks (computer hackers)
    - Drain on system resources caused by large number of programs running simultaneously

# Forms and Server-Based Programs

- Server-Based Programs
  - Common Gateway Interface (CGI) Scripts
    - Most common
  - ASP
  - Cold Fusion
  - C/C++
  - PHP
  - VBScript
- The Web server determines which language your Web form will use.

# Getting Started

- The first step in creating a form is to specify the name and location of the CGI script that will be used to process the form data. To do this, type the following code within your HTML file, and note that there are no spaces:
- `<form METHOD="Post"  
ACTION=http://www.temple.edu/cgi-in/mail?your-e-mail-address@templ  
e.edu>`
- For example, if your e-mail address is jsmith@temple.edu, you would enter:
- `<form METHOD="Post"  
ACTION="http://www.temple.edu/cgi-bin/mail?jsmith@temple.edu">`

# Text Input (type="text")

- **A Text Field:**

- used to create one line fields that viewers can type text. The **default width is 20 characters**, and you can create fields of other sizes by the value in the size option. You can limit the number of characters by the value of the **MAXLENGTH option**. Text input fields will be empty when the page loads, unless you provide an initial text string for its VALUE option
- **<input type="text" name="textfield" size="value" value="with an initial value">**

# Text Input (type="text")

- Example 1: A text field named "text1" that is 30 characters wide.
- `<input type="text" name="text1" size="30">`
- Example 2: A text field named "text2" that is 30 characters wide but will only accept 20 characters.
- `<input type="text" name="text2" size="30" maxlength="20">`
- Example 3: A text field named "text3" that is 40 characters wide with default value.
- `<input type="text" name="text3" size="40" value="We are not alone">`



# Password Input (type="password")

- are exactly the same as text input elements, except that when the viewer types in, they see "bullet" characters rather than the letter they are typing. Password text is scrambled during transmission and then unscrambled when the form data is received at the server end.
- **Example 4: A password field named "pass1" that is 30 characters wide**
- `<input type="password" name="pass1" size="30">`
- **Example 5: A password field named "pass2" that is 30 characters wide but will only accept 20 characters**
- `<input type="password" name="pass2" size="30" maxlength="20">`

# Text Input (type="textarea")


- Text fields that have more than one line and can scroll as the viewer enters more text. The tag options define the size of the field by the number of rows and character columns. By adding the option **WRAP=VIRTUAL**, the text entered will automatically wrap at the right hand side of the field. You can also include default text to appear in the field
- **Example 6: A `textarea` field named "comments" that is 45 characters wide and 6 lines high**
- **`<textarea name="comments" rows="6" cols="45" wrap="virtual">`  
The first time I ever saw a web page, I thought.... (continue)  
`</textarea>`**


# Adding Control Buttons


- A form must include at least one control button for submitting the information once it is filled out. In addition, forms often include a button for resetting all the entries if a person wants to start over.
- When a person presses the submit button, he or she will receive confirmation that the form results were sent to your e-mail address. You will then see an e-mail message in your Inbox with the subject *FORM results*.

# Adding Control Buttons

- A submit button:  
`<input type="submit" name="Submit" value="Submit">`
- A reset button:  
`<input type="reset" name="Submit2" value="Reset">`

A submit button: 

A reset button: 

A plain button: 

- **submit**: send data
- **reset**: restore all form elements to their initial state

- Note that the type is **input**, not “button”

# Radio buttons (type="radio")

- Are sets of controls that are linked so that only one radio button among each set is selected at a time
- If you click one radio button, the others in the set are automatically de-selected
- A set of radio buttons is defined by providing them the same name
- The value sent in the web form is the value of the radio button that was last selected
- Adding the option **CHECKED** to one of the buttons in a set will make that button highlighted when the page loads
- Radio buttons do not contain any text

# Radio buttons (type="radio")

Radio buttons:<br>

```
<input type="radio" name="radiobutton" value="myValue1">  
male<br>
```

```
<input type="radio" name="radiobutton" value="myValue2" checked>  
female
```

Radio buttons:

☐ male

☒ female

# Checkboxes (type="checkbox")

- Are similar to radio buttons, but are not affected by other buttons, so you can have more than one in a group checked at a time
- Note that every checkbox has a unique name. If there is no check in the box, clicking it will place an X or a check mark there
- If the box is checked, clicking it again will remove the mark. The value sent in the web form is the value of the checkbox if it was selected; otherwise the value will be empty
- Adding the option **CHECKED** to a checkbox will make that checkbox highlighted when the page loads.

# Checkboxes (type="checkbox")

- A checkbox:  
`<input type="checkbox" name="checkbox"  
value="checkbox" checked>`

A checkbox: ☒

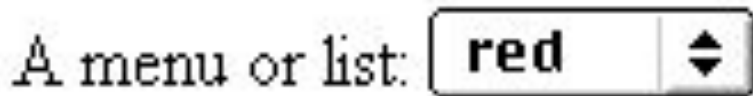
- **type:** "checkbox"
- **name:** used to reference this form element from JavaScript
- **value:** value to be returned when element is checked
- Note that there is *no text* associated with the checkbox—you have to supply text in the surrounding HTML



# Drop-down menu or list

- A menu or list:

```
<select name="select">  
  <option value="red">red</option>  
  <option value="green">green</option>  
  <option value="blue">blue</option>  
</select>
```



- Additional arguments:
  - **size**: the number of items visible in the list (default is "1")
  - **multiple**: if set to "true", any number of items may be selected (default is "false")

# Practice Exercise

**Who are you?**

Name:

Gender: ☐ Male ☐ Female

# A complete example

```
<html>
<head>
<title>Get Identity</title>
</head>
<body>
<p><b>Who are you?</b></p>
<form method="post" action="">
  <p>Name:
    <input type="text" name="textfield">
  </p>
  <p>Gender:
    <input type="radio" name="gender" value="m">Male
    <input type="radio" name="gender" value="f">Female</p>
</form>
</body>
</html>
```

**Who are you?**

Name:

Gender: ☐ Male ☐ Female

# GET Method

```
1.  <?php
2.      if( $_GET["name"] || $_GET["age"] ) {
3.          echo "Welcome ". $_GET['name']. "<br />";
4.          echo "You are ". $_GET['age']. " years old.";
5.          exit();
6.      }
7.  ?>

8.  <html>
9.      <body>
10.         <form action = "<?php $_PHP_SELF ?>" method = "GET">
11.             Name: <input type = "text" name = "name" />
12.             Age: <input type = "text" name = "age" />
13.             <input type = "submit" />
14.         </form>
15.     </body>
16. </html>
```

# POST Method

```
1.  <?php
2.      if( $_POST["name"] || $_POST["age"] ) {
3.          echo "Welcome ". $_POST['name']. "<br />";
4.          echo "You are ". $_POST['age']. " years old.";
5.          exit();
6.      }
7.  ?>

8.  <html>
9.      <body>
10.         <form action = "<?php $_PHP_SELF ?>" method = "POST">
11.             Name: <input type = "text" name = "name" />
12.             Age: <input type = "text" name = "age" />
13.             <input type = "submit" />
14.         </form>
15.     </body>
16. </html>
```

# GET and POST method

The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the ? character.

- The GET method produces a long string that appears in your server logs, in the browser's Location: box.
- The GET method is restricted to send upto 1024 characters only.
- Never use GET method if you have password or other sensitive information to be sent to the server.
- GET can't be used to send binary data, like images or word documents, to the server.
- The data sent by GET method can be accessed using QUERY\_STRING environment variable.
- The PHP provides \$\_GET associative array to access all the sent information using GET method.

The POST method transfers information via HTTP headers. The information is encoded as described in case of GET method and put into a header called QUERY\_STRING.

- The POST method does not have any restriction on data size to be sent.
- The POST method can be used to send ASCII as well as binary data.
- The data sent by POST method goes through HTTP header so security depends on HTTP protocol. By using Secure HTTP you can make sure that your information is secure.
- The PHP provides \$\_POST associative array to access all the sent information using POST method.

# HTML Frames

# Frames

- HTML frames are used to divide your browser window into multiple sections
- Each section can load a separate HTML document.
- A collection of frames in the browser window is known as a frameset.
- The window is divided into frames in a similar way the tables are organized: into rows and columns.



# Disadvantages of Frames

- Some smaller devices cannot cope with frames - size
- Sometimes your page will be displayed differently - resolution
- The browser's *back* button might not work
- There are still few browsers that do not support

# <frameset> Attributes

- cols : specify how many columns
- rows : specify how many rows
- border : the border of each frame
- frameborder :
- framespacing :

# <frame> Attributes

- src : Source
- Name : location name
- frameborder : overrides frameset, frameborder="0"
- marginwidth: marginwidth = "100"
- marginheight: marginheight = "100"
- noresize : noresize = "noresize"
- scrolling : scrolling = "yes"
- longdesc: longdesc = "framedescription.html"

# navigation bars

- Test.html
- Menu.html
- Main.html