|  |
| --- |
| **19CT503 INTERNET PROGRAMMING** |
| **UNIT I WEBSITE BASICS, HTML 5, CSS 3, WEB 2.0 9** |
| *Web Essentials: Clients, Servers and Communication – The Internet – Basic Internet protocols – World wide web – HTTP Request Message – HTTP Response Message – Web Clients – Web Servers – HTML5*  *– Tables – Lists – Image – HTML5 control elements – Semantic elements – Drag and Drop – Audio – Video controls*  *- CSS3 – Inline, embedded and external style sheets – Rule cascading – Inheritance – Backgrounds – Border Images – Colors – Shadows – Text – Transformations – Transitions – Animations.* |

WEB ESSENTIALS

* 1. **Web Essentials:**

***Server:***

The software that distributes the information and the machine where the information and software reside is called the server.

* provides requested service to client
* e.g., Web server sends requested Web page

***Client:***

The software that resides on the remote machine, communicates with the server, fetches the information, processes it, and then displays it on the remote machine is called the client.

* initiates contact with server (―speaks first‖)
* typically requests service from server
* Web: client implemented in browser

***Web server:***

Software that delivers Web pages and other documents to browsers using the HTTP protocol

***Web Page:***

A web page is a document or resource of information that is suitable for the World Wide Web and can be accessed through a web browser.

***Website:***

A collection of pages on the World Wide Web that are accessible from the same URL and typically residing on the same server.

***URL:***

Uniform Resource Locator, the unique address which identifies a resource on the Internet for routing purposes.

* 1. **Client-server paradigm:**

IThe Client-Server paradigm is the most prevalent model for distributed computing protocols. It is

the basis of all distributed computing paradigms at a higher level of abstraction. It is service-oriented, and employs a request-response protocol.

A server process, running on a server host, provides access to a service. A client process, running on a client host, accesses the service via the server process.The interaction of the process proceeds according to a protocol.

The primary idea of a client/server system is that you have a central repository of

information—some kind of data, often in a database—that you want to distribute on demand to some set of people or machines.

* 1. **The Internet:**
* Medium for communication and interaction in inter connected network.
* Makes information constantly and instantly available to anyone with a connection.

**Web Browsers:**

* User agent for Web is called a browser: o Internet Explorer
* Firefox

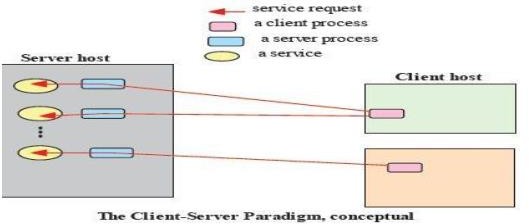
**Web Server:**

* Server for Web is called Web server:
* Apache (public domain)
* MS Internet Information Server

**Protocol:**

Protocols are agreed formats for transmitting data between devices. The protocol determines:

1. The error checking required
2. Data compression method used
3. The way the end of a message is signalled
4. The way the device indicates that it has received the message



* 1. **Internet Protocol:**

There are many protocols used by the Internet and the WWW:

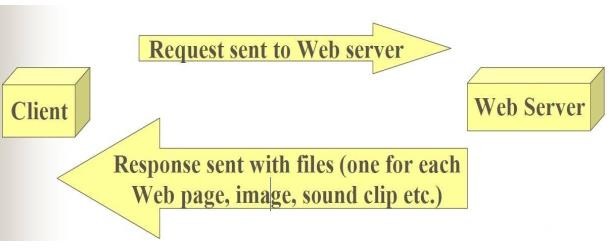
* TCP/IP
* HTTP
* FTP
* Electronic mail protocols IMAP
* POP

**TCP/IP**

The Internet uses two main protocols (developed by Vincent Cerf and Robert Kahn) Transmission control protocol (TCP):Controls disassembly of message into packets at the origin reassembles at the destination

Internet protocol (IP):Specifies the addressing details for each packet Each packet is labelled with its origin and destination.

* 1. **Hypertext Transfer Protocol (HTTP)**
* The hypertext transfer protocol (HTTP) was developed by Tim Berners-Lee in 1991
* HTTP was designed to transfer pages between machines
* The client (or Web browser) makes a request for a given page and the Server is responsible for finding it and returning it to the client
* The browser connects and requests a page from the server
* The server reads the page from the file system, sends it to the client and terminates the connection.



**Electronic Mail Protocols:**

* Electronic mail uses the client/server model
* The organisation has an email server devoted to handling email o Stores and forwards email messages
* Individuals use email client software to read and send email o (e.g. Microsoft Outlook, or Netscape Messenger)
* Simple Mail Transfer Protocol (SMTP)
* Specifies format of mail messages
* Post Office Protocol (POP) tells the email server to:
* Send mail to the user’s computer and delete it from the server
* Send mail to the user’s computer and do not delete it from the server
* Ask whether new mail has arrived.
  1. **Interactive Mail Access Protocol (IMAP)**

Newer than POP, provides similar functions with additional features.

o e.g. can send specific messages to the client rather than all the messages. A user can view email message headers and the sender’s name before downloading the entire message.

Allows users to delete and search mailboxes held on the email server.

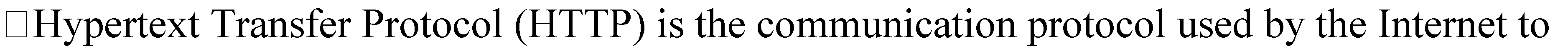
**The disadvantage of POP:** You can only access messages from one PC.

**The disadvantage of IMAP :**Since email is stored on the email server, there is a need for more and more expensive (high speed) storage space.

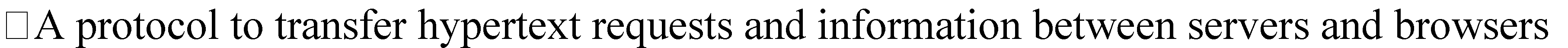
* 1. **World Wide Web:** comprises software (Web server and browser) and data (Web sites).

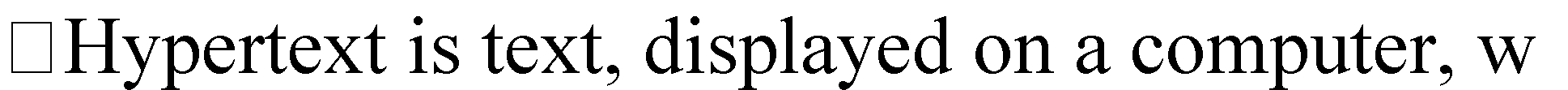
**Internet Protocol (IP) Addresses:**

* Every node has a unique numeric address
* Form: 32-bit binary number
* New standard, IPv6, has 128 bits (1998)
* Organizations are assigned groups of IPs for their computers
* **Domain names**
* Form: host-name. domain-names
* First domain is the smallest (Google)
* Last domain specifies the type of organization (.com)
* Fully qualified domain name - the host name and all of the domain names
* DNS servers - convert fully qualified domain names to IPs
  1. **HTTP:**



transfer hypertext documents.



* ith references (hyperlinks) to

other text that the reader can immediately follow, usually by a mouse HTTP is behind every request for a web document or graph, every click of a hypertext link, and every submission of a form.





**request** data, and how servers **respond** to these requests.



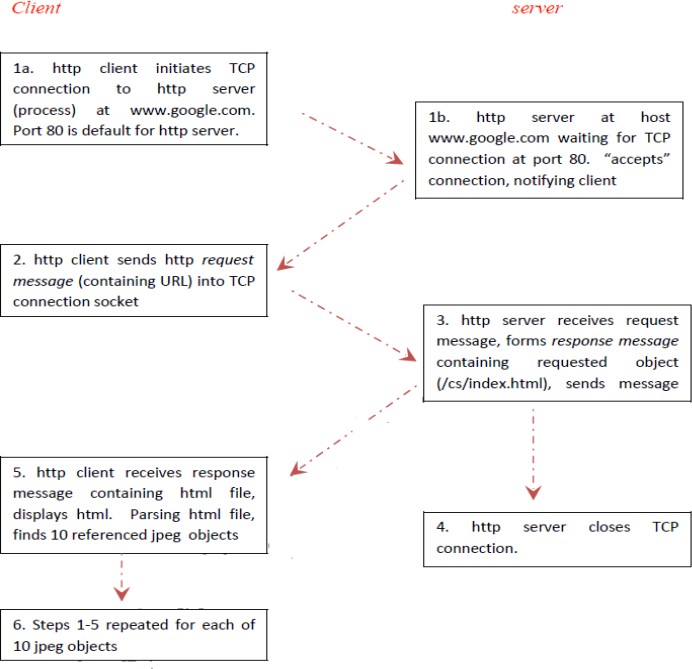
returning it to the client.





* the page from the file system and sends it to the client and then terminates the connection

***HTTP Transactions***

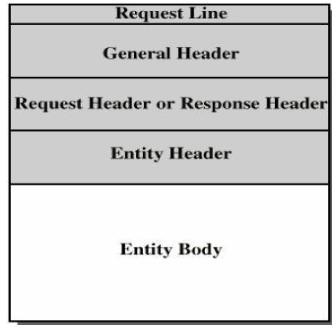


* 1. **HTTP Message:**

HTTP message is the information transaction between the client and server.

**Two types of HTTP Message:**

1. Requests
   1. Client to server
2. Responses
   1. Server to client



**Fields**

* Request line or Response line
* General header
* Request header or Response header
* Entity header
* Entity body

1. **Request Message:**

**Request Line:**

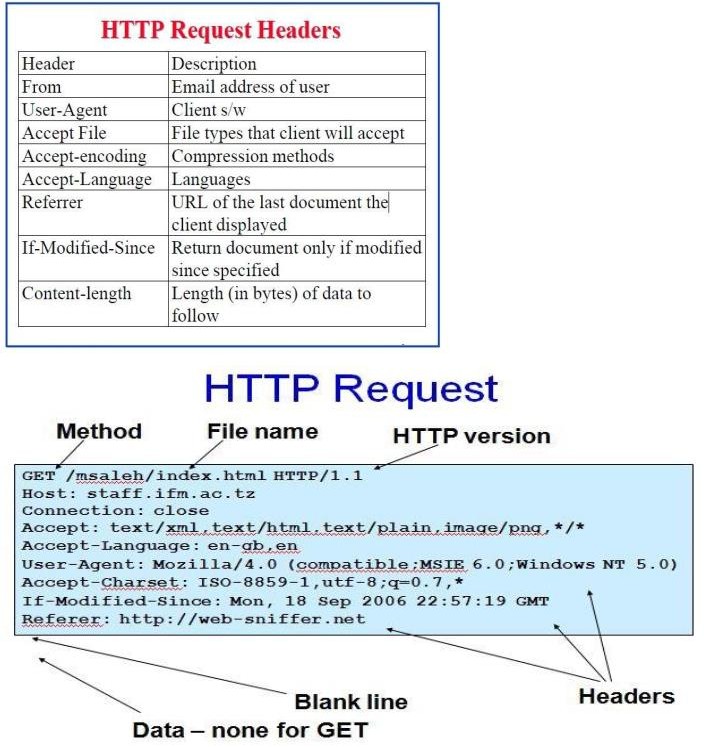
* A request line has three parts, separated by spaces o a *method* name

o the local path of the requested resource

o the version of HTTP being used

* A typical request line is:
* GET /path/to/file/index.html HTTP/1.1
* Notes:
* **GET** is the most common HTTP method; it says "give me this resource". Other methods include **POST** and **HEAD.** Method names are always uppercase
* The path is the part of the URL after the host name, also called the *request URI*
* The HTTP version always takes the form "**HTTP/x.x**", uppercase.

**Request Header:**



1. **Response Message:**

**Response Line:**

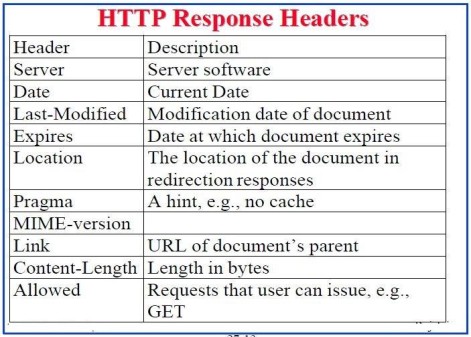
* A request line has three parts, separated by spaces o the HTTP version,

o a *response status code* that gives the result of the request, and

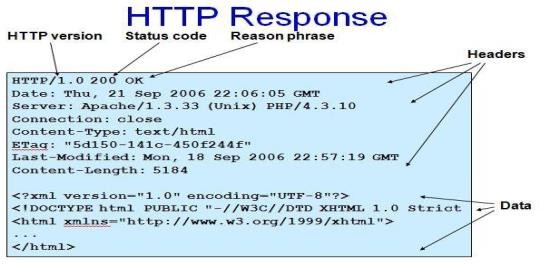
o an English *reason phrase* describing the status code

* Typical status lines are:
* HTTP/1.0 200 OK or
* HTTP/1.0 404 Not Found
* Notes:
* The HTTP version is in the same format as in the request line, "**HTTP/x.x**".
* The status code is meant to be computer-readable; the reason phrase is meant to be human-readable, and may vary.

**HTTP Request Header:**



**EXAMPLE**



**HTTP Method:**

* HTTP method is supplied in the request line and specifies the operation that the client has requested.

**Some common methods:**

* Options
* Get
* Head
* Post
* Put
* Move
* Delete

**Two methods that are mostly used are the GET and POST:**

* **GET** for queries that can be safely repeated
* **POST** for operations that may have side effects (e.g. ordering a book from an on-line store).

**The GET Method**

* It is used to retrieve information from a specified URI and is assumed to be a safe, repeatable operation by browsers, caches and other HTTP aware components
* Operations have no side effects and GET requests can be re-issued.
* For example, displaying the balance of a bank account has no effect on the account and can be safely repeated.
* Most browsers will allow a user to refresh a page that resulted from a **GET**, without displaying any kind of warning
* Proxies may automatically retry **GET** requests if they encounter a temporary network connection problem.
* GET requests is that they can only supply data in the form of parameters encoded in the URI (known as a **Query String**) – [downside]
* Cannot be unused for uploading files or other operations that require large amounts of data to be sent to the server.

**The POST Method**

* Used for operations that have side effects and cannot be safely repeated.
* For example, transferring money from one bank account to another has side effects and should not be repeated without explicit approval by the user.
* If you try to refresh a page in Internet Explorer that resulted from a **POST**, it displays the following message to warn you that there may



The **POST** request message has a content body that is normally used to send parameters and data

* The IIS server returns two status codes in its response for a **POST** request
* The first is **100 Continue** to indicate that it has successfully received the **POST** request
* The second is **200 OK** after the request has been processed.

**HTTP response status codes**

* Informational (1xx)
* Successful (2xx)
* Redirection (3xx)
* 301: moved permanently
* Client error (4xx)
* 403 : forbidden
* 404: Not found
* Server error (5xx)
* 503: Service unavailable
* 505: HTTP version not supported

**1.12 HTTP**

* **Features**
* Persistent TCP Connections: Remain open for multiple requests
* Partial Document Transfers: Clients can specify start and stop positions
* Conditional Fetch: Several additional conditions
* Better content negotiation
* More flexible authentication.
* **Web Browsers :**
* Mosaic - NCSA (Univ. of Illinois), in early 1993 First to use a GUI, led to Explosion of Web use Initially for X-Windows, under UNIX, but was ported to other platforms by late 1993
* Browsers are clients - always initiate, servers react (although sometimes servers require responses)
* Most requests are for existing documents, using Hypertext Transfer Protocol
* (HTTP) But some requests are for program execution, with the output being returned as a document.

Browser: A web browser is a software application for retrieving, presenting, and traversing information resources on the World Wide Web.

* **Web Servers:**
* Provide responses to browser requests, either existing documents or dynamicallyBuilt documents.
* Browser-server connection is now maintained through more than one request- Response cycle
* All communications between browsers and servers use Hypertext Transfer Protocol
* Web servers run as background processes in the operating system.
* Monitor a communications port on the host, accepting HTTP messages when they appear All current Web servers came from either

1. The original from CERN
2. The second one, from NCSA

* Web servers have two main directories:

1. Document root (servable documents)
2. Server root (server system software)

* Document root is accessed indirectly by clients
* Its actual location is set by the server Configuration file
* Requests are mapped to the actual location
* Virtual document trees
* Virtual hosts
* Proxy servers
* Web servers now support other Internet protocols
* Apache (open source, fast, reliable)
* IIS
* Maintained through a program with a GUI interface.

HTML 5

HTML is the main markup language for describing the structure of web pages.

HTML stands for HyperText Markup Language. HTML is the basic building block of World Wide Web.

Hypertext is text displayed on a computer or other electronic device with references to other text that the user can immediately access, usually by a mouse click or key press.

Apart from text, hypertext may contain tables, lists, forms, images, and other presentational elements. It is an easy-to-use and flexible format to share information over the Internet.

Markup languages use sets of markup tags to characterize text elements within a document, which gives instructions to the web browsers on how the document should appear.

HTML was originally developed by Tim Berners-Lee in 1990. He is also known as the father of the web. In 1996, the World Wide Web Consortium (W3C) became the authority to maintain the HTML specifications. HTML also became an international standard (ISO) in 2000. HTML5 is the latest version of HTML. HTML5 provides a faster and more robust approach to web development.

## HTML Tags and Elements

HTML is written in the form of HTML elements consisting of markup tags. These markup tags are the fundamental characteristic of HTML. Every markup tag is composed of a keyword, surrounded by angle brackets, such as <html>, <head>, <body>, <title>, <p>, and so on.

HTML tags normally come in pairs like <html> and </html>. The first tag in a pair is often called the opening tag (or start tag), and the second tag is called the closing tag (or end tag).

An opening tag and a closing tag are identical, except a slash (/) after the opening angle bracket of the closing tag, to tell the browser that the command has been completed.

## Inserting Images into Web Pages

Images enhance visual appearance of the web pages by making them more interesting and colorful.

The <img> tag is used to insert images in the HTML documents. It is an empty element and contains attributes only. The syntax of the <img> tag can be given with:

<img src="*url*" alt="*some\_text*">

The following example inserts three images on the web page:

***Example***

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=html&file=images)

<img src="kites.jpg" alt="Flying Kites">

<img src="sky.jpg" alt="Cloudy Sky">

<img src="balloons.jpg" alt="Balloons">

Each image must carry at least two attributes: the src attribute, and an alt attribute.

The src attribute tells the browser where to find the image. Its value is the URL of the image file.

Whereas, the alt attribute provides an alternative text for the image, if it is unavailable or cannot be displayed for some reason. Its value should be a meaningful substitute for the image.

**HTML Tables**

##### Creating Tables in HTML

HTML table allows you to arrange data into rows and columns. They are commonly used to display tabular data like product listings, customer's details, financial reports, and so on.

You can create a table using the <table> element. Inside the <table> element, you can use

the <tr> elements to create rows, and to create columns inside a row you can use the <td> elements. You can also define a cell as a header for a group of table cells using the <th> element.

The following example demonstrates the most basic structure of a table.

***Example***

<table>

<tr>

<th>No.</th>

<th>Name</th>

<th>Age</th>

</tr>

<tr>

<td>1</td>

<td>Peter Parker</td>

<td>16</td>

</tr>

<tr>

<td>2</td>

<td>Clark Kent</td>

<td>34</td>

</tr>

</table>

Tables do not have any borders by default. You can use the CSS [border](https://www.tutorialrepublic.com/css-reference/css-border-property.php) property to add borders to the tables. Also, table cells are sized just large enough to fit the contents by default. To add more space around the content in the table cells you can use the CSS [padding](https://www.tutorialrepublic.com/css-reference/css-padding-property.php) property.

## Defining a Table Header, Body, and Footer

HTML provides a series of tags [<thead>](https://www.tutorialrepublic.com/html-reference/html-thead-tag.php), [<tbody>](https://www.tutorialrepublic.com/html-reference/html-tbody-tag.php), and [<tfoot>](https://www.tutorialrepublic.com/html-reference/html-tfoot-tag.php) that helps you to create more structured table, by defining header, body and footer regions, respectively.

The following example demonstrates the use of these elements.

***Example***

<table>

<thead>

<tr>

<th>Items</th>

<th>Expenditure</th>

</tr>

</thead>

<tbody>

<tr>

<td>Stationary</td>

<td>2,000</td>

</tr>

<tr>

<td>Furniture</td>

<td>10,000</td>

</tr>

</tbody>

<tfoot>

<tr>

<th>Total</th>

<td>12,000</td>

</tr>

</tfoot>

</table>

**HTML Lists**

HTML lists are used to present list of information in well formed and semantic way. There are three different types of list in HTML and each one has a specific purpose and meaning.

* **Unordered list** — Used to create a list of related items, in no particular order.
* **Ordered list** — Used to create a list of related items, in a specific order.
* **Description list** — Used to create a list of terms and their descriptions.

**HTML Unordered Lists**

An unordered list created using the <ul> element, and each list item starts with the <li> element. The list items in unordered lists are marked with bullets. Here's an example:

***Example***

<ul>

<li>Chocolate Cake</li>

<li>Black Forest Cake</li>

<li>Pineapple Cake</li>

</ul>

— The output of the above example will look something like this:

* Chocolate Cake
* Black Forest Cake
* Pineapple Cake

You can also change the bullet type in your unordered list using the CSS [list-style-type](https://www.tutorialrepublic.com/css-reference/css-list-style-type-property.php) property. The following style rule changes the type of bullet from the default *disc* to *square*:

***Example***

ul {

list-style-type: square;

}

Please check out the tutorial on [CSS lists](https://www.tutorialrepublic.com/css-tutorial/css-lists.php) to learn about styling HTML lists in details.

##### HTML Ordered Lists

An ordered list created using the <ol> element, and each list item starts with the <li> element. Ordered lists are used when the order of the list's items is important.

The list items in an ordered list are marked with numbers. Here's an example:

***Example***

<ol>

<li>Fasten your seatbelt</li>

<li>Starts the car's engine</li>

<li>Look around and go</li>

</ol>

— The output of the above example will look something like this:

1. Fasten your seatbelt
2. Starts the car's engine
3. Look around and go

**HTML5 Image**

HTML Images Syntax

In HTML, images are defined with the <img> tag.

The <img> tag is empty, it contains attributes only, and does not have a closing tag.

The src attribute specifies the URL (web address) of the image:

<img src="*url*">

**EXAMPLE**

<!DOCTYPE html>

<html>

<body>

<h2>HTML Image</h2>

<img src="html5.gif" alt="HTML5 Icon" style="width:128px;height:128px;">

</body>

</body>

</html>

**OUTPUT**

### HTML Image



**HTML Form**

HTML Forms are required to collect different kinds of user inputs, such as contact details like name, email address, phone numbers, or details like credit card information, etc.

Forms contain special elements called **controls** like ***inputbox, checkboxes, radio-buttons, submit buttons***, etc. Users generally complete a form by modifying its controls e.g. entering text, selecting items, etc. and submitting this form to a web server for further processing.

The [<form>](https://www.tutorialrepublic.com/html-reference/html-form-tag.php) tag is used to create an HTML form. Here's a simple example of a login form:

***Example***

<form>

<label>Username: <input type="text"></label>

<label>Password: <input type="password"></label>

<input type="submit" value="Submit">

</form>

The following section describes different types of controls that you can use in your form.

**Input Element**

This is the most commonly used element within HTML forms.

It allows you to specify various types of user input fields, depending on the type attribute. An input element can be of type *text field*, *password field*, *checkbox*, *radio button*, *submit button*, *reset button*, *file select box*, as well as several [new input types](https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php) introduced in HTML5.

The most frequently used input types are described below.

##### Text Fields

Text fields are one line areas that allow the user to input text.

Single-line text input controls are created using an <input> element, whose type attribute has a value of text. Here's an example of a single-line text input used to take username:

***Example***

<form>

<label for="username">Username:</label>

<input type="text" name="username" id="username">

</form>

— The output of the above example will look something like this:



##### Password Field

Password fields are similar to text fields. The only difference is; characters in a password field are masked, i.e. they are shown as asterisks or dots. This is to prevent someone else from reading the password on the screen. This is also a single-line text input controls created using

an <input> element whose type attribute has a value of password.

***Example***

<form>

<label for="user-pwd">Password:</label>

<input type="password" name="user-password" id="user-pwd">

</form>

— The output of the above example will look something like this:



##### Radio Buttons

Radio buttons are used to let the user select exactly one option from a pre-defined set of options. It is created using an <input> element whose type attribute has a value of radio.

***Example***

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=html&file=radio-button)

<form>

<input type="radio" name="gender" id="male">

<label for="male">Male</label>

<input type="radio" name="gender" id="female">

<label for="female">Female</label>

</form>

— The output of the above example will look something like this:



##### Checkboxes

Checkboxes allows the user to select one or more option from a pre-defined set of options. It is created using an <input> element whose type attribute has a value of checkbox.

***Example***

<form>

<input type="checkbox" name="sports" id="soccer">

<label for="soccer">Soccer</label>

<input type="checkbox" name="sports" id="cricket">

<label for="cricket">Cricket</label>

<input type="checkbox" name="sports" id="baseball">

<label for="baseball">Baseball</label>

</form>

— The output of the above example will look something like this:



##### File Select box

The file fields allow a user to browse for a local file and send it as an attachment with the form data. Web browsers such as Google Chrome and Firefox render a file select input field with a Browse button that enables the user to navigate the local hard drive and select a file.

This is also created using an <input> element, whose type attribute value is set to file.

***Example***

<form>

<label for="file-select">Upload:</label>

<input type="file" name="upload" id="file-select">

</form>

— The output of the above example will look something like this:



##### Textarea

Textarea is a multiple-line text input control that allows a user to enter more than one line of text. Multi-line text input controls are created using an <textarea> element.

***Example***

<form>

<label for="address">Address:</label>

<textarea rows="3" cols="30" name="address" id="address"></textarea>

</form>

— The output of the above example will look something like this:



##### Select Boxes

A select box is a dropdown list of options that allows user to select one or more option from a pull- down list of options. Select box is created using the <select> element and <option> element.

The <option> elements within the <select> element define each list item.

***Example***

<form>

<label for="city">City:</label>

<select name="city" id="city">

<option value="sydney">Sydney</option>

<option value="melbourne">Melbourne</option>

<option value="cromwell">Cromwell</option>

</select>

</form>

— The output of the above example will look something like this:



##### Submit and Reset Buttons

A submit button is used to send the form data to a web server. When submit button is clicked the form data is sent to the file specified in the form's action attribute to process the submitted data.

A reset button resets all the forms control to default values. Try out the following example by typing your name in the text field, and click on submit button to see it in action.

***Example***

<form action="action.php" method="post">

<label for="first-name">First Name:</label>

<input type="text" name="first-name" id="first-name">

<input type="submit" value="Submit">

<input type="reset" value="Reset">

</form>

**HTML5 Colors**

<!DOCTYPE html>

<html>

<body>

<h1 style="background-color:Tomato;">Tomato</h1>

<h1 style="background-color:Orange;">Orange</h1>

<h1 style="background-color:DodgerBlue;">DodgerBlue</h1>

<h1 style="background-color:MediumSeaGreen;">MediumSeaGreen</h1>

<h1 style="background-color:Gray;">Gray</h1>

<h1 style="background-color:SlateBlue;">SlateBlue</h1>

<h1 style="background-color:Violet;">Violet</h1>

<h1 style="background-color:LightGray;">LightGray</h1>

</body>

</html>

**OUTPUT**

|  |
| --- |
| Tomato |
| Orange |
| DodgerBlue |
| MediumSeaGreen |
| Gray |
| SlateBlue |
| Violet |

LightGray

### HTML5 Audio

##### Embedding Audio in HTML Document

Inserting audio onto a web page was not easy before, because web browsers did not have a uniform standard for defining embedded media files like audio.

##### Using the HTML5 audio Element

The newly introduced HTML5 <audio> element provides a standard way to embed audio in web pages. However, the audio element is relatively new but it works in most of the modern web browsers.

The following example simply inserts an audio into the HTML5 document, using the browser default set of controls, with one source defined by the src attribute.

***Example***

<audio controls="controls" src="media/birds.mp3">

Your browser does not support the HTML5 Audio element.

</audio>

An audio, using the browser default set of controls, with alternative sources.

***Example***

<audio controls="controls">

<source src="media/birds.mp3" type="audio/mpeg">

<source src="media/birds.ogg" type="audio/ogg">

Your browser does not support the HTML5 Audio element.

</audio>

### HTML5 Video

##### Embedding Video in HTML Document

Inserting video onto a web page was not relatively easy, because web browsers did not have a uniform standard for defining embedded media files like video.

##### Using the HTML5 video Element

The newly introduced HTML5 <video> element provides a standard way to embed video in web pages. However, the video element is relatively new, but it works in most of the modern web browsers.

The following example simply inserts a video into the HTML document, using the browser default set of controls, with one source defined by the src attribute.

***Example***

<video controls="controls" src="media/shuttle.mp4">

Your browser does not support the HTML5 Video element.

</video>

A video, using the browser default set of controls, with alternative sources.

***Example***

<video controls="controls">

<source src="media/shuttle.mp4" type="video/mp4">

<source src="media/shuttle.ogv" type="video/ogg">

Your browser does not support the HTML5 Video element.

</video>

**New HTML5 Elements**

The most interesting new HTML5 elements are:

New **semantic elements** like <header>, <footer>, <article>, and <section>. New **attributes of form elements** like number, date, time, calendar, and range.

New **graphic elements**: <svg> and <canvas>.

New **multimedia elements**: <audio> and <video>.

##### What are Semantic Elements?

A semantic element clearly describes its meaning to both the browser and the developer.

Examples of **non-semantic** elements: <div> and <span> - Tells nothing about its content.

Examples of **semantic** elements: <form>, <table>, and <article> - Clearly defines its content.

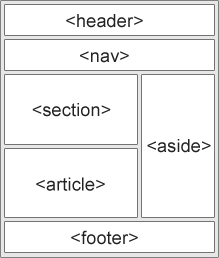
### New Semantic Elements in HTML5

Many web sites contain HTML code like:

<div id="nav"> <div class="header"> <div id="footer"> to indicate navigation, header, and footer.

HTML5 offers new semantic elements to define different parts of a web page:

* <article>
* <aside>
* <details>
* <figcaption>
* <figure>
* <footer>
* <header>
* <main>
* <mark>
* <nav>
* <section>
* <summary>
* <time>



##### HTML5 <section> Element

The <section> element defines a section in a document.

According to W3C's HTML5 documentation: "A section is a thematic grouping of content, typically with a heading."

A home page could normally be split into sections for introduction, content, and contact information.

##### Example

<section>

<h1>WWF</h1>

<p>The World Wide Fund for Nature (WWF) is </p>

</section>

###### HTML5 <article> Element

The <article> element specifies independent, self-contained content.

An article should make sense on its own, and it should be possible to read it independently from the rest of the web site.

Examples of where an <article> element can be used:

* Forum post
* Blog post
* Newspaper article

##### Example

<article>

<h1>What Does WWF Do?</h1>

<p>WWF's mission is to stop the degradation of our planet's natural environment, and build a future in which humans live in harmony with nature.</p>

</article>

##### HTML5 <header> Element

The <header> element specifies a header for a document or section.

The <header> element should be used as a container for introductory content.

You can have several <header> elements in one document. The following example defines a header for an article:

##### Example

<article>

<header>

<h1>What Does WWF Do?</h1>

<p>WWF's mission:</p>

</header>

<p>WWF's mission is to stop the degradation of our planet's natural environment, and build a future in which humans live in harmony with nature.</p>

</article>

##### HTML5 <footer> Element

The <footer> element specifies a footer for a document or section.

A <footer> element should contain information about its containing element.

A footer typically contains the author of the document, copyright information, links to terms of use, contact information, etc.

You may have several <footer> elements in one document.

##### Example

<footer>

<p>Posted by: Hege Refsnes</p>

<p>Contact information: <a href="<mailto:someone@example.com>"> [someone@example.com](mailto:someone@example.com)</a>.</p>

</footer>

##### HTML5 <figure> and <figcaption> Elements

The purpose of a figure caption is to add a visual explanation to an image.

In HTML5, an image and a caption can be grouped together in a <figure> element:

##### Example

<figure>

<img src="pic\_trulli.jpg" alt="Trulli">

<figcaption>Fig1. - Trulli, Puglia, Italy.</figcaption>

</figure>

**OUTPUT**

### Places to Visit

Puglia's most famous sight is the unique conical houses (Trulli) found in the area around Alberobello, a declared UNESCO World Heritage Site.



Fig.1 - Trulli, Puglia, Italy.

### Semantic Elements in HTML5

Below is an alphabetical list of the new semantic elements in HTML5. The links go to our complete [HTML5 Reference](https://www.w3schools.com/tags/default.asp).

|  |  |
| --- | --- |
| **Tag** | **Description** |
| [<article>](https://www.w3schools.com/tags/tag_article.asp) | Defines an article |
| [<aside>](https://www.w3schools.com/tags/tag_aside.asp) | Defines content aside from the page content |
| [<details>](https://www.w3schools.com/tags/tag_details.asp) | Defines additional details that the user can view or hide |
| [<figcaption>](https://www.w3schools.com/tags/tag_figcaption.asp) | Defines a caption for a <figure> element |

|  |  |
| --- | --- |
| [<figure>](https://www.w3schools.com/tags/tag_figure.asp) | Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc. |
| [<footer>](https://www.w3schools.com/tags/tag_footer.asp) | Defines a footer for a document or section |
| [<header>](https://www.w3schools.com/tags/tag_header.asp) | Specifies a header for a document or section |
| [<main>](https://www.w3schools.com/tags/tag_main.asp) | Specifies the main content of a document |
| [<mark>](https://www.w3schools.com/tags/tag_mark.asp) | Defines marked/highlighted text |
| [<nav>](https://www.w3schools.com/tags/tag_nav.asp) | Defines navigation links |
| [<section>](https://www.w3schools.com/tags/tag_section.asp) | Defines a section in a document |
| [<summary>](https://www.w3schools.com/tags/tag_summary.asp) | Defines a visible heading for a <details> element |
| [<time>](https://www.w3schools.com/tags/tag_time.asp) | Defines a date/time |

### HTML5 Drag and Drop

W3Schools

Drag the W3Schools image into the rectangle.

##### Drag and Drop

Drag and drop is a very common feature. It is when you "grab" an object and drag it to a different location. In HTML5, drag and drop is part of the standard: Any element can be draggable.

##### HTML Drag and Drop Example

The example below is a simple drag and drop example:

Example

<!DOCTYPE HTML>

<html>

<head>

<script>

function allowDrop(ev) { ev.preventDefault();

}

function drag(ev) { ev.dataTransfer.setData("text", ev.target.id);

}

function drop(ev) { ev.preventDefault();

var data = ev.dataTransfer.getData("text"); ev.target.appendChild(document.getElementById(data));

}

</script>

</head>

<body>

<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>

<img id="drag1" src="img\_logo.gif" draggable="true" ondragstart="drag(event)" width="336" height="69">

</body>

</html>

### OUTPUT

Drag the W3Schools image into the rectangle:



### HTML5 <nav> Element

The <nav> element defines a set of navigation links.

Notice that NOT all links of a document should be inside a <nav> element. The <nav> element is intended only for major block of navigation links.

### Example

<nav>

<a href="/html/">HTML</a> |

<a href="/css/">CSS</a> |

<a href="/js/">JavaScript</a> |

<a href="/jquery/">jQuery</a>

</nav>