**This book has been prepared exclusively for**

# VEIS COMPUTER EDUCATION

***HTML***

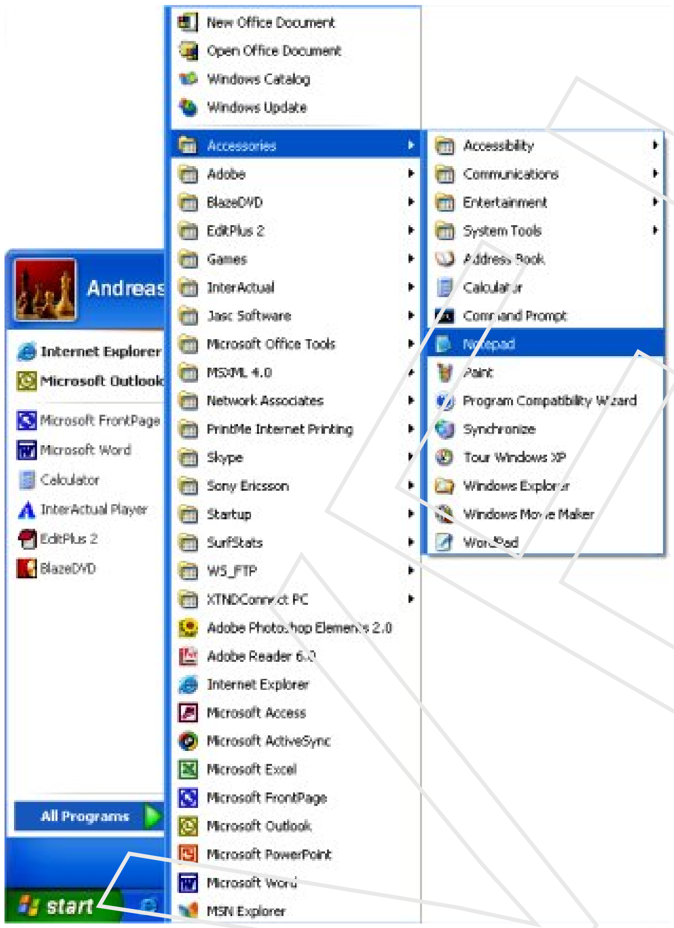
## Needs for HTML

Most likely you already have everything you need.

You have a "browser". A browser is the program that makes it possible to browse and open websites. Right now you are looking at this page in your browser.

It is not important which browser you use. The most common is Microsoft Internet Explorer. But there are others such as Opera and Mozilla Firefox and they can all be used.

A simple text editor is needed. If you are using Windows you can use Notepad, which is usually found in the start menu under Programs in Accessories:



A browser and Notepad (or a similar simple text editor) are all you need to go through this tutorial and make your own websites.

### Need of Internet:

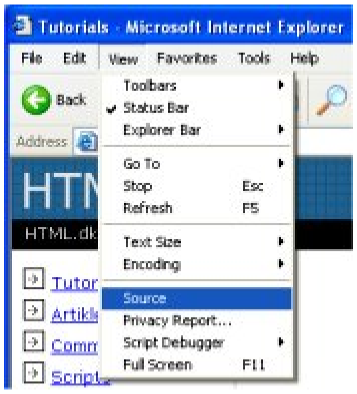
You do not need to be connected to the Internet - neither while reading this tutorial, nor while making your websites.

If you want to avoid being online while reading this tutorial, you can either print it out or simply disconnect from the Internet while reading on screen. You can make the website on your computer's hard disk and upload it to the Internet when it is finished.

# What is HTML?

**HTML is the "mother tongue" of your browser.**

HTML was invented in 1990 by a scientist called Tim Berners-Lee. The purpose was to make it easier for scientists at different universities to gain access to each other's research documents. The project became a bigger success than Tim Berners-Lee had ever imagined. By inventing HTML he laid the foundation for the web as we know it today.
HTML is a language, which makes it possible to present information on the Internet. What you see when you view a page on the Internet is your browser's interpretation of HTML. To see the HTML code of a page on the Internet, simply click "View" in the top menu of your browser and choose "Source".



HTML code looks complicated but we will help you make sense of it all.

## H-T-M-L stand for:

HTML is a language for describing web pages.

- HTML stands for **H**yper **T**ext **M**arkup **L**anguage
- HTML is a **markup** language
- A markup language is a set of markup **tags**
- The tags **describe** document content
- HTML documents contain HTML **tags** and plain **text**
- HTML documents are also called **web pages**

## HTML Tags

HTML markup tags are usually called HTML tags

- HTML tags are keywords (tag names) surrounded by **angle brackets** like <html>
- HTML tags normally **come in pairs** like <b> and </b>
- The first tag in a pair is the **start tag,** the second tag is the **end tag**
- The end tag is written like the start tag, with a **forward slash** before the tag name
- Start and end tags are also called **opening tags** and **closing tags**

<tagname>content</tagname>

**All text between the opening tag <em> and the closing tag </em> is emphasised in the browser. ("em" is short for "emphasis".) Emphasis text is look like Italics Text.**

**Example 1:**

```
<em>Emphasised text.</em>
```

Result: Will look like this in the browser:
*Emphasised text.*

The elements `h1`, `h2`, `h3`, `h4`, `h5` and `h6` is used to make headings (h stands for "heading"), where `h1` is the first level and normally the largest text, `h2` is the second level and normally slightly smaller text, and `h6` is the sixth and last in the hierarchy of headings and normally the smallest text.

**Example 2:**

```
<h1>This is a heading</h1>
<h2>This is a subheading</h2>
```

Will look like this in the browser:

## This is a heading

## This is a subheading

## Opening tag and a closing tag:

There's an exception to every rule and in HTML the exception is that there are a few elements which both open and close in the same tag. These so-called empty elements are not connected to a specific passage in the text but rather are isolated labels, for example, a line break which looks like this: <br />.

## Should tags be typed in uppercase or lowercase?

Most browsers might not care if you type your tags in upper, lower or mixed cases. <HTML>, <html> or <HtMl> will normally give the same result. However, the **correct** way is to type tags in lowercase. So get into the **habit of writing your tags in lowercase**.

## HTML Elements

"HTML tags" and "HTML elements" are often used to describe the same thing.

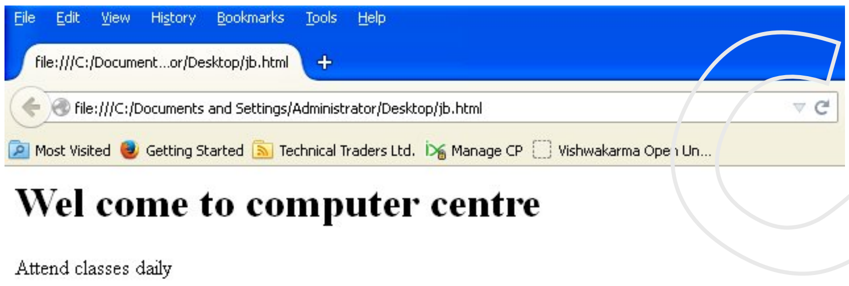
But strictly speaking, an HTML element is everything between the start tag and the end tag, including the tags:

HTML Element:

```
<p>This is a paragraph.</p>
```

## Web Browsers

The purpose of a web browser (such as Google Chrome, Internet Explorer, Firefox, Safari) is to read HTML documents and display them as web pages.
The browser does not display the HTML tags, but uses the tags to determine how the content of the HTML page is to be presented/ displayed to the user:


File Edit View History Bookmarks Tools Help
file:///C:/Document...or/Desktop/jb.html
file:///C:/Documents and Settings/Administrator/Desktop/jb.html
Most Visited
Getting Started
Technical Traders Ltd.
Manage CP
Vishwakarma Ope Un...

Wel come to computer centre

Attend classes daily

## HTML Page Structure

Below is a visualization of an HTML page structure:

```
<html>
<body >
<h1>Wel come to computer centre</h1>
<p>Attend classes daily</p>
</body>
</html>
```

## HTML Versions

Since the early days of the web, there have been many versions of HTML:

| Version | Year |
|---|---|
| HTML | 1991 |
| HTML+ | 1993 |
| HTML 2.0 | 1995 |
| HTML 3.2 | 1997 |
| HTML 4.01 | 1999 |
| XHTML | 2000 |
| HTML5 | 2012 |

## The <!DOCTYPE> Declaration

The <!DOCTYPE> declaration helps the browser to display a web page correctly.

There are many different documents on the web, and a browser can only display an HTML page 100% correctly if it knows the HTML type and version used.

## Common Declarations

### HTML5

```
<!DOCTYPE html>
```

### HTML 4.01

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.veinstitution.com/TR/html4/loose.dtd">
```

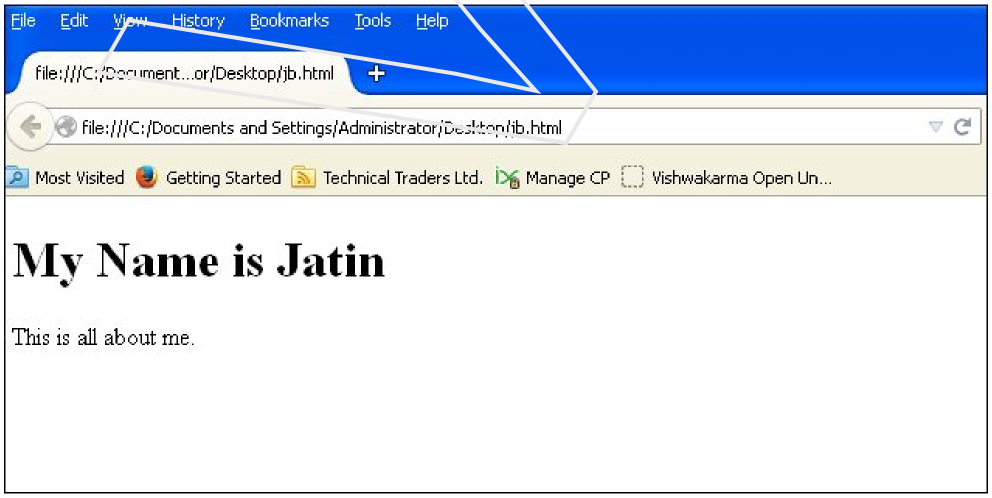
### XHTML 1.0

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.veinstitution.com/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

## HTML Example

```
<!DOCTYPE html>
<html>
<body>

<h1>My Name is Jatin</h1>

<p>This is all about me.</p>

</body>
</html>
```

**Result**:


File Edit View History Bookmarks Tools Help
file:///C:/Document...or/Desktop/jb.html
file:///C:/Documents and Settings/Administrator/Desktop/jb.html
Most Visited Getting Started Technical Traders Ltd. Manage CP Vishwakarma Open Un...

My Name is Jatin

This is all about me.

**Example Explained**

- The DOCTYPE declaration defines the document type
- The text between <html> and </html> describes the web page
- The text between <body> and </body> is the visible page content
- The text between <h1> and </h1> is displayed as a heading
- The text between <p> and </p> is displayed as a paragraph

## HTML Editors: Writing HTML Using Notepad or Text Edit

HTML can be edited by using a professional HTML editor like:

- Adobe Dreamweaver
- Microsoft Expression Web
- CoffeeCup HTML Editor

However, for learning HTML we recommend a text editor like Notepad (PC) or TextEdit (Mac). We believe using a simple text editor is a good way to learn HTML.

## Create your first website

With what you learned in the previous lessons, you are now only minutes away from making your first website.

**Open Notepad (in Accessories under Programs in the Start menu):**

**Let us start**: Let us start with something simple. A page that says: "Hurrah! This is my first website." Read on and you'll find out how simple it is.

HTML is simple and logical. **The browser reads HTML like you read English: from the top down and from left to right**. Thus, an simple HTML document begins with what should come first and ends with what should come last.

The first thing you need to do is to tell the browser that you will "talk" to it in the language HTML. This is done with the tag <html>. Type "<html>" in the first line of your document in Notepad.

As <html> is an opening tag and must be closed with a closing tag when you are finished typing HTML. So to make sure you don't forget the HTML close tag now type "</html>" a couple of lines down and write the rest of the document between <html> and </html>.

The next thing your document needs is a "head", which provides information about your document, and a "body", which is the content of the document. Since HTML is nothing if not logical, the head (<head> and </head>) is on top of the body (<body> and </body>).

Your document should now look like this:

```
<html>

  <head>
  </head>

  <body>
  </body>

</html>
```

Note It is strongly recommended that you structure your HTML in a neat way with line breaks and indents, like the above example.
If your document looks like the above example, you have made your first.

## Add content to website.

Your HTML document has two parts: first head and second body. In the head section you write information about the page, while the body contains the information that constitutes the page.
For example, if you want to give the page a title which will appear in the top bar of the browser, it should be done in the "head" section. The element used for a title is title. I.e. write the title of the page between the opening tag <title> and the closing tag </title>:

```
<title>My first website</title>
```

Note that this title will not appear on the page itself. Anything you want to appear on the page is content and must therefore be added between the "body" tags.
We want the page to say "Hurrah! This is my first website." This is the text that we want to communicate and it therefore belongs in the body section. So in the body section, type the following:

```
<p>Hurrah! This is my first website.</p>
```

The p in <p> is short for "paragraph" which is exactly what it is - a text paragraph.
Your HTML document should now look like this:

```
<html>

  <head>
  <title>My first website </title>
  </head>

  <body>
  <p>Hurrah! This is my website. Congratulations to Jatin</p>
  </body>

</html>
```

Done! You have now made your first real website!

Next all you have to do is to save it to your hard drive and then open it in your browser:

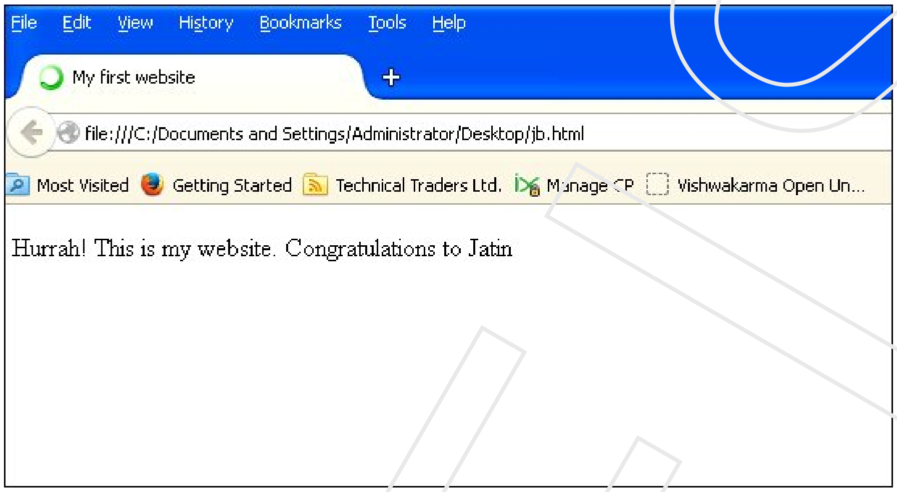
- In Notepad choose "Save as..." under "File" in the top menu.
- Choose "All Files" in the "Save as type" box. This is very important - otherwise, you save it as a text document and not as an HTML document.

- Now save your document as "page1.htm" (the ending ".htm" indicates that it is an HTML document. ".html" gives the same result.

Now go to the browser:

- In the top menu choose "Open" under "File" (or press CTRL+O).
- Click "Browse" in the box that appears.
- Now find your HTML document and click "Open".

**Result**:


File Edit View History Bookmarks Tools Help
My first website
file:///C:/Documents and Settings/Administrator/Desktop/jb.html
Most Visited Getting Started Technical Traders Ltd. Manage CP Vishwakarma Open Un...

Hurrah! This is my website. Congratulations to Jatin

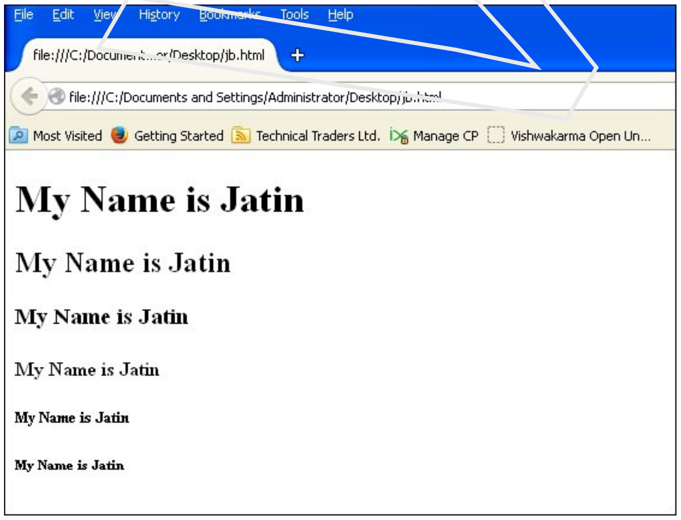
## HTML Basic : HTML Headings

HTML headings are defined with the <h1> to <h6> tags.

**Example**

<h1>My Name is Jatin</h1>
<h2> My Name is Jatin</h2>
<h3> My Name is Jatin</h3>
<h4>My Name is Jatin</h4>
<h5> My Name is Jatin</h5>
<h6> My Name is Jatin</h6>

**Result:**


File Edit View History Bookmarks Tools Help
file:///C:/Documen...r/Desktop/jb.html
file:///C:/Documents and Settings/Administrator/Desktop/jb.html
Most Visited Getting Started Technical Traders Ltd. Manage CP Vishwakarma Open Un...

# My Name is Jatin
## My Name is Jatin
### My Name is Jatin
#### My Name is Jatin
##### My Name is Jatin
###### My Name is Jatin

### HTML Paragraphs
HTML paragraphs are defined with the <p> tag.

### Example
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>

## HTML Links
HTML links are defined with the <a> tag.

### Example
<a href="http://www.veinstitution.com">This is a link</a>
Note: The link address is specified in the href attribute.

## HTML Images
HTML images are defined with the <img> tag.

### Example
<img src="veis1.jpg" alt="veinstitution.com" width="104" height="142">

Note: The filename and the size of the image are provided as attributes.

## HTML Elements

An HTML element is everything from the start tag to the end tag:

| Start tag * | Element content | End tag * |
|---|---|---|
| <p> | This is a paragraph | </p> |
| <a href="default.htm"> | This is a link | </a> |
| <br> | | |

Note : * The start tag is often called the opening tag. The end tag is often called the closing tag.

## HTML Element Syntax

- An HTML element starts with a **start tag / opening tag**
- An HTML element ends with an **end tag / closing tag**
- The **element content** is everything between the start and the end tag
- Some HTML elements have **empty content**
- Empty elements are **closed in the start tag**
- Most HTML elements can have **attributes**

## Nested HTML Elements

Most HTML elements can be nested (can contain other HTML elements).

HTML documents consist of nested HTML elements.

## HTML Document Example

```
<!DOCTYPE html>
<html>
<body>
<p>This is my first paragraph.</p>
</body>
</html>
```

The example above contains 3 HTML elements.

## HTML Example Explained

**The <p> element:**

<p>This is my first paragraph.</p>

The <p> element defines a paragraph in the HTML document.
The element has a start tag <p> and an end tag </p>.
The element content is: This is my first paragraph.

**The <body> element:**

```
<body>
<p>This is my first paragraph.</p>
</body>
```

The <body> element defines the body of the HTML document.
The element has a start tag <body> and an end tag </body>.
The element content is another HTML element (a p element).

**The <html> element:**

```
<html>

<body>
<p>This is my first paragraph.</p>
</body>

</html>
```

The <html> element defines the whole HTML document.
The element has a start tag <html> and an end tag </html>.
The element content is another HTML element (the body element).

## Don't Forget the End Tag

Some HTML elements might display correctly even if you forget the end tag:

```
<p>This is a paragraph
<p>This is a paragraph
```

The example above works in most browsers, because the closing tag is considered optional.

Never rely on this. Many HTML elements will produce unexpected results and/or errors if you forget the end tag .

## Empty HTML Elements

HTML elements with no content are called empty elements.

<br> is an empty element without a closing tag (the <br> tag defines a line break).

## HTML Tip: Use Lowercase Tags

HTML tags are not case sensitive: <P> means the same as <p>. Many web sites use uppercase HTML tags.

We use lowercase tags because the World Wide Web Consortium (W3C) **recommends** lowercase in HTML 4, and **demands** lowercase tags in XHTML.

## HTML Attributes

- HTML elements can have **attributes**
- Attributes provide **additional information** about an element
- Attributes are always specified in **the start tag**
- Attributes come in name/value pairs like: **name="value"**

### What is an attribute?

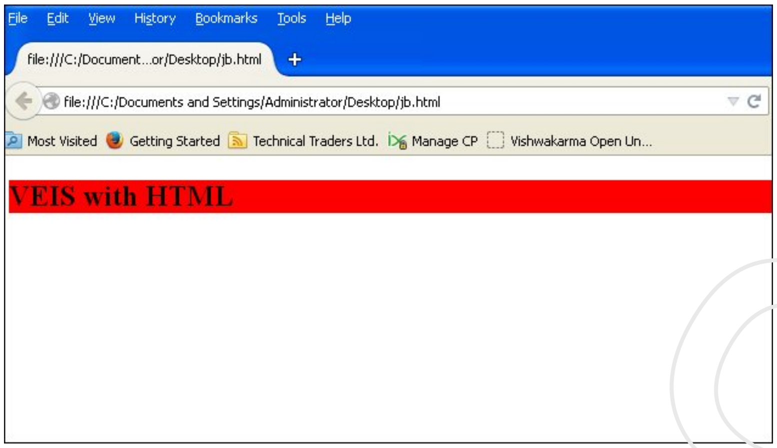
Elements give structure to a HTML document and tells the browser how you want your website to be presented (for example, <br /> informs the browser to make a line break). In some elements you can add more information. Such additional information is called an attribute.

**Example 1:**

```
<h2 style="background-color:#ff0000;">VEIS with HTML</h2>
```

Attributes are always written within a start tag and are followed by an equals sign and the attribute details written between inverted commas. The semicolon after the attribute is for separating different style commands.

**Result:**


File Edit View History Bookmarks Tools Help
file:///C:/Document...or/Desktop/jb.html
file:///C:/Documents and Settings/Administrator/Desktop/jb.html
Most Visited Getting Started Technical Traders Ltd. Manage CP Vishwakarma Open Un...

VEIS with HTML

## What is the catch?

There are many different attributes. The first one you will learn is style. With the style attribute you can add layout to your website. For instance a background colour:

**Example 2:**

```
<html>

  <head>
  </head>

  <body style="background-color:#ff0000;">
  </body>

</html>
```

will show a completely red page in the browser - go ahead and see for yourself.

## Attribute Example

HTML links are defined with the <a> tag. The link address is specified in the href attribute:

## Example

<a href="http://www.veinstitution.com">This is a link</a>
Always Quote Attribute Values
Attribute values should always be enclosed in quotes.
Double style quotes are the most common, but single style quotes are also allowed.
Tip: In some rare situations, when the attribute value itself contains quotes, it is necessary to use single quotes: name='Jatin "Master" Bedi'

## HTML Attributes Reference

Below is a list of some attributes that can be used on any HTML element:

| Attribute | Description |
|---|---|
| class | Specifies one or more classnames for an element (refers to a class in a style sheet) |
| id | Specifies a unique id for an element |
| style | Specifies an inline CSS style for an element |
| title | Specifies extra information about an element (displayed as a tool tip) |

## HTML Global Attributes

H5 = Attribute added in HTML5.

| Attribute | Description |
|---|---|
| accesskey | Specifies a shortcut key to activate/focus an element |
| class | Specifies one or more classnames for an element (refers to a class in a style sheet) |
| contenteditable H5 | Specifies whether the content of an element is editable or not |
| contextmenu H5 | Specifies a context menu for an element. The context menu appears when a user right-clicks on the element |
| data-* H5 | Used to store custom data private to the page or application |
| dir | Specifies the text direction for the content in an element |
| draggable H5 | Specifies whether an element is draggable or not |
| dropzone H5 | Specifies whether the dragged data is copied, moved, or linked, when dropped |
| hidden H5 | Specifies that an element is not yet, or is no longer, relevant |
| id | Specifies a unique id for an element |
| lang | Specifies the language of the element's content |
| spellcheck H5 | Specifies whether the element is to have its spelling and grammar checked or not |
| style | Specifies an inline CSS style for an element |
| tabindex | Specifies the tabbing order of an element |
| title | Specifies extra information about an element |
| translate H5 | Specifies whether the content of an element should be translated or not |

## HTML Reference – Including HTML5

## Ordered by Function

H5 = Tag added in HTML5.

| Tag | Description |
|---|---|
| **Basic** | |
| <!DOCTYPE> | Defines the document type |
| <html> | Defines an HTML document |
| <title> | Defines a title for the document |
| <body> | Defines the document's body |
| <h1> to <h6> | Defines HTML headings |
| <p> | Defines a paragraph |
| <br> | Inserts a single line break |
| <hr> | Defines a thematic change in the content |
| <!--...--> | Defines a comment |
| **Formatting** | |

| | |
|---|---|
| <acronym> | Not supported in HTML5. Use <abbr> instead.<br>Defines an acronym |
| <abbr> | Defines an abbreviation |
| <address> | Defines contact information for the author/owner of a document/article |
| <b> | Defines bold text |
| <bdi> H5 | Isolates a part of text that might be formatted in a different direction from other text outside it |
| <bdo> | Overrides the current text direction |
| <big> | Not supported in HTML5. Use CSS instead.<br>Defines big text |
| <blockquote> | Defines a section that is quoted from another source |
| <center> | Not supported in HTML5. Use CSS instead. Defines centered text |
| <cite> | Defines the title of a work |
| <code> | Defines a piece of computer code |
| <del> | Defines text that has been deleted from a document |
| <dfn> | Defines a definition term |
| <em> | Defines emphasized text |
| <font> | Not supported in HTML5. Use CSS instead.<br>Defines font, color, and size for text |
| <i> | Defines a part of text in an alternate voice or mood |
| <ins> | Defines a text that has been inserted into a document |
| <kbd> | Defines keyboard input |
| <mark> H5 | Defines marked/highlighted text |
| <meter> H5 | Defines a scalar measurement within a known range (a gauge) |
| <pre> | Defines preformatted text |
| <progress> H5 | Represents the progress of a task |
| <q> | Defines a short quotation |
| <rp> H5 | Defines what to show in browsers that do not support ruby annotations |
| <rt> H5 | Defines an explanation/pronunciation of characters (for East Asian typography) |
| <ruby> H5 | Defines a ruby annotation (for East Asian typography) |
| <s> | Defines text that is no longer correct |
| <samp> | Defines sample output from a computer program |
| <small> | Defines smaller text |
| <strike> | Not supported in HTML5. Use <del> instead.<br>Defines strikethrough text |
| <strong> | Defines important text |
| <sub> | Defines subscripted text |
| <sup> | Defines superscripted text |
| <time> H5 | Defines a date/time |

| | |
|---|---|
| &lt;tt&gt; | Not supported in HTML5. Use CSS instead.<br>Defines teletype text |
| &lt;u&gt; | Defines text that should be stylistically different from normal text |
| &lt;var&gt; | Defines a variable |
| &lt;wbr&gt; H5 | Defines a possible line-break |
| **Forms** | |
| &lt;form&gt; | Defines an HTML form for user input |
| &lt;input&gt; | Defines an input control |
| &lt;textarea&gt; | Defines a multiline input control (text area) |
| &lt;button&gt; | Defines a clickable button |
| &lt;select&gt; | Defines a drop-down list |
| &lt;optgroup&gt; | Defines a group of related options in a drop-down list |
| &lt;option&gt; | Defines an option in a drop-down list |
| &lt;label&gt; | Defines a label for an &lt;input&gt; element |
| &lt;fieldset&gt; | Groups related elements in a form |
| &lt;legend&gt; | Defines a caption for a &lt;fieldset&gt; element |
| &lt;datalist&gt; H5 | Specifies a list of pre-defined options for input controls |
| &lt;keygen&gt; H5 | Defines a key-pair generator field (for forms) |
| &lt;output&gt; H5 | Defines the result of a calculation |
| **Frames** | |
| &lt;frame&gt; | Not supported in HTML5.<br>Defines a window (a frame) in a frameset |
| &lt;frameset&gt; | Not supported in HTML5.<br>Defines a set of frames |
| &lt;noframes&gt; | Not supported in HTML5.<br>Defines an alternate content for users that do not support frames |
| &lt;iframe&gt; | Defines an inline frame |
| **Images** | |
| &lt;img&gt; | Defines an image |
| &lt;map&gt; | Defines a client-side image-map |
| &lt;area&gt; | Defines an area inside an image-map |
| &lt;canvas&gt; H5 | Used to draw graphics, on the fly, via scripting (usually JavaScript) |
| &lt;figcaption&gt; H5 | Defines a caption for a &lt;figure&gt; element |
| &lt;figure&gt; H5 | Specifies self-contained content |
| **Audio/Video** | |
| &lt;audio&gt; H5 | Defines sound content |
| &lt;source&gt; H5 | Defines multiple media resources for media elements (&lt;video&gt; and &lt;audio&gt;) |
| &lt;track&gt; H5 | Defines text tracks for media elements (&lt;video&gt; and &lt;audio&gt;) |
| &lt;video&gt; H5 | Defines a video or movie |
| **Links** | |

| | |
|---|---|
| <a> | Defines a hyperlink |
| <link> | Defines the relationship between a document and an external resource (most used to link to style sheets) |
| <nav> H5 | Defines navigation links |
| **Lists** | |
| <ul> | Defines an unordered list |
| <ol> | Defines an ordered list |
| <li> | Defines a list item |
| <dir> | Not supported in HTML5. Use <ul> instead. Defines a directory list |
| <dl> | Defines a description list |
| <dt> | Defines a term/name in a description list |
| <dd> | Defines a description of a term/name in a description list |
| <menu> | Defines a list/menu of commands |
| <menuitem> H5 | Defines a command/menu item that the user can invoke from a popup menu |
| **Tables** | |
| <table> | Defines a table |
| <caption> | Defines a table caption |
| <th> | Defines a header cell in a table |
| <tr> | Defines a row in a table |
| <td> | Defines a cell in a table |
| <thead> | Groups the header content in a table |
| <tbody> | Groups the body content in a table |
| <tfoot> | Groups the footer content in a table |
| <col> | Specifies column properties for each column within a <colgroup> element |
| <colgroup> | Specifies a group of one or more columns in a table for formatting |
| **Style/Sections** | |
| <style> | Defines style information for a document |
| <div> | Defines a section in a document |
| <span> | Defines a section in a document |
| <header> H5 | Defines a header for a document or section |
| <footer> H5 | Defines a footer for a document or section |
| <section> H5 | Defines a section in a document |
| <article> H5 | Defines an article |
| <aside> H5 | Defines content aside from the page content |
| <details> H5 | Defines additional details that the user can view or hide |
| <dialog> H5 | Defines a dialog box or window |
| <summary> H5 | Defines a visible heading for a <details> element |
| **Meta Info** | |
| <head> | Defines information about the document |

| | |
|---|---|
| <meta> | Defines metadata about an HTML document |
| <base> | Specifies the base URL/target for all relative URLs in a document |
| <basefont> | Not supported in HTML5. Use CSS instead. Specifies a default color, size, and font for all text in a document |
| **Programming** | |
| <script> | Defines a client-side script |
| <noscript> | Defines an alternate content for users that do not support client-side scripts |
| <applet> | Not supported in HTML5. Use <object> instead. Defines an embedded applet |
| <embed> H5 | Defines a container for an external (non-HTML) application |
| <object> | Defines an embedded object |
| <param> | Defines a parameter for an object |

Table No 1.1

## HTML Headings

Headings are defined with the <h1> to <h6> tags.

<h1> defines the most important heading. <h6> defines the least important heading.

### Example

<h1>This is a heading</h1>
<h2>This is a heading</h2>
<h3>This is a heading</h3>

**Note:** Browsers automatically add some empty space (a margin) before and after each heading.

## Headings Are Important

Use HTML headings for headings only. Don't use headings to make text BIG or bold.
Search engines use your headings to index the structure and content of your web pages.
Since users may read quickly your pages by its headings, it is important to use headings to show the document structure.
H1 headings should be used as main headings, followed by H2 headings, then the less important H3 headings, and so on.

## HTML Lines

The <hr> tag creates a horizontal line in an HTML page.

The hr element can be used to separate content:

**Example**

```
<p>This is a paragraph.</p>
<hr>
<p>This is a paragraph.</p>
<hr>
<p>This is a paragraph.</p>
```

## HTML Paragraphs

Paragraphs are defined with the <p> tag.

## Example

```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```

**Note:** Browsers automatically add an empty line before and after a paragraph.

## HTML Line Breaks

Use the <br> tag if you want a line break (a new line) without starting a new paragraph:

### Example

```
<p>This is<br>a para<br>graph with line breaks</p>
```

The <br> element is an empty HTML element. It has no end tag.

## HTML Output - Useful Tips

You cannot be sure how HTML will be displayed. Large or small screens, and resized windows will create different results.
With HTML, you cannot change the output by adding extra spaces or extra lines in your HTML code.
The browser will remove extra spaces and extra lines when the page is displayed. Any number of lines count as one line, and any number of spaces count as one space.

## HTML Formatting Tags

HTML uses tags like <b> and <i> for formatting output, like **bold** or *italic* text.

These HTML tags are called formatting tags (As given in table no 1.1 ).

**Often <strong> renders as <b>, and <em> renders as <i>.**

However, there is a difference in the meaning of these tags:

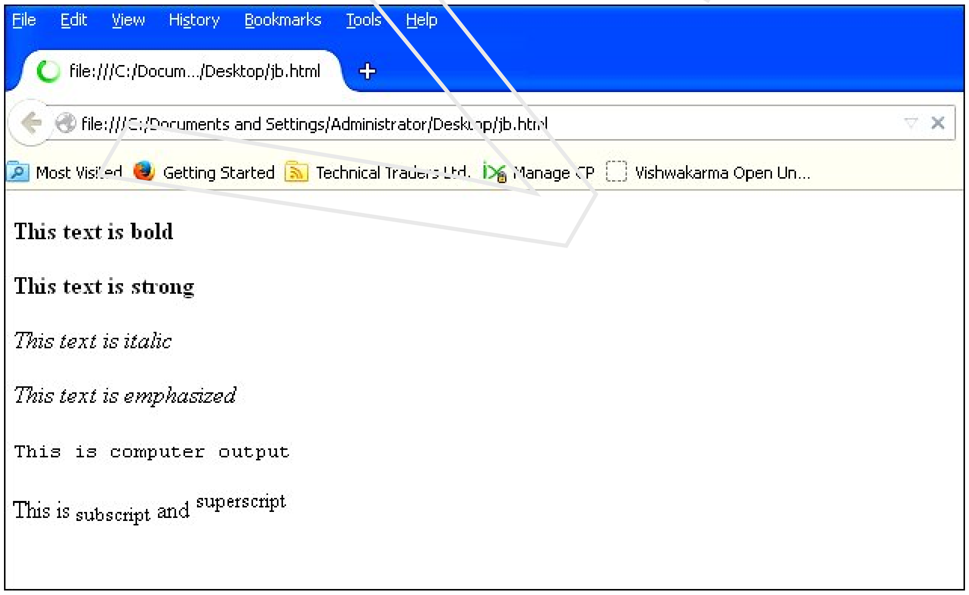
<b> or <i> defines bold or italic text only.

<strong> or <em> means that you want the text to be rendered in a way that the user understands as "important". Today, all major browsers render strong as bold and em as italics. However, if a browser one day wants to make a text highlighted with the strong feature, it might be cursive for example and not bold!

## Example:

```
<!DOCTYPE html>

<html>

<body>

<p><b>This text is bold</b></p>

<p><strong>This text is strong</strong></p>

<p><i>This text is italic</i></p>

<p><em>This text is emphasized</em></p>

<p><code>This is computer output</code></p>

<p>This is<sub> subscript</sub> and <sup>superscript</sup></p>

</body>

</html>
```

## Result:


File Edit View History Bookmarks Tools Help
file:///C:/Docum.../Desktop/jb.html
file:///C:/Documents and Settings/Administrator/Desktop/jb.html
Most Visited Getting Started Technical Traders Ltd. Manage CP Vishwakarma Open Un...

**This text is bold**

**This text is strong**

*This text is italic*

*This text is emphasized*

```
This is computer output
```

This is $_{\text{subscript}}$ and $^{\text{superscript}}$

## HTML Comment Tags

You can add comments to your HTML source by using the following syntax:
<!-- Your comments here -->
**Note:** There is an exclamation point (!) in the opening tag, but not in the closing tag.

Comments are not displayed by the browser, but they can help document your HTML.

With comments you can place notifications and reminders in your HTML:

**Example   <!-- This is a comment about Jatin -->**

**<p>Jatin is a good developer.</p>**

**<!-- Remember to add more information here -->**

### Result :

```
Jatin is a good developer.
```

Comments are also great for debugging HTML, because you can comment out HTML lines of code, one at a time, to search for errors:

### Example

```
<!DOCTYPE html>
<html>
<body>
<!-- Do not display this at the moment
<img border="1" src="/images/jatin.jpg" alt="Jatin Bedi" width="304" height="228">
-->
</body>
</html>
```

### Result

Note: Image specifications are given in comment tag, so image will not be displayed in page.

## Software Program Tags

HTML comments tags can also be generated by various HTML software programs.

For example the <!--webbot bot--> tags which are wrapped inside HTML comments by FrontPage.

As a rule, let these tags stay, to help support the software.

## Conditional Comments

Only Internet Explorer recognizes conditional comments.

Conditional comments enable you to add a browser specific code that executes only if the browser is IE but is treated as a comment by other browsers.

You can add conditional comments to your HTML document by using the following syntax:

### Example

```
<!--[if IE 5]>This is IE 5<br><![endif]-->
<!--[if IE 6]>This is IE 6<br><![endif]-->
<!--[if IE 7]>This is IE 7<br><![endif]-->
<!--[if IE 8]>This is IE 8<br><![endif]-->
<!--[if IE 9]>This is IE 9<br><![endif]-->
```

## Make a link:

To make links, you use what you always use when coding HTML: an element. **A simple element with one attribute and you will be able to link to anything and everything**. Here is an example of what a link to Veinstitution.com could look like:

### Example 1:

```
<a href="http://www.veinstitution.com/">Here is a link to Veinstitution.com</a>
```

Would look like this in the browser:

Here is a link to Veinstitution.com

The element a stands for "anchor". And the attribute href is short for "hypertext reference", which specifies where the link leads to - typically an address on the internet or a file name.

In the above example the attribute href has the value "http://www.veinstitution.com", which is the full address of Veinstitution.com and is called a URL (Uniform Resource Locator). Note that "http://" must always be included in URLs. The sentence "Here is a link to Veinstitution.com" is the text that is shown in the browser as the link. Remember to close the element with an </a>.

### Links between my own pages:

If you want to make a link between pages on the same website, you do not need to spell out the entire address (URL) for the document. For example, if you have made two pages (let us call them page1.htm and page2.htm) and saved them in the same folder you can make a link from one page to the other by only typing the name of the file in the link. Under such circumstances a link from page1.htm to page2.htm could look like this:

### Example 2:

```
<a href="page2.htm">Click here to go to page 2</a>
```

If page 2 were placed in a subfolder (named "subfolder"), the link could look like this:

### Example 3:

```
<a href="subfolder/page2.htm">Click here to go to page 2</a>
```

The other way around, a link from page 2 (in the subfolder) to page 1 would look like this:

### Example 4:

```
<a href="../page1.htm">A link to page 1</a>
```

"../" points to the folder one level up from position of the file from which the link is made. Following the same system, you can also point two (or more) folders up by writing "../../".

Alternatively, you can always type the complete address for the file (URL).

## HTML Hyperlinks (Links)

The HTML <a> tag defines a hyperlink.

A hyperlink (or link) is a word, group of words, or image that you can click on to jump to another document.
When you move the cursor over a link in a Web page, the arrow will turn into a little hand.
The most important attribute of the <a> element is the href attribute, which indicates the link's destination.

By default, links will appear as follows in all browsers:

- An unvisited link is underlined and blue
- A visited link is underlined and purple
- An active link is underlined and red

## HTML Link Syntax

The HTML code for a link is simple. It looks like this:

<a href="*url*">*Link text*</a>

The href attribute specifies the destination of a link.

### Example

```
<a href="http://www.veinstitution.com/">Visit VISHAL EDUCATION</a>
```

which will display like this: Visit VISHAL EDUCATION

Clicking on this hyperlink will send the user to veinstitution's homepage.

**Tip:** The "*Link text*" doesn't have to be text. It can be an image or any other HTML element.

## HTML Links - The target Attribute

The target attribute specifies where to open the linked document.

## Internal links within a page:

You can also create internal links within a page - for example a table of contents at the top with links to each chapter below. All you need to use is a very useful attribute called id (identification) and the symbol "#".

Use the id attribute to mark the element to which you want to link. For example:

```
<h1 id="heading1">heading 1</h1>
```

You can now create a link to that element by using "#" in the link attribute. The "#" must be followed by the id of the tag you want to link to. For example:

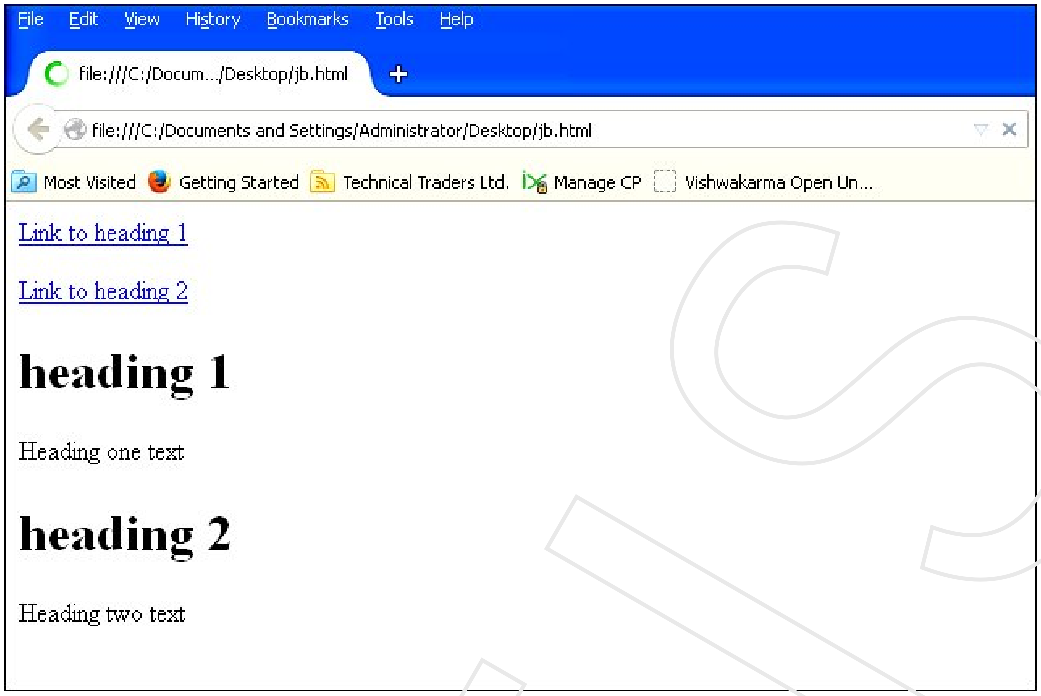
```
<a href="#heading1">Link to heading 1</a>
```

All will become clear with an example:

### Example 5:

```
<html>

  <head>
  </head>

  <body>

        <p><a href="#heading1">Link to heading 1</a></p>
        <p><a href="#heading2">Link to heading 2</a></p>

        <h1 id="heading1">heading 1</h1>
        <p>Heading one text</p>

        <h1 id="heading2">heading 2</h1>
        <p> Heading two text </p>

  </body>

</html>
```

**Result**



Link to heading 1

Link to heading 2

heading 1

Heading one text

heading 2

Heading two text

## Link to anything else

You can also make a link to an e-mail address. It is done in almost the same way as when you link to a document:

**Example 6:**

<a href="mailto:nobody@veinstitution.com">Send an e-mail to nobody at Veinstitution.com</a>

will look like this in the browser:

Send an e-mail to nobody at VEINSTITUTION.COM

The only difference between a link to an e-mail and a link to a file is that instead of typing the address of a document, you type mailto: followed by an e-mail address. When the link is clicked, the default e-mail program opens with a new blank message addressed to the specified e-mail address.

## Are there any other attributes I should know of?

To create a link, you always have to use the href attribute. In addition, you can also put a title on your link:

**Example 7:**

<a href="http://www.veinstitution.com/" title="Visit veinstitution.com and learn HTML">Veinstitution.com</a>

Would look like this in the browser:

veinstitution.com

The title attribute is used to type a short description of the link. If you - without clicking - place the cursor over the link, you will see the text "Visit Veinstitution.com and learn HTML" appears.

## HTML Links - The id Attribute

The id attribute can be used to create a bookmark inside an HTML document.

**Tip:** Bookmarks are not displayed in any special way. They are invisible to the reader.

### Example

An anchor with an id inside an HTML document:
<a id=" course "> Course Section</a>
Create a link to the " Course Section" inside the same document:
<a href="# course">Visit the Course Section</a>
Or, create a link to the "Course Section" from another page:
<a href="http://www.veinstitution.com/html_links.htm# course">
Visit the Course Section</a>

## Basic Notes - Useful Tips

**Note:** Always add a trailing slash to subfolder references. If you link like this: href="http://www.veinstitution.com/html", you will generate two requests to the server, the server will first add a slash to the address, and then create a new request like this: href="http://www.veinstitution.com/html/".

## The HTML <head> Element

The <head> element is a container for all the head elements. Elements inside <head> can include scripts, instruct the browser where to find style sheets, provide meta information, and more. The following tags can be added to the head section: <title>, <style>, <meta>, <link>, <script>, <noscript>, and <base>.

## The HTML <title> Element

The <title> tag defines the title of the document.

The <title> element is required in all HTML/XHTML documents.

The <title> element:

- defines a title in the browser toolbar
- provides a title for the page when it is added to favorites
- displays a title for the page in search-engine results

A simplified HTML document:

```
<!DOCTYPE html>
<html>
<head>
<title>Title of the document</title>
</head>

<body>
The content of the document......
</body>

</html>
```

## The HTML <base> Element

The <base> tag specifies the base URL/target for all relative URLs in a page:

```
<head>
<base href="http://www.veinstitution.com/images/" target="_blank">
</head>
```

## The HTML <link> Element

The <link> tag defines the relationship between a document and an external resource.

The <link> tag is most used to link to style sheets:

```
<head>
<link rel="stylesheet" type="text/css" href="style.css">
</head>
```

## The HTML <style> Element

The <style> tag is used to define style information for an HTML document.

Inside the <style> element you specify how HTML elements should render in a browser:

```
<head>
<style type="text/css">
body {background-color:yellow;}
p {color:blue;}
</style>
</head>
```

## The HTML <meta> Element

Metadata is data (information) about data. The <meta> tag provides metadata about the HTML document. Metadata will not be displayed on the page, but will be machine passable.

Meta elements are typically used to specify page description, keywords, author of the document, last modified, and other metadata.

The metadata can be used by browsers (how to display content or reload page), search engines (keywords), or other web services. <meta> tags always go inside the <head> element.

## <meta> Tags - Examples of Use

**Define keywords for search engines:**

<meta name="keywords" content="HTML, CSS, XML, XHTML, JavaScript">

**Define a description of your web page:**

<meta name="description" content="Free Web tutorials on HTML and CSS">

**Define the author of a page:**

<meta name="author" content="Hege Refsnes">

**Refresh document every 30 seconds:**

<meta http-equiv="refresh" content="30">

## The HTML <script> Element

The <script> tag is used to define a client-side script, such as a JavaScript.

## Styling HTML

- Inline - using the style **attribute** in HTML elements
- Internal - using the <style> **element** in the <head> section
- External - using an external CSS **file**

The preferred way to add CSS to HTML, is to put CSS syntax in separate CSS files. However, in this HTML tutorial we will introduce you to CSS using the style attribute. This is done to simplify the examples. It also makes it easier for you to edit the code and try it yourself.

## Inline Styles
An inline style can be used if a unique style is to be applied to one single occurrence of an element.
To use inline styles, use the style attribute in the relevant tag. The style attribute can contain any CSS property. The example below shows how to change the text color and the left margin of a paragraph:

<p style="color:blue;margin-left:20px;">This is a paragraph.</p>

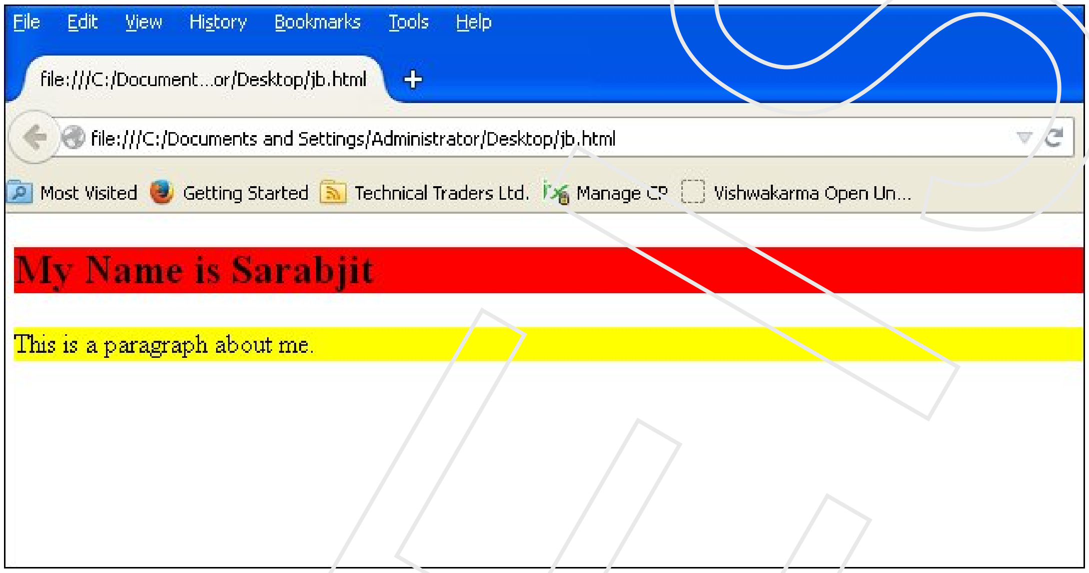
### HTML Style Example - Background Color

The background-color property defines the background color for an element:

## Example

```
<!DOCTYPE html>
<html>
<body style="background-color:white;">
<h2 style="background-color:red;">My Name is Sarabjit</h2>
<p style="background-color:yellow;">This is a paragraph about me.</p>
</body>
</html>
```

## Result


File Edit View History Bookmarks Tools Help
file:///C:/Document...or/Desktop/jb.html
file:///C:/Documents and Settings/Administrator/Desktop/jb.html
Most Visited Getting Started Technical Traders Ltd. Manage CP Vishwakarma Open Un...

My Name is Sarabjit

This is a paragraph about me.

The background-color property makes the "old" bgcolor attribute obsolete.

## HTML Style Example - Font, Color and Size

The font-family, color, and font-size properties defines the font, color, and size of the text in an element:

## Example

```
<!DOCTYPE html>
<html>
<body>
<h1 style="font-family:verdana;">A heading</h1>
<p style="font-family:arial;color:red;font-size:20px;">A paragraph.</p>
</body>
</html>
```

## Try your self

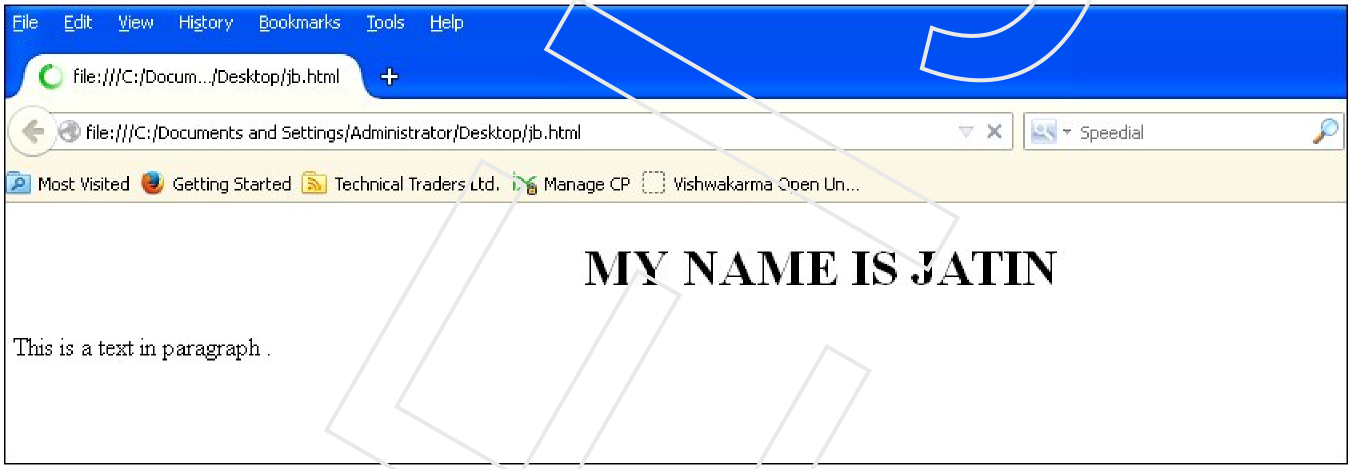
## HTML Style Example - Text Alignment

The text-align property specifies the horizontal alignment of text in an element:

### Example

```
<!DOCTYPE html>
<html>
<body>
<h1 style="text-align:center;">MY NAME IS JATIN</h1>
<p>This is a text in paragraph .</p>
</body>
</html>
```

### Result


File Edit View History Bookmarks Tools Help
file:///C:/Docum.../Desktop/jb.html
file:///C:/Documents and Settings/Administrator/Desktop/jb.html
Speedial
Most Visited
Getting Started
Technical Traders Ltd.
Manage CP
Vishwakarma Open Un...

MY NAME IS JATIN

This is a text in paragraph .

The text-align property makes the old <center> tag obsolete.

## Internal Style Sheet

An internal style sheet can be used if one single document has a unique style. Internal styles are defined in the <head> section of an HTML page, by using the <style> tag, like this:

```
<head>
<style>
body {background-color:yellow;}
p {color:blue;}
</style>
</head>
```

## External Style Sheet

An external style sheet is ideal when the style is applied to many pages. With an external style sheet, you can change the look of an entire Web site by changing one file. Each page must link to the style sheet using the <link> tag. The <link> tag goes inside the <head> section:

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

## Deprecated Tags and Attributes

In HTML 4, several tags and attributes were used to style documents. These tags are not supported in newer versions of HTML.

Avoid using the elements: <font>, <center>, and <strike>, and the attributes: color and bgcolor.

## HTML Images - The <img> Tag and the Src Attribute

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

The <img> tag is empty, which means that it contains attributes only, and has no closing tag.

To display an image on a page, you need to use the src attribute. Src stands for "source". The value of the src attribute is the URL of the image you want to display.

### Syntax for defining an image:

<img src="*url*" alt="*some_text*">

The URL points to the location where the image is stored. An image named "jatin.gif", located in the "images" directory on " www.veinstitution.com" has the URL:
http://www.veinstitution.com/images/jatin.gif

The browser displays the image where the <img> tag occurs in the document. If you put an image tag between two paragraphs, the browser shows the first paragraph, then the image, and then the second paragraph.

## HTML Images - The Alt Attribute

The required alt attribute specifies an alternate text for an image, if the image cannot be displayed.

The value of the alt attribute is an author-defined text:

```
<img src="main.gif" alt="Main">
```

The alt attribute provides alternative information for an image if a user for some reason cannot view it

## HTML Images - Set Height and Width of an Image

The height and width attributes are used to specify the height and width of an image.

The attribute values are specified in pixels by default:

```
<img src="main.gif" alt="Main" width="42" height="42">
```

**Tip:** It is a good to specify both the height and width attributes for an image. If these attributes are set, the space required for the image is reserved when the page is loaded. However, without these attributes, the browser does not know the size of the image. The page layout will change during loading.

## Useful Tips:

**Note: GIF images are usually best for graphics and drawings, while JPEG images are usually better for photographs**. This is for two reasons: first, GIF images only consist of 256 colours, while JPEG images comprise of millions of colours and second, the GIF format is better at compressing simple images, than the JPEG format. The better the compression, the smaller the size of the image file, the faster your page will load. **The PNG format contains in many ways the best of both the JPEG and GIF format: millions of colours and effective compressing**. Use small images.
**Note:** When a web page is loaded, it is the browser, at that moment, that actually gets the image from a web server and inserts it into the page. Therefore, make sure that the images are placed in correct directory in relation to the web page, otherwise your visitors will get a broken link icon.

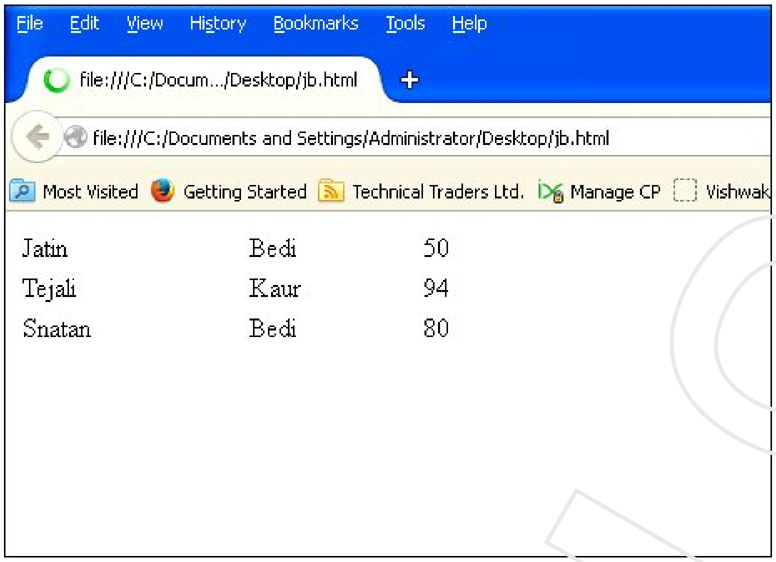
### HTML Tables

Tables are defined with the **<table>** tag. A table is divided into rows with the **<tr>** tag. (tr stands for table row) A row is divided into data cells with the **<td>** tag. (td stands for table data) A row can also be divided into headings with the **<th>** tag. (th stands for table heading) The <td> elements are the data containers in the table. The <td> elements can contain all sorts of HTML elements like text, images, lists, other tables, etc. The width of a table can be defined using CSS.

### Example

```
<!DOCTYPE html>
<html>
<body>

<table style="width:300px">
<tr>
  <td>Jatin</td>
  <td>Bedi</td>
  <td>50</td>
  </tr>
<tr>
  <td>Tejaii</td>
  <td>Kaur</td>
  <td>94</td>
</tr>
<tr>
  <td>Snatan</td>
  <td>Bedi</td>
  <td>80</td>
</tr>
</table>
</body>
</html>
```

## Result


File Edit View History Bookmarks Tools Help
file:///C:/Docum.../Desktop/jb.html
file:///C:/Documents and Settings/Administrator/Desktop/jb.html
Most Visited Getting Started Technical Traders Ltd. Manage CP Vishwak

```
Jatin     Bedi  50
Tejali    Kaur  94
Snatan    Bedi  80
```

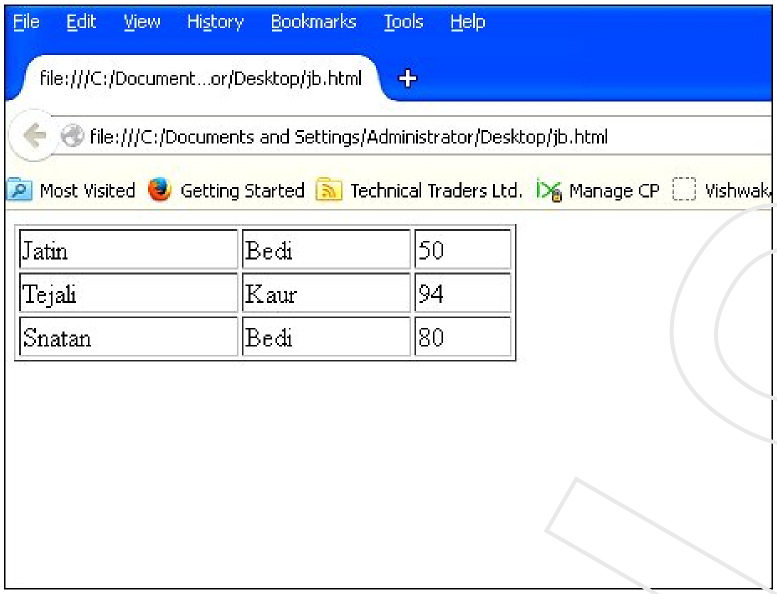
## An HTML Table with a Border Attribute

If you do not specify a border for the table, it will be displayed without borders. A border can be added using the border attribute:

### Example

```
<!DOCTYPE html>
<html>
<body>

<table border="1" style="width:300px">
<tr>
  <td>Jatin</td>
  <td>Bedi</td>
  <td>50</td>
  </tr>
<tr>
  <td>Tejali</td>
  <td>Kaur</td>
  <td>94</td>
</tr>
<tr>
  <td>Snatan</td>
  <td>Bedi</td>
  <td>80</td>
</tr>
</table>

</body>
</html>
```

**Result**


File Edit View History Bookmarks Tools Help
file:///C:/Document...or/Desktop/jb.html
file:///C:/Documents and Settings/Administrator/Desktop/jb.html
Most Visited Getting Started Technical Traders Ltd. Manage CP Vishwak

| | | |
|---|---|---|
| Jatin | Bedi | 50 |
| Tejali | Kaur | 94 |
| Snatan | Bedi | 80 |

**Note:** However, the border attribute is on its way out of the HTML standard! It is better to use CSS.

## To add borders with CSS, use the border property:

### Example

```
<style>
table,th,td
{
border:1px solid black;
}
</style>
```

Remember to define borders for both the table and the table cells.

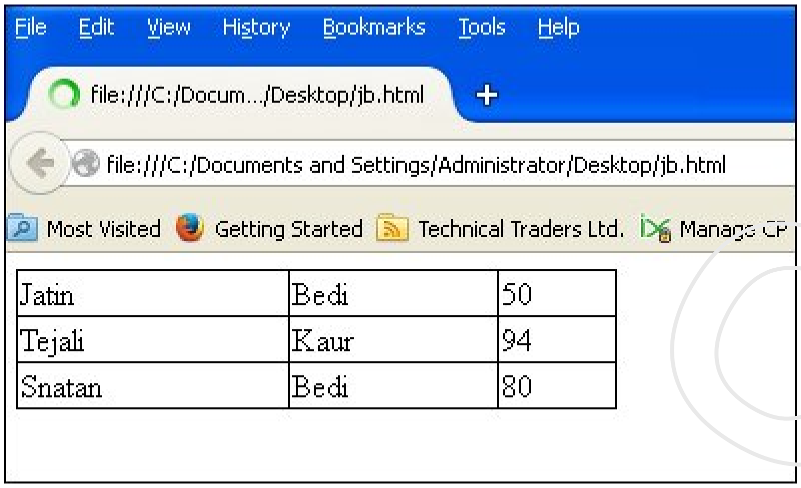
## An HTML Table with Collapsed Borders

If you want the borders to collapse into one border, add border-collapse to your CSS:

### Example

```
<style>
table,th,td
{
border:1px solid black;
border-collapse:collapse
}
</style>
```

**Result**


File Edit View History Bookmarks Tools Help
file:///C:/Docum.../Desktop/jb.html
file:///C:/Documents and Settings/Administrator/Desktop/jb.html
Most Visited Getting Started Technical Traders Ltd. Manage CP

| | | |
|---|---|---|
| Jatin | Bedi | 50 |
| Tejali | Kaur | 94 |
| Snatan | Bedi | 80 |

## An HTML Table with Cell Padding

Cell padding specifies the space between the cell content and its borders.
If you do not specify a padding, the table cells will be displayed without padding.
To set the padding, use the CSS padding property:

**Example**

```
th,td
{
padding:15px;
}
```

**Result**

| | | |
|---|---|---|
| Jatin | Bedi | 50 |
| Tejali | Kaur | 94 |
| Snatan | Bedi | 80 |

## HTML Table Headings

Table headings are defined with the <th> tag.

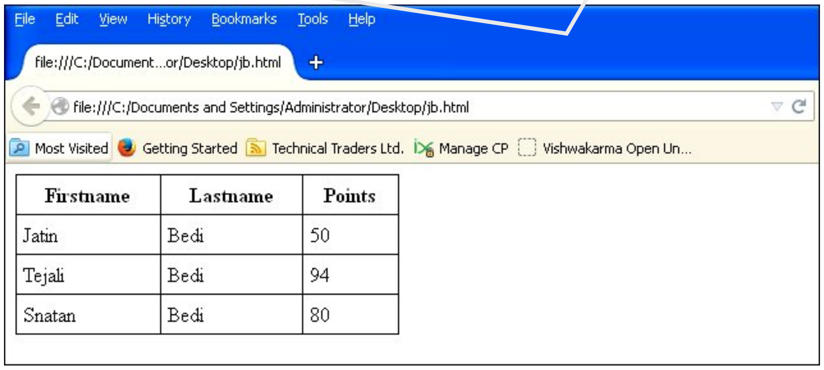
By default, all major browsers display table headings as bold and centered:

**Example**

```
<!DOCTYPE html>
<html>
<head>
<style>
table,th,td
```

```
{
border:1px solid black;
border-collapse:collapse;
}
th,td
{
padding:5px;
}
</style>
</head>
<body>
<table style="width:300px">
<tr>
  <th>Firstname</th>
  <th>Lastname</th>
  <th>Points</th>
  </tr>
<tr>
<tr>
  <td>Jatin</td>
  <td>Bedi</td>
  <td>50</td>
  </tr>
<tr>
  <td>Tejali</td>
  <td>Bedi</td>
  <td>94</td>
</tr>
<tr>
  <td>Snatan</td>
  <td>Bedi</td>
  <td>80</td>
</tr>
</table>
</body>

</html>
```

**Result**


File Edit View History Bookmarks Tools Help
file:///C:/Document...or/Desktop/jb.html
file:///C:/Documents and Settings/Administrator/Desktop/jb.html
Most Visited
Getting Started
Technical Traders Ltd.
Manage CP
Vishwakarma Open Un...

| Firstname | Lastname | Points |
|---|---|---|
| Jatin | Bedi | 50 |
| Tejali | Bedi | 94 |
| Snatan | Bedi | 80 |

To left-align the table headings, use the CSS text-align property:

**Example**

```
th
{
text-align:left;
}
```

## An HTML Table with Cell Spacing

Cell spacing specifies the space between the cells.

To set the cell spacing for the table, use the CSS border-spacing property:

**Example :**

```
table
{
border-spacing:5px;
}
```

HTML Unordered Lists

An unordered list starts with the <ul> tag. Each list item starts with the <li> tag. The list items are marked with bullets (typically small black circles).

```
<ul>
<li>Coffee</li>
<li>Milk</li>
</ul>
```

How the HTML code above looks in a browser:

- Coffee
- Milk

## HTML Ordered Lists

An ordered list starts with the <ol> tag. Each list item starts with the <li> tag.

The list items are marked with numbers.

```
<ol>
<li>Coffee</li>
<li>Milk</li>
</ol>
```

**Result:** How the HTML code above looks in a browser:

1. Coffee
2. Milk

## HTML Description Lists

A description list is a list of terms/names, with a description of each term/name.

The <dl> tag defines a description list.

The <dl> tag is used in conjunction with <dt> (defines terms/names) and <dd> (describes each term/name):

```
<dl>
<dt>Coffee</dt>
<dd>- black hot drink</dd>
<dt>Milk</dt>
<dd>- white cold drink</dd>
</dl>
```

How the HTML code above looks in a browser:

```
Coffee
- black hot drink
Milk
- white cold drink
```

## Useful Tips

**Tip:** Inside a list item you can put text, line breaks, images, links, other lists, etc.

## HTML Block Elements

Most HTML elements are defined as **block level** elements or as **inline** elements.

Block level elements normally start (and end) with a new line when displayed in a browser.

Examples: <h1>, <p>, <ul>, <table>

## HTML Inline Elements

Inline elements are normally displayed without starting a new line.

Examples: <b>, <td>, <a>, <img>

## The HTML <div> Element

The HTML <div> element is a block level element that can be used as a container for grouping other HTML elements.
The <div> element has no special meaning. Except that, because it is a block level element, the browser will display a line break before and after it.
When used together with CSS, the <div> element can be used to set style attributes to large blocks of content.
Another common use of the <div> element, is for document layout. It replaces the "old way" of defining layout using tables. Using <table> elements for layout is not the correct use of <table>. The purpose of the <table> element is to display tabular data.

The HTML <span> Element
The HTML <span> element is an inline element that can be used as a container for text.
The <span> element has no special meaning.
When used together with CSS, the <span> element can be used to set style attributes to parts of the text.

## Website Layouts

Most websites have put their content in multiple columns (formatted like a magazine or newspaper).

Multiple columns are created by using <div> or <table> elements. CSS are used to position elements, or to create backgrounds or colorful look for the pages.

### HTML Layouts - Using <div> Elements

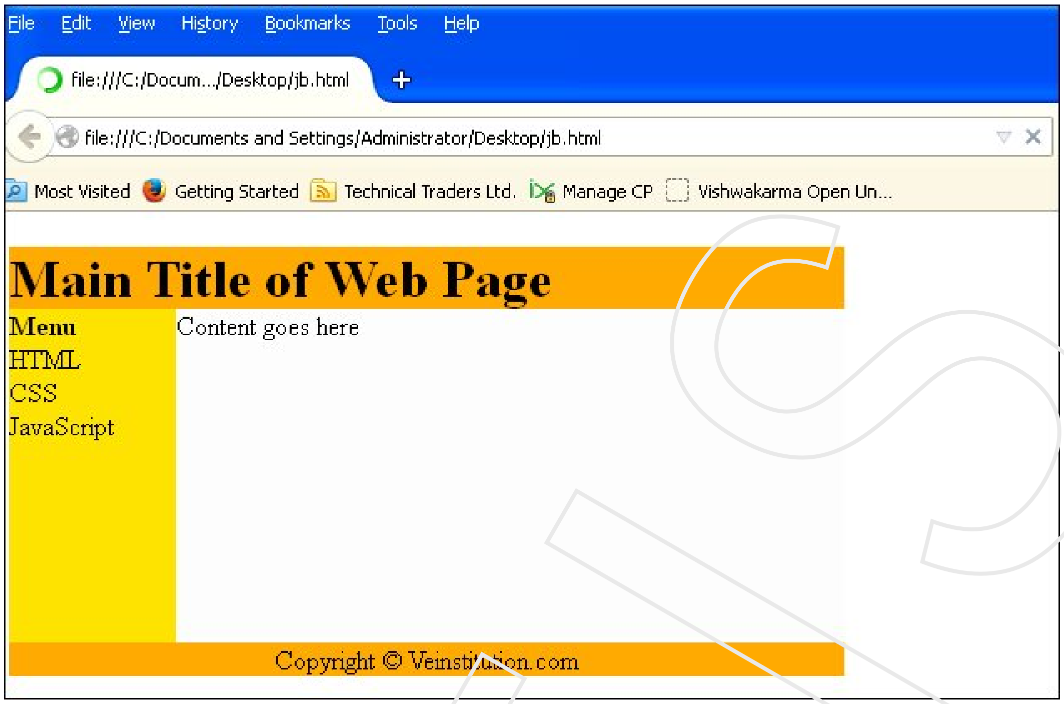
The div element is a block level element used for grouping HTML elements.

The following example uses five div elements to create a multiple column layout, creating the same result as in the previous example:

### Example

```
<!DOCTYPE html>
<html>
<body>
<div id="container" style="width:500px">
<div id="header" style="background-color:#FFA500;">
<h1 style="margin-bottom:0;">Main Title of Web Page</h1></div>
<div id="menu" style="background-color:#FFD700;height:200px;width:100px;float:left;">
<b>Menu</b><br>
HTML<br>
CSS<br>
JavaScript</div>

<div id="content" style="background-color:#EEEEEE;height:200px;width:400px;float:left;">
Content goes here</div>

<div id="footer" style="background-color:#FFA500;clear:both;text-align:center;">
Copyright © Veinstitution.com</div>

</div>

</body>
</html>
```

## Result:


File Edit View History Bookmarks Tools Help
file:///C:/Docum.../Desktop/jb.html
file:///C:/Documents and Settings/Administrator/Desktop/jb.html
Most Visited Getting Started Technical Traders Ltd. Manage CP Vishwakarma Open Un...

Main Title of Web Page

**Menu**
HTML
CSS
JavaScript

Content goes here

Copyright © Veinstitution.com

## HTML Layouts - Using Tables

A simple way of creating layouts is by using the HTML <table> tag.

Multiple columns are created by using <div> or <table> elements. CSS are used to position elements, or to create backgrounds or colorful look for the pages.

**Note:** Using <table> to create a nice layout is NOT the correct use of the element. The purpose of the <table> element is to display tabular data.

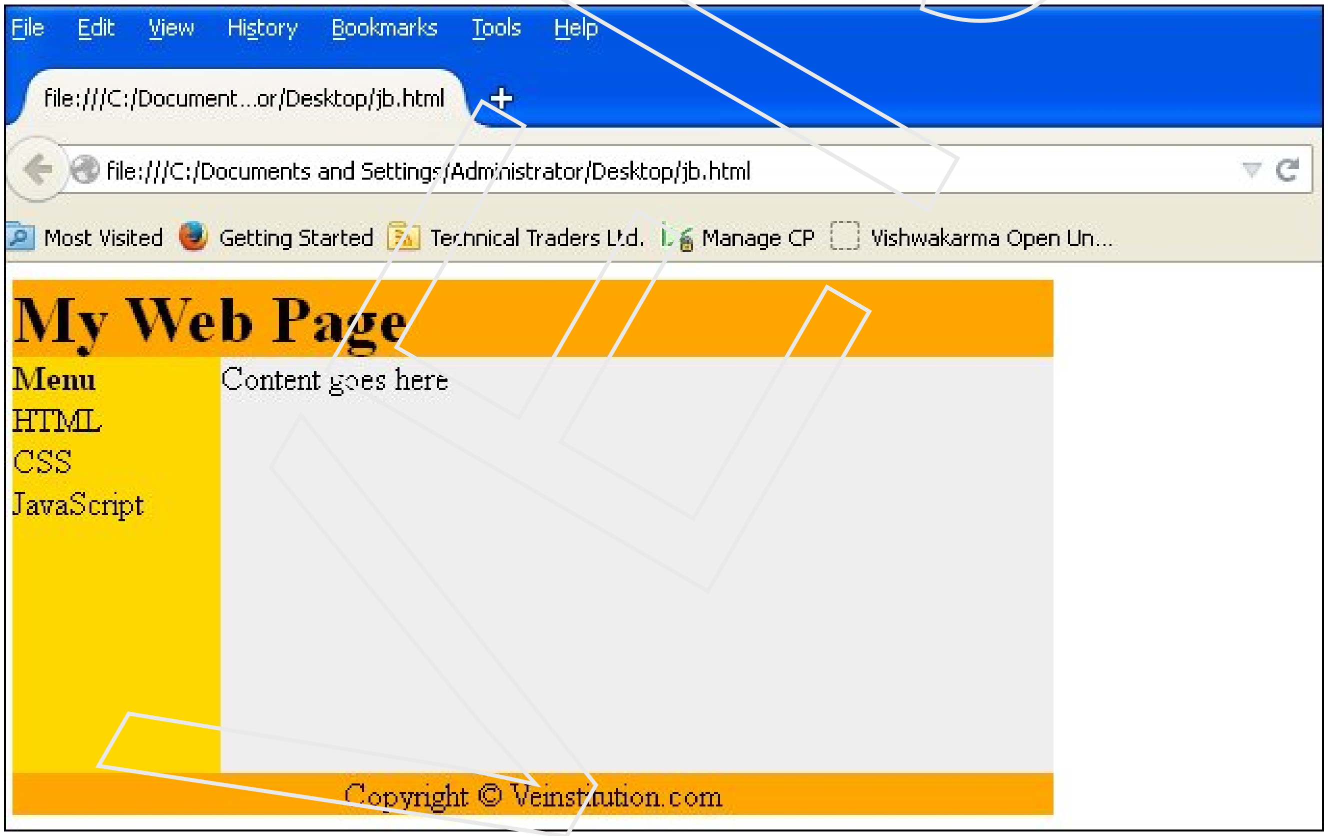
The following example uses a table with 3 rows and 2 columns - the first and last row spans both columns using the colspan attribute:

## Example

```
<!DOCTYPE html>
<html>
<body>
<table style="width:500px;" cellpadding="0" cellspacing="0">
<tr>
<td colspan="2" style="background-color:#FFA500;">
<h1 style="margin:0;padding:0;">My  Web Page</h1>
</td>
</tr>
<tr>
<td style="background-color:#FFD700;width:100px;vertical-align:top;">
<b>Menu</b><br>
```

```
HTML<br>
CSS<br>
JavaScript
</td>
<td style="background-color:#eeeeee;height:200px;width:400px;vertical-align:top;">
Content goes here</td>
</tr>
<tr>
<td colspan="2" style="background-color:#FFA500;text-align:center;">
Copyright © Veinstitution.com</td>
</tr>
</table>
</body>
</html>
```

## Result:


File Edit View History Bookmarks Tools Help
file:///C:/Document...or/Desktop/jb.html
file:///C:/Documents and Settings/Administrator/Desktop/jb.html
Most Visited
Getting Started
Technical Traders Ltd.
Manage CP
Vishwakarma Open Un...

My Web Page

Menu
HTML
CSS
JavaScript

Content goes here

Copyright © Veinstitution.com

## HTML Layout - Useful Tips

**Tip:** The biggest advantage of using CSS is that, if you place the CSS code in an external style sheet, your site becomes much easier to maintain. You can change the layout of all your pages by editing one file.

## HTML Forms

HTML forms are used to pass data to a server.

An HTML form can contain input elements like text fields, checkboxes, radio-buttons, submit buttons and more. A form can also contain select lists, textarea, fieldset, legend, and label elements.

The <form> tag is used to create an HTML form:

<form>
*.input elements*
.</form>

## HTML Forms - The Input Element

The most important form element is the <input> element. The <input> element is used to select user information.
An <input> element can vary in many ways, depending on the type attribute. An <input> element can be of type text field, checkbox, password, radio button, submit button, and more.
The most common input types are described below.

## Text Fields

<input type="text"> defines a one-line input field that a user can enter text into:

```
<form>
First name: <input type="text" name="firstname"><br>
Last name: <input type="text" name="lastname">
</form>
```

### Result:

First name:
Last name:

**Note:** The form itself is not visible. Also note that the default width of a text field is 20 characters.

### Password Field

<input type="password"> defines a password field:

```
<form>
Password: <input type="password" name="pwd">
</form>
```

### Result:

Password:

**Note:** The characters in a password field are masked (shown as asterisks or circles).

## Radio Buttons

<input type="radio"> defines a radio button. Radio buttons let a user select ONLY ONE of a limited number of choices:

```
<form>
<input type="radio" name="sex" value="male">Male<br>
<input type="radio" name="sex" value="female">Female
</form>
```

## Result:

○ Male

○ Female

## Checkboxes

<input type="checkbox"> defines a checkbox. Checkboxes let a user select ZERO or MORE options of a limited number of choices.

```
<form>
<input type="checkbox" name="vehicle" value="Bike">I have a bike<br>
<input type="checkbox" name="vehicle" value="Car">I have a car
</form>
```

### Result:

☐ I have a bike

☐ I have a car

## Submit Button

<input type="submit"> defines a submit button.

A submit button is used to send form data to a server. The data is sent to the page specified in the form's action attribute. The file defined in the action attribute usually does something with the received input:

```
<form name="input" action="demo_form_action.asp" method="get">
Username: <input type="text" name="user">
<input type="submit" value="Submit">
</form>
```

### Result:

Username: ☐

If you type some characters in the text field above, and click the "Submit" button, the browser will send your input to a page called "demo_form_action.asp". The page will show you the received input.

## HTML Iframes

An iframe is used to display a web page within a web page.

**Syntax for adding an iframe:**

<iframe src="*URL*"></iframe>

The URL points to the location of the separate page.

## Iframe - Set Height and Width

The height and width attributes are used to specify the height and width of the iframe.

The attribute values are specified in pixels by default, but they can also be in percent (like "80%").

**Example : <iframe src="demo_iframe.htm" width="200" height="200"></iframe>**

### Iframe - Remove the Border

The frameborder attribute specifies whether or not to display a border around the iframe.

Set the attribute value to "0" to remove the border:

**Example <iframe src="demo_iframe.htm" frameborder="0"></iframe>**

## Use iframe as a Target for a Link

An iframe can be used as the target frame for a link.

The target attribute of a link must refer to the name attribute of the iframe:

### Example

```
<iframe src="demo_iframe.htm" name="iframe_a"></iframe>
<p><a href="http://www.veinstitution.com" target="iframe_a"> veinstitution.com</a></p>
```

## Getting fancy

Say you wanted someone to only have to type in the first few letters of their country instead of having to type in the whole thing. This is what we call an "autocomplete" feature.
Let's do that using a <datalist> element. First, add list attribute to your country input, then add the <datalist> with the same name as its ID . Each line inside the datalist as its own option, showing the countries you want them to get prompted with.

**Form with autocomplete**

```
        <form action="output.htm" method="get">

          Name: <input type="text" name="name"><br>
          Country: <input type="text" list="country" name="countries">
```

```
 <datalist id="country">
  <option value="UK">
  <option value="Canada">
  <option value="USA">
  <option value="India">
  <option value="Brasil">
</datalist>
<br>
<input type="submit" value="Submit">

</form>
```

## Dropdown

```
<select id = "myList">
  <option value = "1">one</option>
  <option value = "2">two</option>
  <option value = "3">three</option>
  <option value = "4">four</option>
</select>
```

## HTML Colors

Colors are displayed combining RED, GREEN, and BLUE light.

## Color Values

CSS colors are defined using a hexadecimal (hex) notation for the combination of Red, Green, and Blue color values (RGB). The lowest value that can be given to one of the light sources is 0 (hex 00). The highest value is 255 (hex FF).

Hex values are written as 3 double digit numbers, starting with a # sign.

## HTML Scripts

JavaScripts make HTML pages more dynamic and interactive.
The HTML <script> Tag
The <script> tag is used to define a client-side script, such as a JavaScript.
The <script> element either contains scripting statements or it points to an external script file through the src attribute.
Common uses for JavaScript are image manipulation, form validation, and dynamic changes of content.
The script below writes Hello World! to the HTML output:

### Example

```
<!DOCTYPE html>
<html>
<body>

<script>
document.write("Hello World!")
```

```
</script>

</body>
</html>
```

## Result :

Hello World!

## The HTML <noscript> Tag

The <noscript> tag is used to provide an alternate content for users that have disabled scripts in their browser or have a browser that doesn't support client-side scripting. The <noscript> element can contain all the elements that you can find inside the <body> element of a normal HTML page.

The content inside the <noscript> element will only be displayed if scripts are not supported, or are disabled in the user's browser:

### Example

```
<!DOCTYPE html>
<html>
<body>
<script>
document.write("Hello World!")
</script>
<noscript>Sorry, your browser does not support JavaScript!</noscript>
<p>A browser without support for JavaScript will show the text in the noscript element.</p>

</body>
</html>
```

**Result**:

```
Hello World!
A browser without support for JavaScript will show the text in the noscript element.
```

### JavaScript examples

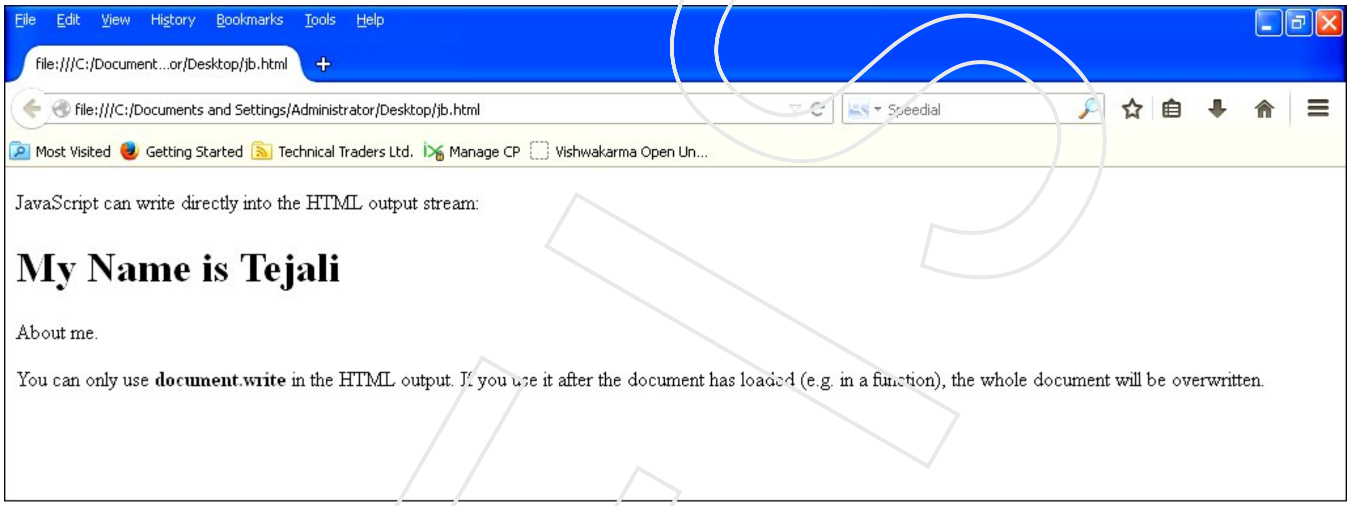
Here are some examples of what JavaScript can do:

**JavaScript can write directly into the HTML output stream:**

```
<!DOCTYPE html>
<html>
<body>

<p>
JavaScript can write directly into the HTML output stream:
</p>

<script>
document.write("<h1>My Name is Tejali</h1>");
document.write("<p>About me.</p>");
</script>
```

```
<p>
You can only use <strong>document.write</strong> in the HTML output.
If you use it after the document has loaded (e.g. in a function), the whole document will be overwritten.
</p>

</body>
</html>
```

**Result**


File Edit View History Bookmarks Tools Help
file:///C:/Document...or/Desktop/jb.html
file:///C:/Documents and Settings/Administrator/Desktop/jb.html
Speedial
Most Visited Getting Started Technical Traders Ltd. Manage CP Vishwakarma Open Un...

JavaScript can write directly into the HTML output stream:

# My Name is Tejali

About me.

You can only use **document.write** in the HTML output. If you use it after the document has loaded (e.g. in a function), the whole document will be overwritten.

## JavaScript can react to events:

```
<!DOCTYPE html>
<html>
<body>

<h1>My First JavaScript</h1>

<p id="demo">
JavaScript can react to events. Like the click of a button.
</p>

<script>
function myFunction()
{
document.getElementById("demo").innerHTML="Hello JavaScript!";
}
</script>

<button type="button" onclick="myFunction()">Click Me!</button>

</body>
</html>
```

**Result**
**My First JavaScript**
JavaScript can react to events. Like the click of a button.
**Click Me**

## JavaScript can manipulate HTML styles:

document.getElementById("demo").style.color="#ff0000";

### HTML Script Tags

| **Tag** | **Description** |
|---|---|
| &lt;script&gt; | Defines a client-side script |
| &lt;noscript&gt; | Defines an alternate content for users that do not support client-side scripts |

### HTML Entities

Some characters are reserved in HTML.

If you use the less than (<) or greater than (>) signs in your text, the browser might mix them with tags.

Character entities are used to display reserved characters in HTML.

A character entity looks like this: *&entity_name*; OR & # *entity_number*;

To display a less than sign we must write: **&lt;** or **&#60;**

Note:The advantage of using an entity name, instead of a number, is that the name is easier to remember.
The disadvantage is that browsers may not support all entity names, but the support for numbers is good.

### Non Breaking Space

A common character entity used in HTML is the non breaking space (&nbsp;).

Remember that browsers will always truncate spaces in HTML pages. If you write 10 spaces in your text, the browser will remove 9 of them. To add real spaces to your text, you can use the **&nbsp;** character entity.

## Combining Diacritical Marks

A diacritical mark is a "glyph" added to a letter.

Some diacritical marks, like grave ( ̀) and acute ( ́) are called accents.

Diacritical marks can appear both above and below a letter, inside a letter, and between two letters.

Diacritical marks can be used in combination with alphanumeric characters, to produce a character that is not present in the character set (encoding) used in the page.

Here are some examples:

| Mark | Character | Construct | Result |
|---|---|---|---|
| ̀ | a | a&#768; | à |
| ́ | a | a&#769; | á |
|  | a | a&#770; | â |
| ̃ | a | a&#771; | ã |
| ̀ | O | O&#768; | Ò |
| ́ | O | O&#769; | Ó |
|  | O | O&#770; | Ô |
| ̃ | O | O&#771; | Õ |

## Some Other Useful HTML Character Entities

Note: Entity names are case sensitive.

| Result | Description | Entity Name | Entity Number |
|---|---|---|---|
|  | non-breaking space | &nbsp; | &#160; |
| < | less than | &lt; | &#60; |
| > | greater than | &gt; | &#62; |
| & | ampersand | &amp; | &#38; |
| ¢ | cent | &cent; | &#162; |
| £ | pound | &pound; | &#163; |
| ¥ | yen | &yen; | &#165; |
| € | euro | &euro; | &#8364; |
| © | copyright | &copy; | &#169; |
| ® | registered trademark | &reg; | &#174; |

### HTML Symbol Entities

HTML entities was described in the previous chapter.

HTML symbols like mathematical operators, arrows, technical symbols and shapes, are not present on a normal keyboard. To add these symbols to an HTML page, you can use the HTML entity name. If no entity name exists, you can use the entity number. If the character does not have an entity name, you can use a decimal (or hexadecimal) reference.

Note: If you use an HTML entity name, or number, the character will always display correctly. This is independent of what character set (encoding) your page uses!

### Example

```
<p>I will display &euro;<p>
<p>I will display &#8364;<p>
<p>I will display &#x20AC;<p>
```

## Some Mathematical Symbols Supported by HTML

| Char | Number Entity | Description |
|---|---|---|
| □ | &#8704; &forall; | FOR ALL |
| ∂ | &#8706; &part; | PARTIAL DEFFERENCIAL |
| □ | &#8707; &exist; | THERE EXISTS |
| □ | &#8709; &empty; | EMPTY SETS |
| □ | &#8711; &nabla; | NABLA |
| □ | &#8712; &isin; | ELEMENT OF |
| □ | &#8713; &notin; | NOT AN ELEMENT OF |
| □ | &#8715; &ni; | CONTAINS A MEMBER |
| ∏ | &#8719; &prod; | N-ARY PRODUCT |
| ∑ | &#8721; &sum; | N-ARY SUMMATION |

## Some Greek Letters Supported by HTML

| Char | Number Entity | Description |
|---|---|---|
| Α | &#913; &Alpha; | GREEK CAPITAL LETTER ALPHA |
| Β | &#914; &Beta; | GREEK CAPITAL LETTER BETA |
| Γ | &#915; &Gamma; | GREEK CAPITAL LETTER GAMMA |
| Δ | &#916; &Delta; | GREEK CAPITAL LETTER DELTA |
| Ε | &#917; &Epsilon; | GREEK CAPITAL LETTER EPSILON |
| Ζ | &#918; &Zeta; | GREEK CAPITAL LETTER ZETA |

## Some Other Entities Supported by HTML

| Char | Number Entity | Description |
|---|---|---|
| © | &#169; &copy; | REGISTERED SIGN |
| ® | &#174; &reg; | REGISTERED SIGN |
| € | &#8364; &euro; | EURO SIGN |
| ™ | &#8482; &trade; | TRADEMARK |
| ← | &#8592; &larr; | LEFTWARDS ARROW |
| ↑ | &#8593; &uarr; | UPWARDS ARROW |
| → | &#8594; &rarr; | RIGHTWARDS ARROW |
| ↓ | &#8595; &darr; | DOWNWARDS ARROW |
| ♠ | &#9824; &spades; | BLACK SPADE SUIT |
| ♣ | &#9827; &clubs; | BLACK CLUB SUIT |
| ♥ | &#9829; &hearts; | BLACK HEART SUIT |
| ♦ | &#9830; &diams; | BLACK DIAMOND SUIT |

## What is Character Encoding?

ASCII was the first **character encoding standard** (also called character set). It define 127 different alphanumeric characters that could be used on the internet.

ASCII supported numbers (0-9), English letters (A-Z), and some special characters like ! $ + - ( ) @ < > .

ANSI (Windows-1252) was the default character set for Windows (up to Windows 95). It supported 256 different codes.

ISO-8859-1, an extension to ASCII, was the default character set for HTML 4. It also supported 256 different codes.

Because ANSI and ISO was too limited, the default character encoding was changed to Unicode (UTF-8) in HTML5.

Unicode covers (almost) all the characters and symbols in the world.

Note: All HTML 4 processors also support UTF-8.

## The HTML charset Attribute

To display an HTML page correctly, a web browser must know the character set used in the page.

This is specified in the <meta> tag:

## For HTML4:

<meta http-equiv="Content-Type" content="text/html;charset=ISO-8859-1">

## For HTML5:

<meta charset="UTF-8">

Note: If a browser detect ISO-8859-1 in a web page, it normally defaults to ANSI, because ANSI is identical to ISO-8859-1 except that ANSI has 32 extra characters.

### HTML Uniform Resource Locators

A URL is another word for a web address.
A URL can be composed of words, such as "veinstitution.com", or an Internet Protocol (IP) address: 192.68.20.50. Most people enter the name of the website when surfing, because names are easier to remember than numbers.
**URL - Uniform Resource Locator**

Web browsers request pages from web servers by using a URL.

When you click on a link in an HTML page, an underlying <a> tag points to an address on the world wide web.

A Uniform Resource Locator (URL) is used to address a document (or other data) on the world wide web.

A web address, like this: http://www.veinstitution/html/default.asp follows these syntax rules:

**scheme://host.domain:port/path/filename**

## Explanation:

- **scheme** - defines the **type** of Internet service. The most common type is **http**
- **host** - defines the **domain host** (the default host for http is **www**)
- **domain** - defines the Internet **domain name**, like veinstitution.com
- **port** - defines the **port number** at the host (the default port number for http is **80**)
- **path** - defines a **path** at the server (If omitted, the document must be stored at the root directory of the web site)
- **filename** - defines the name of a document/resource

## Common URL Schemes

The table below lists some common schemes:

| Scheme | Short for.... | Which pages will the scheme be used for... |
|---|---|---|
| http | HyperText Transfer Protocol | Common web pages starts with http://. Not encrypted |
| https | Secure HyperText Transfer Protocol | Secure web pages. All information exchanged are encrypted |
| ftp | File Transfer Protocol | For downloading or uploading files to a website. Useful for domain maintenance |
| file | | A file on your computer |

## URL Encoding

URLs can only be sent over the Internet using the ASCII character-set.

Since URLs often contain characters outside the ASCII set, the URL has to be converted into a valid ASCII format.

URL encoding converts characters into a format that can be transmitted over the Internet.

URL encoding replaces non ASCII characters with a "%" followed by two hexadecimal digits.

URLs cannot contain spaces. URL encoding normally replaces a space with a + sign.

## HTML and XHTML

## XHTML

- XHTML stands for E**X**tensible **H**yper**T**ext **M**arkup **L**anguage
- XHTML is almost identical to HTML 4.01
- XHTML is a stricter and cleaner version of HTML
- XHTML is HTML defined as an XML application
- XHTML is supported by all major browsers.

## Why XHTML?

Many pages on the internet contain "bad" HTML.

The following HTML code will work fine if you view it in a browser (even if it does NOT follow the HTML rules):

```
<html>
<head>
<title>This is bad HTML</title>
<body>
<h1>Bad HTML
<p>This is a paragraph
</body>
```

XML is a markup language where documents must be marked up correctly and "well-formed".

Today's market consists of different browser technologies. Some browsers run on computers, and some browsers run on mobile phones or other small devices. Smaller devices often lack the resources or power to interpret a "bad" markup language.

Therefore - by combining the strengths of HTML and XML, XHTML was developed. XHTML is HTML redesigned as XML.

## The Most Important Differences from HTML:

### Document Structure

- XHTML DOCTYPE is **mandatory**
- The XML namespace attribute in <html> is **mandatory**
- <html>, <head>, <title>, and <body> is **mandatory**

### XHTML Elements

- XHTML elements must be **properly nested**
- XHTML elements must always be **closed**
- XHTML elements must be in **lowercase**
- XHTML documents must have **one root element**

## XHTML Attributes

- Attribute names must be in **lower case**
- Attribute values must be **quoted**
- Attribute minimization is **forbidden**

## <!DOCTYPE ....> Is Mandatory

An XHTML document must have an XHTML DOCTYPE declaration.

A complete list of all the XHTML Doctypes is found in our HTML Tags Reference.

The <html>, <head>, <title>, and <body> elements must also be present, and the xmlns attribute in <html>, must specify the xml namespace for the document.

The example below shows an XHTML document with a minimum of required tags:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.veinstitution.com/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.veinstitution.com/1999/xhtml">

<head>
<title>Title of document</title>
</head>

<body>
......
</body>

</html>
```

### XHTML Elements Must Be Properly Nested

In HTML, some elements can be improperly nested within each other, like this:

```
<b><i>This text is bold and italic</b></i>
```

In XHTML, all elements must be properly nested within each other, like this:

```
<b><i>This text is bold and italic</i></b>
```

### XHTML Elements Must Always Be Closed

This is wrong:

```
<p>This is a paragraph
<p>This is another paragraph
```

This is correct:

```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```

### Empty Elements Must Also Be Closed

This is wrong:

```
A break: <br>
A horizontal rule: <hr>
An image: <img src="happy.gif" alt="Happy face">
```

This is correct:

```
A break: <br />
A horizontal rule: <hr />
An image: <img src="happy.gif" alt="Happy face" />
```

### XHTML Elements Must Be In Lower Case

This is wrong:

```
<BODY>
<P>This is a paragraph</P>
</BODY>
```

This is correct:

```
<body>
<p>This is a paragraph</p>
</body>
```

## Attribute Names Must Be In Lower Case

This is wrong:

```
<table WIDTH="100%">
```

This is correct:

```
<table width="100%">
```

## Attribute Values Must Be Quoted

This is wrong:

```
<table width=100%>
```

This is correct:

```
<table width="100%">
```

## Attribute Minimization Is Forbidden

This is wrong:

```
<input checked>
<input readonly>
<input disabled>
<option selected>
```

This is correct:

```
<input checked="checked">
<input readonly="readonly">
<input disabled="disabled">
<option selected="selected">
```

## How to Convert from HTML to XHTML

1. Add an XHTML <!DOCTYPE> to the first line of every page
2. Add an xmlns attribute to the html element of every page
3. Change all element names to lowercase
4. Close all empty elements
5. Change all attribute names to lowercase
6. Quote all attribute values

# HTML5

## What is HTML5?

HTML5 is the latest standard for HTML.
The previous version of HTML, HTML 4.01, came in 1999, and the internet has changed significantly since then.
HTML5 was designed to replace both HTML 4, XHTML, and the HTML DOM Level 2.
It was specially designed to deliver rich content without the need for additional plugins. The current version delivers everything from animation to graphics, music to movies, and can also be used to build complicated web applications.
HTML5 is also cross-platform. It is designed to work whether you are using a PC, or a Tablet, a Smartphone, or a Smart TV.

## How Did HTML5 Get Started?

HTML5 is a cooperation between the World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG).
WHATWG was working with web forms and applications, and W3C was working with XHTML 2.0. In 2006, they decided to cooperate and create a new version of HTML.
Some rules for HTML5 were established:
New features should be based on HTML, CSS, DOM, and JavaScript
The need for external plugins (like Flash) should be reduced
Error handling should be easier than in previous versions
Scripting has to be replaced by more markup
HTML5 should be device-independent
The development process should be visible to the public **The HTML5 <!DOCTYPE>**

In HTML5 there is only one DOCTYPE declaration, and it is very simple: <!DOCTYPE html>

## A Minimum HTML5 Document

Below is a simple HTML5 document, with the minimum of required tags:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Title of the document</title>
</head>

<body>
Content of the document......
</body>

</html>
```

## HTML5 - New Features

Some of the most interesting new features in HTML5 are:

- The <canvas> element for 2D drawing
- The <video> and <audio> elements for media playback
- Support for local storage
- New content-specific elements, like <article>, <footer>, <header>, <nav>, <section>
- New form controls, like calendar, date, time, email, url, search

## Browser Support for HTML5

All major browsers (Chrome, Firefox, Internet Explorer, Safari, Opera) support the new HTML5 elements and APIs, and continue to add new HTML5 features to their latest versions.
The HTML 5 working group includes AOL, Apple, Google, IBM, Microsoft, Mozilla, Nokia, Opera, and hundreds of other vendors.

## New Elements in HTML5

The internet, and the use of the internet, has changed a lot since 1999, when HTML 4.01 became a standard.
Today, several elements in HTML 4.01 are obsolete, never used, or not used the way they were intended. All those elements are removed or re-written in HTML5.
To better handle today's internet needs, HTML5 has also included new elements for drawing graphics, displaying media content, for better page structure and better form handling, and several new APIs, such as drag and drop, get the geographical position of a user, store local data, and more.
Below is a list of the new HTML elements, introduced by HTML5, and a description of what they are used for.

## The New <canvas> Element

**Note:** The links in the tables below point to our HTML5 Reference. However, you will learn more about these new elements in this tutorial.

| **Tag** | **Description** |
|---|---|
| <canvas> | Defines graphic drawing using JavaScript |

## New Media Elements

| **Tag** | **Description** |
|---|---|
| <audio> | Defines sound or music content |
| <embed> | Defines containers for external applications (like plug-ins) |
| <source> | Defines sources for <video> and <audio> |
| <track> | Defines tracks for <video> and <audio> |
| <video> | Defines video or movie content |

## New Form Elements

| Tag | Description |
|---|---|
| <datalist> | Defines pre-defined options for input controls |
| <keygen> | Defines a key-pair generator field (for forms) |
| <output> | Defines the result of a calculation |

## New Semantic/Structural Elements

HTML5 offers new elements for better structure:

| Tag | Description |
|---|---|
| <article> | Defines an article in the document |
| <aside> | Defines content aside from the page content |
| <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. |
| <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 (a gauge) |
| <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 |

## Removed Elements

The following HTML 4.01 elements has been removed from HTML5:

- <acronym>
- <applet>
- <basefont>
- <big>
- <center>
- <dir>
- <font>
- <frame>
- <frameset>
- <noframes>
- <strike>
- <tt>

## 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 <img> - Clearly defines its content.

## Browser Support

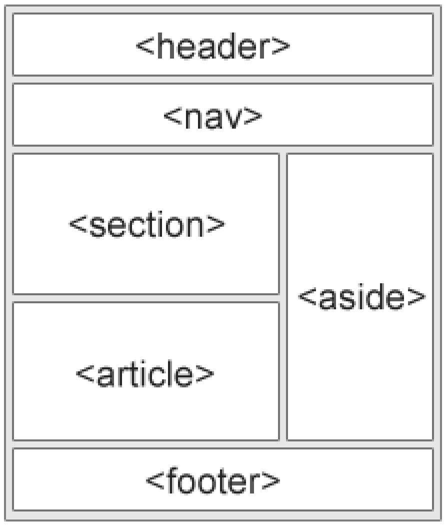
Internet Explorer 9+, Firefox, Chrome, Safari and Opera supports the semantic elements described in this chapter.
**Note:** Internet Explorer 8 and earlier does not support these elements. However, there is a solution. Look at the end of this chapter.

## New Semantic Elements in HTML5

Many of existing web sites today contains HTML code like this: <div id="nav">, <div class="header">, or <div id="footer">, to indicate navigation links, header, and footer.

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

- <header>
- <nav>
- <section>
- <article>
- <aside>
- <figure>
- <figcaption>
- <footer>
- <details>
- <summary>
- <mark>
- <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."

### 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 distribute it independently from the rest of the web site.
Examples of where an <article> element can be used:

- Forum post
- Blog post
- News story
- Comment

### Example

```
<article>
  <h1>Internet Explorer 9</h1>
  <p>Windows Internet Explorer 9 (abbreviated as IE9) was released to
  the  public on March 14, 2011 at 21:00 PDT.....</p>
</article>
```

## HTML5 <nav> Element

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

The <nav> element is intended for large blocks of navigation links. However, not all links in a document should be inside a <nav> element!

### Example

```
<nav>
<a href="/html/">HTML</a> |
<a href="/css/">CSS</a> |
<a href="/js/">JavaScript</a> |
<a href="/jquery/">jQuery</a>
</nav>
```

## HTML5 <aside> Element

The <aside> element defines some content aside from the content it is placed in (like a sidebar).
The aside content should be related to the surrounding content.

### Example

```
<p>My family and I visited The Epcot center this summer.</p>

<aside>
  <h4>Epcot Center</h4>
  <p>The Epcot Center is a theme park in Disney World, Florida.</p>
</aside>
```

## 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>Internet Explorer 9</h1>
    <p><time pubdate datetime="2011-03-15"></time></p>
  </header>
  <p>Windows Internet Explorer 9 (abbreviated as IE9) was released to
  the  public on March 14, 2011 at 21:00 PDT.....</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 can have several <footer> elements in one document.

### Example

```
<footer>
  <p>Posted by: Hege Refsnes</p>
  <p><time pubdate datetime="2012-03-01"></time></p>
</footer>
```

## HTML5 <figure> and <figcaption> Elements

The <figure> tag specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
While the content of the <figure> element is related to the main flow, its position is independent of the main flow, and if removed it should not affect the flow of the document.
The <figcaption> tag defines a caption for a <figure> element.
The <figcaption> element can be placed as the first or last child of the <figure> element.

### Example

```
<figure>
  <img src="img_pulpit.jpg" alt="The Pulpit Rock" width="304" height="228">
  <figcaption>Fig1. - The Pulpit Rock, Norway.</figcaption>
</figure>
```

## HTML5 New Input Types

HTML5 has several new input types for forms. These new features allow better input control and validation. This chapter covers the new input types:

- color
- date
- datetime
- datetime-local
- email
- month
- number
- range
- search
- tel
- time
- url
- week

## Input Type: color

The color type is used for input fields that should contain a color.

### Example

Select a color from a color picker: Select your favorite color: <input type="color" name="favcolor">

## Input Type: date

The date type allows the user to select a date.

### Example

Define a date control:  Birthday: <input type="date" name="bday">

## Input Type: datetime

The datetime type allows the user to select a date and time

### Example

Define a date and time control (with time zone):

Birthday (date and time): <input type="datetime" name="bdaytime">

## Input Type: datetime-local

The datetime-local type allows the user to select a date and time (no time zone).

### Example

Define a date and time control (no time zone):

Birthday (date and time): <input type="datetime-local" name="bdaytime">

## Input Type: email

The email type is used for input fields that should contain an e-mail address.

### Example

Define a field for an e-mail address (will be automatically validated when submitted):

E-mail: <input type="email" name="email">

**Tip:** Safari on iPhone recognizes the email type, and changes the on-screen keyboard to match it (adds @ and .com options).

## Input Type: month

The month type allows the user to select a month and year.

### Example

Define a month and year control (no time zone):

Birthday (month and year): <input type="month" name="bdaymonth">

## Input Type: number

The number type is used for input fields that should contain a numeric value.

You can also set restrictions on what numbers are accepted:

### Example

Define a numeric field (with restrictions):

Quantity (between 1 and 5): <input type="number" name="quantity" min="1" max="5">

Use the following attributes to specify restrictions:

- max - specifies the maximum value allowed
- min - specifies the minimum value allowed
- step - specifies the legal number intervals
- value - Specifies the default value

## Input Type: range

The range type is used for input fields that should contain a value from a range of numbers.
You can also set restrictions on what numbers are accepted.

### Example

Define a control for entering a number whose exact value is not important (like a slider control):

<input type="range" name="points" min="1" max="10">

Use the following attributes to specify restrictions:

- max - specifies the maximum value allowed
- min - specifies the minimum value allowed
- step - specifies the legal number intervals
- value - Specifies the default value

## Input Type: search

The search type is used for search fields (a search field behaves like a regular text field).

### Example

Define a search field (like a site search, or Google search):

Search Google: <input type="search" name="googlesearch">

## Input Type: tel

The tel type is used for input fields that should contain a telephone number.

### Example

Define a field for entering a telephone number:

Telephone: <input type="tel" name="usrtel">

## Input Type: time

The time type allows the user to select a time.

### Example

Define a control for entering a time (no time zone):

Select a time: <input type="time" name="usr_time">

## Input Type: url

The url type is used for input fields that should contain a URL address.
The value of the url field is automatically validated when the form is submitted.

### Example

Define a field for entering a URL: Add your homepage: <input type="url" name="homepage">
**Tip:** Safari on iPhone recognizes the url input type, and changes the on-screen keyboard to match it (adds .com option).

## Input Type: week

The week type allows the user to select a week and year.

### Example

Define a week and year control (no time zone): Select a week: <input type="week" name="week_year">

## HTML5 <input> Tag

| Tag | Description |
|---|---|
| <input> | Defines an input control |

## HTML5 New Form Attributes

HTML5 has several new attributes for <form> and <input>.

New attributes for <form>:

- autocomplete
- novalidate

New attributes for <input>:

- autocomplete
- autofocus
- form
- formaction
- formenctype
- formmethod
- formnovalidate
- formtarget
- height and width
- list
- min and max
- multiple
- pattern (regexp)
- placeholder
- required
- step

## <form> / <input> autocomplete Attribute

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.
**Tip:** It is possible to have autocomplete "on" for the form, and "off" for specific input fields, or vice versa.
**Note:** The autocomplete attribute works with <form> and the following <input> types: text, search, url, tel, email, password, datepickers, range, and color.

### Example

An HTML form with autocomplete on (and off for one input field):

```
<form action="demo_form.asp" autocomplete="on">
  First name:<input type="text" name="fname"><br>
  Last name: <input type="text" name="lname"><br>
  E-mail: <input type="email" name="email" autocomplete="off"><br>
  <input type="submit">
</form>
```

**Tip:** In some browsers you may need to activate the autocomplete function for this to work.

## <form> novalidate Attribute

The novalidate attribute is a boolean attribute.

When present, it specifies that the form-data (input) should not be validated when submitted.

### Example

Indicates that the form is not to be validated on submit:

```
<form action="demo_form.asp" novalidate>
  E-mail: <input type="email" name="user_email">
  <input type="submit">
</form>
```

## <input> autofocus Attribute

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

### Example

Let the "First name" input field automatically get focus when the page loads:

```
First name:<input type="text" name="fname" autofocus>
```

## <input> form Attribute

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

**Tip:** To refer to more than one form, use a space-separated list of form ids.

### Example

An input field located outside the HTML form (but still a part of the form):

```
<form action="demo_form.asp" id="form1">
  First name: <input type="text" name="fname"><br>
  <input type="submit" value="Submit">
</form>

Last name: <input type="text" name="lname" form="form1">
```

## <input> formaction Attribute

The formaction attribute specifies the URL of a file that will process the input control when the form is submitted.

The formaction attribute overrides the action attribute of the <form> element.

**Note:** The formaction attribute is used with type="submit" and type="image".

### Example

An HTML form with two submit buttons, with different actions:

```
<form action="demo_form.asp">
  First name: <input type="text" name="fname"><br>
  Last name: <input type="text" name="lname"><br>
  <input type="submit" value="Submit"><br>
  <input type="submit" formaction="demo_admin.asp"
  value="Submit as admin">
</form>
```

## <input> formenctype Attribute

The formenctype attribute specifies how the form-data should be encoded when submitting it to the server (only for forms with method="post")

The formenctype attribute overrides the enctype attribute of the <form> element.

**Note:** The formenctype attribute is used with type="submit" and type="image".

### Example

Send form-data that is default encoded (the first submit button), and encoded as "multipart/form-data" (the second submit button):

```
<form action="demo_post_enctype.asp" method="post">
  First name: <input type="text" name="fname"><br>
  <input type="submit" value="Submit">
  <input type="submit" formenctype="multipart/form-data"
  value="Submit as Multipart/form-data">
</form>
```

## <input> formmethod Attribute

The formmethod attribute defines the HTTP method for sending form-data to the action URL.

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

**Note:** The formmethod attribute can be used with type="submit" and type="image".

**Example**

The second submit button overrides the HTTP method of the form:

```
<form action="demo_form.asp" method="get">
  First name: <input type="text" name="fname"><br>
  Last name: <input type="text" name="lname"><br>
  <input type="submit" value="Submit">
  <input type="submit" formmethod="post" formaction="demo_post.asp"
  value="Submit using POST">
</form>
```

## <input> formnovalidate Attribute

The novalidate attribute is a boolean attribute.

When present, it specifies that the <input> element should not be validated when submitted.

The formnovalidate attribute overrides the novalidate attribute of the <form> element.

**Note:** The formnovalidate attribute can be used with type="submit".

**Example** A form with two submit buttons (with and without validation):

```
<form action="demo_form.asp">
  E-mail: <input type="email" name="userid"><br>
  <input type="submit" value="Submit"><br>
  <input type="submit" formnovalidate value="Submit without validation">
</form>
```

## <input> formtarget Attribute

The formtarget attribute specifies a name or a keyword that indicates where to display the response that is received after submitting the form.

The formtarget attribute overrides the target attribute of the <form> element.

**Note:** The formtarget attribute can be used with type="submit" and type="image".

**Example** A form with two submit buttons, with different target windows:

```
<form action="demo_form.asp">
  First name: <input type="text" name="fname"><br>
  Last name: <input type="text" name="lname"><br>
  <input type="submit" value="Submit as normal">
  <input type="submit" formtarget="_blank"
  value="Submit to a new window">
</form>
```

## <input> height and width Attributes

The height and width attributes specify the height and width of an <input> element.

**Note:** The height and width attributes are only used with <input type="image">.

**Tip:** Always specify both the height and width attributes for images. If height and width are set, the space required for the image is reserved when the page is loaded. However, without these attributes, the browser does not know the size of the image, and cannot reserve the appropriate space to it.

**Example** Define an image as the submit button, with height and width attributes:

```
<input type="image" src="img_submit.gif" alt="Submit" width="48" height="48">
```

## <input> list Attribute

The list attribute refers to a <datalist> element that contains pre-defined options for an <input> element.

### Example

An <input> element with pre-defined values in a <datalist>:

```
<input list="browsers">

<datalist id="browsers">
  <option value="Internet Explorer">
  <option value="Firefox">
  <option value="Chrome">
  <option value="Opera">
  <option value="Safari">
</datalist>
```

## <input> min and max Attributes

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

**Note:** The min and max attributes works with the following input types: number, range, date, datetime, datetime-local, month, time and week.

### Example

<input> elements with min and max values:

```
Enter a date before 1980-01-01:
<input type="date" name="bday" max="1979-12-31">

Enter a date after 2000-01-01:
<input type="date" name="bday" min="2000-01-02">
```

Quantity (between 1 and 5):
<input type="number" name="quantity" min="1" max="5">

## <input> multiple Attribute

The multiple attribute is a boolean attribute.

When present, it specifies that the user is allowed to enter more than one value in the <input> element.

**Note:** The multiple attribute works with the following input types: email, and file.

### Example

A file upload field that accepts multiple values:

Select images: <input type="file" name="img" multiple>

## <input> pattern Attribute

The pattern attribute specifies a regular expression that the <input> element's value is checked against.

**Note:** The pattern attribute works with the following input types: text, search, url, tel, email, and password.

**Tip:** Use the global title attribute to describe the pattern to help the user.

### Example

An input field that can contain only three letters (no numbers or special characters):

Country code: <input type="text" name="country_code" pattern="[A-Za-z]{3}" title="Three letter country code">

## <input> placeholder Attribute

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). The short hint is displayed in the input field before the user enters a value.

**Note:** The placeholder attribute works with the following input types: text, search, url, tel, email, and password.

### Example

An input field with a placeholder text:

<input type="text" name="fname" placeholder="First name">

## <input> required Attribute

The required attribute is a boolean attribute.
When present, it specifies that an input field must be filled out before submitting the form.
**Note:** The required attribute works with the following input types: text, search, url, tel, email, password, date pickers, number, checkbox, radio, and file.

### Example

A required input field:

Username: <input type="text" name="usrname" required>

## <input> step Attribute

The step attribute specifies the legal number intervals for an <input> element.
Example: if step="3", legal numbers could be -3, 0, 3, 6, etc.
**Tip:** The step attribute can be used together with the max and min attributes to create a range of legal values.
**Note:** The step attribute works with the following input types: number, range, date, datetime, datetime-local, month, time and week.

### Example

An input field with a specified legal number intervals:

<input type="number" name="points" step="3">

## HTML5 <input> Tag

| Tag | Description |
|---|---|
| <form> | Defines an HTML form for user input |
| <input> | Defines an input control |

## What is an API?

An API is an application programming interface, which is just a fancy way of saying that it's a way to send instructions between programs. In this case, the instructions are between your web page and the browser to, for example, show a Google Map or offer a full screen view of your page.
Generally speaking, APIs are a way for you to offer more interactivity into your page.
Before HTML5, most APIs were written with JavaScript, an entirely different language. With HTML5, you can now add interactivity without having to always write JavaScript.

## What kinds of APIs are there?

There are a lot of APIs you can use in HTML5, too many to cover in this tutorial, but we'll cover the big ones.

1. Drawing: You can let people draw on your web page using the <canvas> tag. However, the canvas tag is just a holder… this one still needs JavaScript to actually draw.
2. Audio/Video: You can now add a video right into your web page without having to embed a player or use You Tube. You can even add play/pause and other controls.
3. Drag and drop: You can allow people to move things around on your page.
4. Auto focus: Focusses the page on a specific item by moving the cursor there.
5. Editable: You can make content editable. We mentioned this briefly in Lesson 16.
6. History: You can add controls for going back or forward to specific pages or to relative pages (the page you were at before this one, for example).

There are a lot more HTML5 APIs and most of them require some knowledge or interaction with JavaScript as well.

## Autofocus

The simplest API is autofocus. When the page loads, it takes your reader right to that spot. Let's try it on our forms page. Locate the country input box on your forms.htm page and add the autofocus="autofocus" attribute.

**Autofocus**

```
<input autofocus="autofocus" type="text" list="country" name="countries">
```

Save the file and try it out! Your cursor should be right inside the "Country" text box and the box should be highlighted.

## What is Canvas?

The HTML5 <canvas> element is used to draw graphics, on the fly, via scripting (usually JavaScript).
The <canvas> element is only a container for graphics. You must use a script to actually draw the graphics.
Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

## Browser Support

Internet Explorer 9+, Firefox, Opera, Chrome, and Safari support the <canvas> element.
**Note:** Internet Explorer 8 and earlier versions, do not support the <canvas> element.

## Create a Canvas

A canvas is a rectangular area on an HTML page, and it is specified with the <canvas> element.

**Note:** By default, the <canvas> element has no border and no content.

The markup looks like this:

```
<canvas id="myCanvas" width="200" height="100"></canvas>
```

**Note:** Always specify an id attribute (to be referred to in a script), and a width and height attribute to define the size of the canvas.

**Tip:** You can have multiple <canvas> elements on one HTML page. To add a border, use the style attribute:

### Example

```
<canvas id="myCanvas" width="200" height="100"
style="border:1px solid #000000;">
</canvas>
```

## Draw Onto The Canvas With JavaScript

All drawing on the canvas must be done inside a JavaScript:

### Example

```
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.fillStyle = "#FF0000";
ctx.fillRect(0,0,150,75);
</script>
```

### Example explained:

First, find the <canvas> element:

```
var c = document.getElementById("myCanvas");
```

Then, call its getContext() method (you must pass the string "2d" to the getContext() method):

```
var ctx = c.getContext("2d");
```

The getContext("2d") object is a built-in HTML5 object, with many properties and methods for drawing paths, boxes, circles, text, images, and more.

The next two lines draw a red rectangle:

```
ctx.fillStyle = "#FF0000";
ctx.fillRect(0,0,150,75);
```

The fillStyle property can be a CSS color, a gradient, or a pattern. The default fillStyle is #000000 (black).

The fillRect(*x,y,width,height*) method draws a rectangle filled with the current fill style.

## Canvas Coordinates

The canvas is a two-dimensional grid.

The upper-left corner of the canvas has coordinate (0,0)

So, the fillRect() method above had the parameters (0,0,150,75).

This means: Start at the upper-left corner (0,0) and draw a 150x75 pixels rectangle.

**Coordinates Example**

Mouse over the rectangle below to see its x and y coordinates:

X
Y

## Canvas - Paths

To draw straight lines on a canvas, we will use the following two methods:

- moveTo(*x,y*) defines the starting point of the line
- lineTo(*x,y*) defines the ending point of the line

To actually draw the line, we must use one of the "ink" methods, like stroke().

### Example

Define a starting point in position (0,0), and an ending point in position (200,100). Then use the stroke() method to actually draw the line:

JavaScript:

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.moveTo(0,0);
ctx.lineTo(200,100);
ctx.stroke();
```

To draw a circle on a canvas, we will use the following method:

- arc(x,y,r,start,stop)

To actually draw the circle, we must use one of the "ink" methods, like stroke() or fill().

## Example

Create a circle with the arc() method:

**JavaScript:**

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.beginPath();
ctx.arc(95,50,40,0,2*Math.PI);
ctx.stroke();
```

## Canvas - Text

To draw text on a canvas, the most important property and methods are:
font - defines the font properties for text
fillText(text,x,y) - Draws "filled" text on the canvas
strokeText(text,x,y) - Draws text on the canvas (no fill)
Using fillText():

## Example

Write a 30px high filled text on the canvas, using the font "Arial":

JavaScript:

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.fillText("Hello World",10,50);
```

## Using strokeText():

**Example** Write a 30px high text (no fill) on the canvas, using the font "Arial":

JavaScript:

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.strokeText("Hello World",10,50);
```

## Canvas - Gradients

Gradients can be used to fill rectangles, circles, lines, text, etc. Shapes on the canvas are not limited to solid colors.

There are two different types of gradients:

- createLinearGradient(*x,y,x1,y1*) - Creates a linear gradient
- createRadialGradient(*x,y,r,x1,y1,r1*) - Creates a radial/circular gradient

Once we have a gradient object, we must add two or more color stops.

The addColorStop() method specifies the color stops, and its position along the gradient. Gradient positions can be anywhere between 0 to 1.

To use the gradient, set the fillStyle or strokeStyle property to the gradient, and then draw the shape, like a rectangle, text, or a line.

Using createLinearGradient():

**Example** Create a linear gradient. Fill rectangle with the gradient:

JavaScript:

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");

// Create gradient
var grd = ctx.createLinearGradient(0,0,200,0);
grd.addColorStop(0,"red");
grd.addColorStop(1,"white");

// Fill with gradient
ctx.fillStyle = grd;
ctx.fillRect(10,10,150,80);
```

Using createRadialGradient():

### Example

Create a radial/circular gradient. Fill rectangle with the gradient:

JavaScript:

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");

// Create gradient
var grd = ctx.createRadialGradient(75,50,5,90,60,100);
grd.addColorStop(0,"red");
grd.addColorStop(1,"white");

// Fill with gradient
ctx.fillStyle = grd;
ctx.fillRect(10,10,150,80);
```

## Canvas - Images

To draw an image on a canvas, we will use the following method:

- drawImage(*image,x,y*)

## Image to use:

### Example

Draw the image onto the canvas:

JavaScript:

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
var img = document.getElementById("scream");
ctx.drawImage(img,10,10);
```

## The HTML <canvas> Tag

| Tag | Description |
|---|---|
| <canvas> | Used to draw graphics, on the fly, via scripting (usually JavaScript) |

# What is SVG?

- SVG stands for Scalable Vector Graphics
- SVG is used to define vector-based graphics for the Web
- SVG defines the graphics in XML format
- SVG graphics do NOT lose any quality if they are zoomed or resized
- Every element and every attribute in SVG files can be animated
- SVG is a W3C recommendation

## SVG Advantages

Advantages of using SVG over other image formats (like JPEG and GIF) are:

- SVG images can be created and edited with any text editor
- SVG images can be searched, indexed, scripted, and compressed
- SVG images are scalable
- SVG images can be printed with high quality at any resolution
- SVG images are zoomable (and the image can be zoomed without degradation)

## Browser Support

Internet Explorer 9+, Firefox, Opera, Chrome, and Safari support inline SVG.
Embed SVG Directly Into HTML Pages
In HTML5, you can embed SVG elements directly into your HTML page:

**Example**

```
<!DOCTYPE html>
<html>
<body>

<svg width="300" height="200">
  <polygon points="100,10 40,180 190,60 10,60 160,180"
  style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;" />
</svg>

</body>
</html>
```

## Differences Between SVG and Canvas

SVG is a language for describing 2D graphics in XML.

Canvas draws 2D graphics, on the fly (with a JavaScript).

SVG is XML based, which means that every element is available within the SVG DOM. You can attach JavaScript event handlers for an element.

In SVG, each drawn shape is remembered as an object. If attributes of an SVG object are changed, the browser can automatically re-render the shape.

Canvas is rendered pixel by pixel. In canvas, once the graphic is drawn, it is forgotten by the browser. If its position should be changed, the entire scene needs to be redrawn, including any objects that might have been covered by the graphic.

## Comparison of Canvas and SVG

The table below shows some important differences between Canvas and SVG:

**Canvas**

- Resolution dependent
- No support for event handlers
- Poor text rendering capabilities
- You can save the resulting image as .png or .jpg
- Well suited for graphic-intensive games

**SVG**

- Resolution independent
- Support for event handlers
- Best suited for applications with large rendering areas (Google Maps)
- Slow rendering if complex (anything that uses the DOM a lot will be slow)
- Not suited for game applications

### Video on the Web

Before HTML5, there was no standard for showing videos/movies on web pages.
Before HTML5, videos could only be played with a plug-in (like flash). However, different browsers supported different plug-ins.
HTML5 defines a new element which specifies a standard way to embed a video or movie on a web page: the <video> element.

### Browser Support

Internet Explorer 9+, Firefox, Opera, Chrome, and Safari support the <video> element.

**Note:** Internet Explorer 8 and earlier versions, do not support the <video> element.

**HTML5 Video - How It Works** To show a video in HTML5, this is all you need:

### Example

```
<video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
Your browser does not support the video tag.
</video>
```

The control attribute adds video controls, like play, pause, and volume.
It is also a good idea to always include width and height attributes. If height and width are set, the space required for the video is reserved when the page is loaded. However, without these attributes, the browser does not know the size of the video, and cannot reserve the appropriate space to it.
You should also insert text content between the <video> and </video> tags for browsers that do not support the <video> element.
The <video> element allows multiple <source> elements. <source> elements can link to different video files. The browser will use the first recognized format.

## Video Formats and Browser Support

Currently, there are 3 supported video formats for the <video> element: MP4, WebM, and Ogg:

- MP4 = MPEG 4 files with H264 video codec and AAC audio codec
- WebM = WebM files with VP8 video codec and Vorbis audio codec
- Ogg = Ogg files with Theora video codec and Vorbis audio codec

## MIME Types for Video Formats

| **Format** | **MIME-type** |
|---|---|
| MP4 | video/mp4 |
| WebM | video/webm |
| Ogg | video/ogg |

## HTML5 <video> - DOM Methods and Properties

HTML5 has DOM methods, properties, and events for the <video> and <audio> elements.
These methods, properties, and events allow you to manipulate <video> and <audio> elements using JavaScript.
There are methods for playing, pausing, and loading, for example and there are properties (like duration and volume). There are also DOM events that can notify you when the <video> element begins to play, is paused, is ended, etc.

## HTML5 Video Tags

| **Tag** | **Description** |
|---|---|
| <video> | Defines a video or movie |
| <source> | Defines multiple media resources for media elements, such as <video> and <audio> |
| <track> | Defines text tracks in media players |

## Audio on the Web

Before HTML5, there was no standard for playing audio files on a web page.
Before HTML5, audio files had to be played with a plug-in (like flash). However, different browsers supported different plug-ins.
HTML5 defines a new element which specifies a standard way to embed an audio file on a web page: the <audio> element.

## Browser Support

Internet Explorer 9+, Firefox, Opera, Chrome, and Safari support the <audio> element.
**Note:** Internet Explorer 8 and earlier versions, do not support the <audio> element.

## HTML5 Audio - How It Works

To play an audio file in HTML5, this is all you need:

**Example**

```
<audio controls>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

The control attribute adds audio controls, like play, pause, and volume.

You should also insert text content between the <audio> and </audio> tags for browsers that do not support the <audio> element.

The <audio> element allows multiple <source> elements. <source> elements can link to different audio files. The browser will use the first recognized format.

## Audio Formats and Browser Support

Currently, there are 3 supported file formats for the <audio> element: MP3, Wav, and Ogg:

## MIME Types for Audio Formats

| Format | MIME-type |
|---|---|
| MP3 | audio/mpeg |
| Ogg | audio/ogg |
| Wav | audio/wav |

## HTML5 Audio Tags

| Tag | Description |
|---|---|
| <audio> | Defines sound content |
| <source> | Defines multiple media resources for media elements, such as <video> and <audio> |

## What is Multimedia?

Multimedia comes in many different formats. It can be almost anything you can hear or see.
Examples: Pictures, music, sound, videos, records, films, animations, and more.
Modern web pages often have embedded multimedia elements, and modern browsers have support for various multimedia formats.
In this tutorial you will learn about the different multimedia formats.

## Internet Browser Support

The first Internet browsers had support for text only, and even the text support was limited to a single font in a single color. Then came browsers with support for colors, fonts and text styles, and support for pictures was also added.

The support for sounds, animations, and videos is handled in different ways by various browsers. Some multimedia elements is supported, and some requires an extra helper program (a plug-in) to work.

## Multimedia Formats

Multimedia elements (like sounds or videos) are stored in media files.
The most common way to discover the type of a file, is to look at the file extension. When a browser sees the file extension .htm or .html, it will treat the file as an HTML file. The .xml extension indicates an XML file, and the .css extension indicates a style sheet file. Pictures are recognized by extensions like .gif, .png and .jpg.
Multimedia files also have their own formats and different extensions
like: .swf, .wav, .mp3, .mp4, .mpg, .wmv, and .avi.

## Video Formats

| |
|---|
| MP4 is the new and upcoming format for internet video.<br>MP4 is recommended by YouTube.<br>MP4 is supported by Flash players and HTML5. |

## 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. Examples of well-known plug-ins are Java applets and Adobe Flash Player. Plug-ins can be added to web pages with the <object> tag or the <embed> tag. Plug-ins can be used for many purposes: to display maps, scan for viruses, verify your bank id, and much more. The restrictions are few.

## What is The Best Way to Play Audio or Video in HTML?

The best way to embed audio in a web page is to use the HTML5 <audio> element.

The best way to embed video in a web page is to use the HTML5 <video> element.

## The <object> Element

The <object> element is supported in all major browsers.

The <object> element defines an embedded object within an HTML document.

It is used to to embed plug-ins (like Java applets, ActiveX, PDF, and Flash) in web pages.

It can also be used to embed another webpage, or web content like images, into HTML documents.

### Example

```
<object width="400" height="50" data="bookmark.swf"></object>
```

## The <embed> Element

The <embed> element is supported in all major browsers.
The <embed> element defines a container for an external application or interactive content (a plug-in).
Many web browsers have supported the <embed> element for a long time. However, it has not been a part of the HTML specification before HTML5.
The <embed> element will validate in an HTML5 page, but not in an HTML 4 page.

### Example

```
<embed width="400" height="50" src="bookmark.swf">
```

Note: that the <embed> element does not have a closing tag. It can not contain alternative text.

## Playing a YouTube Video in HTML

If you want to play a video in a web page, you can upload the video to YouTube and insert the proper HTML code to display the video:

### Example - YouTube iFrame

```
<iframe width="420" height="345"
src="http://www.youtube.com/embed/YGSx2_Czz4k">
</iframe>
```

## Advanced APIs

### HTML5 and JavaScript

HTML5 gives you the ability to include some advanced features and interactions that can really add some interesting features to your website.
The challenge is that none of them is pure HTML5 code. They all require a combination of HTML and JavaScript, which is a more advanced coding language. I know, I know, *another language*!
HTML, CSS, and JavaScript often all work together in coordination to give us some great websites.
When you get to the more advanced HTML functionality, you need to start learning more. But we'll make this as painless as possible. Now that you understand HTML, it's much easier to learn JavaScript.

### Introduction to JavaScript

JavaScript *functions* are the backbone of JavaScript. They connect your HTML elements with actually doing something. A function looks like this:

```
        function functionName()
```

A function always has some sort of code following it that defines what action should occur. The code is between curly brackets { } and usually ends with a semicolon.
Functions always occur inside <script> elements. And <script> elements are either put in your <head> or right before the </body> element. If you have a lot of JavaScript running on a page, you should put them as low as you can on the page so they won't affect the speed of loading your page. For now, you can just put them in the <head> element.

A full JavaScript script looks like this:

```
<head>
  <script>
    function functionName()
    {
    some code;
    }
  </script>
</head>
```

In your HTML, you "call" your script by adding its name and event to an element, like a button. Events could be, for example, onclick or onhover where somthing happens (the event) when you click or hover the element, respectively. There are a number of other events that you'll learn about in the JavaScript tutorial.

## Calling a script

```
<button onclick="functionName()">Go!</button>
```

JavaScript can do all sorts of things, but one of the most important is to be able to use some logic that says "If...then...else". So "If you're browser supports geolocation, then show coordinates. Else, show this error message."

## If, then, else example

```
if (navigator.geolocation)
  {
  navigator.geolocation.getCurrentPosition(showPosition);
  }
else
  {
  x.innerHTML="Geolocation is not supported by this browser.";
  }
```

## Geolocation

Geolcation is an API that returns and can display your physical location in the world. It can be displayed as coordinates (latitude and longitude) or using a map. Please notice how your browser will prompt you before allowing trusted sites to run this API. Privacy is always a concern, so you can't force anyone to use this API.

Geolocation is built from two things:

1. A button that invokes the script to get coordinates.
2. A script that actually fetches and displays the coordinates (or displays a message if the browser doesn't support it).

The function for geolocation is called getCurrentPosition(). We call the function using getLocation() both in the button element and in the script that follows.

Here's the full code script for a page with simple geolocation. We will then walk through it piece by piece.

**HTML:**

```
<p id="veinstitution.com">Where are you in the world?</p>
<button onclick="getLocation()">Get Coordinates</button>
```

**Javascript:**

```
<script>

  function getLocation()
    {
    if (navigator.geolocation)
      {
      navigator.geolocation.getCurrentPosition(showPosition);
      }
    else
          {
          document.getElementById("veinstitution.com").innerHTML="Geolocation is not
supported by this browser.";
          }
    }

  function showPosition(position)
    {
    document.getElementById("veinstitution.com").innerHTML="Latitude: " +
position.coords.latitude +
    "<br>Longitude: " + position.coords.longitude;
    }

</script>
```

First, let's take a look at the HTML. It includes an ID that is later fetched in the script (id="veinstitution.com") as well as an event on the button and the name of the function we are calling: getLocation(). This code might be anywhere on your page.

Next, let's walk through the Javascript. We start the script element (remember, this is either placed in the <head> element or before the </body> tag). Thereafter, we invoke the geolocation function itself.

```
function getLocation()
```

After that, we cover the situation for when geolocation is not supported by the browser version or because they have JavaScript turned off. In that circumstance, the following message will display: Geolocation is not supported by this browser.

```
        function getLocation()
          {
          if (navigator.geolocation)
            {
            navigator.geolocation.getCurrentPosition(showPosition);
            }
          else
            {
            document.getElementById("veinstitution.com").innerHTML="Geolocation is not supported
by this browser.";
            }
          }
```

After that, we actually call another function because we don't just want to get the location; we also want to display it to the reader as coordinates.

```
        function showPosition(position)
          {
          document.getElementById("veinstitution.com").innerHTML="Latitude: " +
position.coords.latitude +
          "<br>Longitude: " + position.coords.longitude;
          }
```

The text inside the quotes can be modified. It's what people will see, followed by their actual latitude and longitude.
Then you end your script with </script> and it's all complete.

## Show using Google Maps

If you want to show a map of the coordinates, you need to add replace the script above with a connector to the map generator you want, such as Google Maps.

```
        function showPosition(position)
          {
          var latlon=position.coords.latitude+","+position.coords.longitude;
          var img_url="http://maps.googleapis.com/maps/api/staticmap?center="
          +latlon+"&zoom=14&size=400x300&sensor=false";
          document.getElementById("veinstitution.com").innerHTML="<img src='"+img_url+"'>";
          }
```

Copy and paste the full code above into a new page, save it, and upload it to the internet and try it out. Try using the Google Maps option too.

## Web storage

You could use web storage for a number of different purposes:

- User preferences
- Localization (language they use)
- Saving products in a shopping cart (emptied after a session or remembered the next time they visit?)
- Creating a to do list
- Anything else where you want input/choices to persist for a session or forever

Web storage means the end of cookies!
A cookie is a small text file saved on the user's hard drive in which a website can store different information. But HTML5 lets you store data inside the web page instead of using cookies. This makes the web page faster and more secure. You can actually store quite a bit of data in the web page without making the page slow to load.
Below, we've included code that lets people like a page.

**Web storage for "Like this page"**

```
 <head>
<script>
 function clickCounter()
   {
  if(typeof(Storage)!=="undefined")
    {
    if (localStorage.clickcount)
      {
      localStorage.clickcount=Number(localStorage.clickcount)+1;
      }
    else
      {
      localStorage.clickcount=1;
      }
    document.getElementById("result").innerHTML= + localStorage.clickcount + " people
have liked this page.";
    }
  else
    {
    document.getElementById("result").innerHTML="Your browser does not support web
storage.";
    }
  }
</script>
</head>

<body>
  <p><button onclick="clickCounter()" type="button">Like!</button></p>
  <div id="result"></div>
</body>
```

**Details**:

First we call the function clickCounter, which counts the number of times something is clicked.

```
<head>
<script>
function clickCounter()
```

Next, we have a nested *if, else, then*. It first determines if there's any information already stored in web storage (someone has already click at least once) and then increments that number by 1 each time.

The else is for the circumstance where the click is actually the very first click, so make that number equal to 1.

```
{
if(typeof(Storage)!=="undefined")
  {
  if (localStorage.clickcount)
    {
    localStorage.clickcount=Number(localStorage.clickcount)+1;
    }
  else
    {
    localStorage.clickcount=1;
    }
```

We then fetch the HTML element by ID and add the number of clicks plus some text. We also handle the circumstance where someone's browser doesn't support web storage at all and give them a little error message: Your browser does not support web storage. We finally end the script.

```
  document.getElementById("result").innerHTML= + localStorage.clickcount + " people have
liked this page.";
  }
else
  {
  document.getElementById("result").innerHTML="Your browser does not support web
storage.";
  }
}
</script>
```

Next, we have the HTML code (the button) that calls this JavaScript. The button has an onclick event that correlates to the name of the function: clickCounter().

```
<body>
  <p><button onclick="clickCounter()" type="button">Like!</button></p>
```

After that, we have a place where the results will be displayed once the button is clicked. Note that the ID in the <div> element correlates to the ID in the script above.

```
      <div id="result"></div>
    </body>
```

Copy and paste the entire code above and put it into a new page. This API can run locally, so you don't need to upload the page to the internet unless you want to.
The point that you should take away here is that the number of times someone has clicked the button is stored on the web, not in a cookie or any other location. Although we've demonstrated Local Storage, there is also a different kind called Session Storage, which only lasts until that web browser session is running. When someone closes their browser, it resets.

## Drag and drop

You can let readers drag and drop objects on your web page from one spot to another. Maybe they are taking a test or quiz (match objects to definitions, for example) or maybe it's a new way of dropping products into an online shopping cart. The one we'll build in this tutorial is just for fun though.

You need code that sets three things:

- **What you can drag:** the image has draggable="true" on the event ondragstart and the script includes function drag(ev).
- **Where you can drop the object:** event.preventDefault() on the event ondragover.
- **What happens when you drop the object:** function drop(ev) on the event ondrop.

Try it yourself by setting draggable="true" on any element in your page (like an image or paragraph) and then try dragging it around. Just setting draggable lets you drag, but you can't drop it anywhere.

Here's the full code for it:

### Drag and drop

```
    <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>
      <img id="img1" src="puzzle1.png" ondrop="drop(event)"
      ondragover="allowDrop(event)"></div>

      <img id="img2" src="puzzle2.png" draggable="true"
      ondragstart="drag(event)">

    </body>
```

Try the full code by copying/pasting and then changing the src attribute value to point to a graphic you have.

In the first part of the script, we have the code that allows us to drop the object. We have to override the default behaviour, which is to NOT let objects be dropped anywhere.

## Allowing an element to be dropped

```
<head>
  <script>
    function allowDrop(ev)
      {
      ev.preventDefault();
      }
```

Next, we have the drag portion of the script which sets the type of information that is being dragged.

## Drag

```
 function drag(ev)
{
ev.dataTransfer.setData("Text",ev.target.id);
}
```

Next, we have the drop portion of the code, which sets the place where you can drop the object.

## Drop

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

```
            var data=ev.dataTransfer.getData("Text");
            ev.target.appendChild(document.getElementById(data));
            }
        </script>
```

Next, we have our HTML code that specifies which object is draggable and invokes the JavaScript using the ondrop and ondragover events on the place to drop it and the ondragstart event on the object to drag.

```
        <body>

          <img id="img1" src="puzzle1.png" ondrop="drop(event)"
          ondragover="allowDrop(event)"></div>

          <img id="img2" src="puzzle2.png" draggable="true"
          ondragstart="drag(event)">

        </body>
```

Try it out with your own graphic!

You are no longer new at HTML5. You have used advanced code to add some pretty awesome features to your web pages. You are now ready to build some pretty great pages and to learn more about CSS and Javascript.

## Final Tips:

Congratulations from VEIS, you have now reached the final lesson.
You have learned a lot and you are now capable of making your own websites! However, what you have learned are the basics and there is still a lot more to be mastered. But you now have a good foundation from which to build on.

### Tip 1

Remember that HTML5 is still fairly new and that some elements and attributes are only partially supported by browsers. Make sure you check that what you're building is mostly supported on the major browsers.

### Tip 2

Your website should still be all about content. HTML is your tool for sharing and presenting information on the Internet, so make sure there is information to present. Pretty pages may look nice but most people use the Internet to find information. There is no substitute for good content.

### Tip 3

Avoid overloading your pages with heavy images and other fancy stuff you have found on the Internet. It slows down the loading of the page and could be confusing for visitors.

Tip 4
Use the right element for the right content. Those semantic elements (<header>, <footer>, <article>, etc.) aren't merely for show. Use them appropriately and minimize your use of the not-so-semantic <div> element. If you start abusing the tags.

Tip 5
Remember to add your website to search engines/directories so people can find and enjoy it. On the front page of all search engines, you will find a link to add new pages. The most important is Google, but there are also others like Bing, Yahoo, Duck Duck Go, Webopedia, Dogpile, and Ask (Ask Jeeves).
Tip 6
So far, you have learned to use Notepad, which is a simple and very easy to use editor, you may use a more advanced editor which gives a better overview and more possibilities.
Uploading Pages
Until now, only you have had the pleasure of viewing your pages. Now it is time for the rest of the world to see your creations.

## Get your presence on net

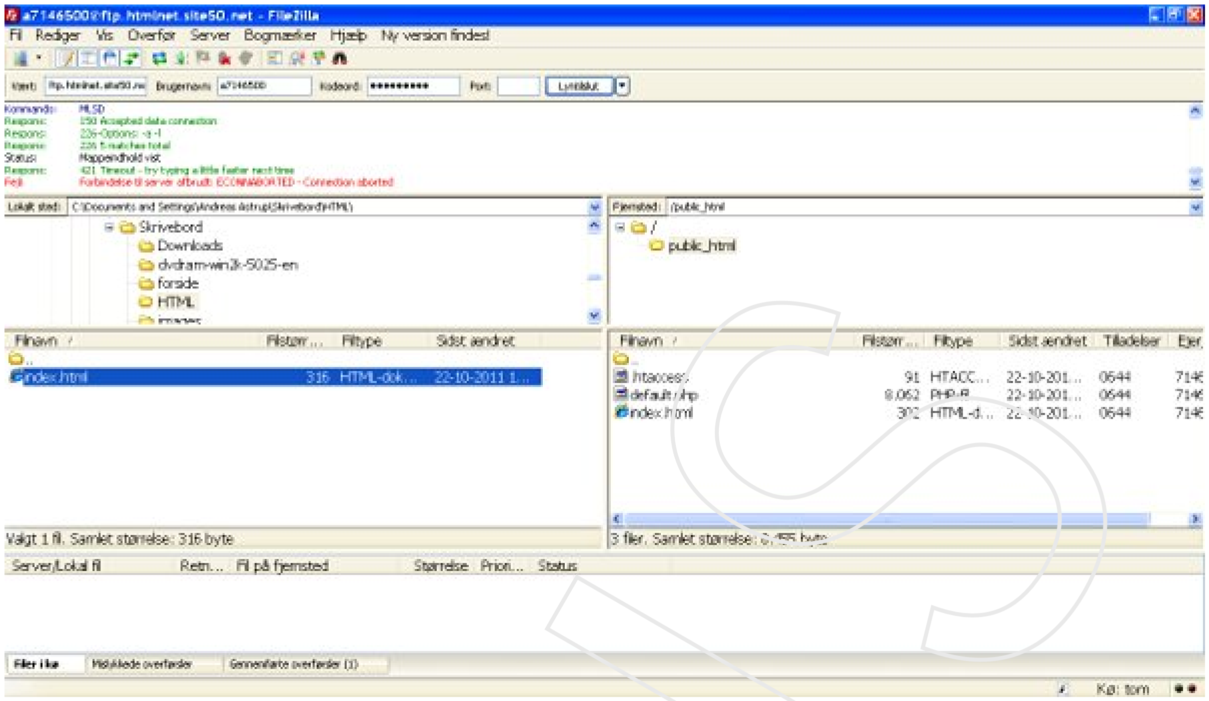
**To get your website on the internet, you just need some server space and a free FTP program**.
If you have Internet access, you might already have some free server space for your website. Your server space will then probably be called something like http://home.provider.com/~usernumber. But you might need to activate it first. Read more about this in the information from your Internet provider or on their support pages.
Another option is to get some free server space on the Internet. In the same way that you set up an e-mail account (like Gmail), **you can register for free server space on the Internet**. Several companies offer such a service — including 000webhost.com . It only takes a couple of minutes to register.
To have access to the server, you need to know the "Host Name" (for example, ftp.veinstitution.com) and have your username and password ready.

## Needs

To access the server and upload your pages, you also need an FTP program. FTP is short for File Transfer Protocol. A FTP program is used to connect two computers over the Internet so that you can transfer files from your computer to another computer (the server). You might not have such a program yet, but fortunately, this can be downloaded for free.
**There are many different FTP programs. One of the better is FileZilla, which is entirely free**. So now you can download FileZilla at filezilla.sourceforge.net (pick the Server option).

## Upload the pages

Described below is how you upload your pages to a free account at 000webhost.com with FileZilla. But the procedure is, more or less, the same for all providers and FTP programs.
Open the FTP program while connected to the Internet. Insert "Host Name" ("ftp.htmlnet.site50.net" under "Address"), username (under "User") and password (under "Password") and click "Connect". You should now have access to the server. In one side of the program you can see the contents of your computer ("Local Site"), and in the other side, you can see the content of the server ("Remote Site"):



```
Kommando:   MLSD
Respons:    150 Accepted data connection
Respons:    226-Options: -a -l
Respons:    226 5 matches total
Status:     Mappeindhold vist
Respons:    421 Timeout - try typing a little faster next time
Fejl:       Forbindelse til server afbrudt: ECONNABORTED - Connection aborted
```

| Filnavn | Filstørr... | Filtype | Sidst ændret |
|---|---|---|---|
| .. | | | |
| index.html | 316 | HTML-dok... | 22-10-2011 1... |

| Filnavn | Filstørr... | Filtype | Sidst ændret | Tilladelser | Ejer |
|---|---|---|---|---|---|
| .. | | | | | |
| .htaccess | 91 | HTACC... | 22-10-201... | 0644 | 714 |
| default.php | 8.062 | PHP-fil | 22-10-201... | 0644 | 714 |
| index.html | 302 | HTML-d... | 22-10-201... | 0644 | 714 |

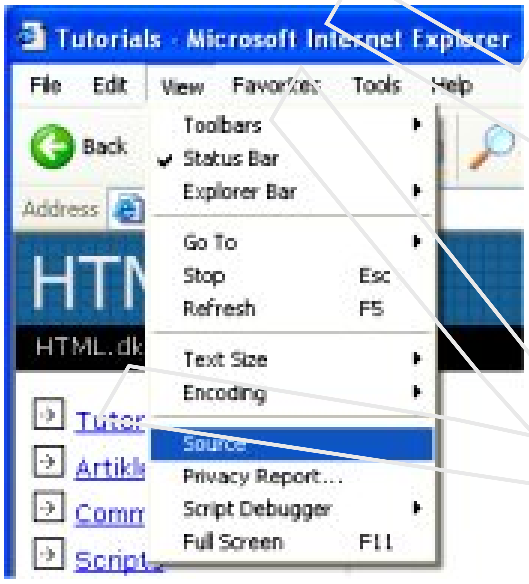
Valgt 1 fil. Samlet størrelse: 316 byte

a7146500@ftp.htmlnet.site50.net - FileZilla
Fil Rediger Vis Overfør Server Bogmærker Hjælp Ny version findes!
Lokalt sted: C:\Documents and Settings\Andreas Astrup\Skrivebord\HTML\
Skrivebord
Downloads
dvdram-win2k-5025-en
forside
HTML
images
Fjernsted: /public_html
/
public_html
Server/Lokal fil
Retn...
Fil på fjernsted
Størrelse
Priori...
Status
Filer i kø
Mislykkede overførsler
Gennemførte overførsler (1)
Kø: tom

Find your HTML documents and images on your computer (on the "Local site") and transfer them to the server (the "Remote site") by double clicking on them. Now the whole world can see them!

How do I learn more?

First of all, it is important that you continue to work and experiment with the things you have learned in this tutorial. Study other people's websites and if you find something you like see how it was made with "View Source" (Click "View" in the menu in your browser and choose "Source").



Search the Internet for examples and articles on HTML. There are lots of websites with great contents on HTML.

Last, but not least, you should - whenever you feel ready - **continue learning CSS** .

# Glossary

## Keyboard Shortcuts For Windows and Mac

Keyboard shortcuts are often used in modern operating systems and computer software programs. Using keyboard shortcuts could save you a lot of time.

### Basic Shortcuts

| Description | Windows | Mac OS |
|---|---|---|
| Edit menu | Alt + E | Ctrl + F2 + F |
| File menu | Alt + F | Ctrl + F2 + E |
| View menu | Alt + V | Ctrl + F2 + V |
| Select all text | Ctrl + A | Cmd + A |
| Copy text | Ctrl + C | Cmd + C |
| Find text | Ctrl + F | Cmd + F |
| Find and replace text | Ctrl + H | Cmd + F |
| New Document | Ctrl + N | Cmd + N |
| Open a file | Ctrl + O | Cmd + O |
| Print options | Ctrl + P | Cmd + P |
| Save file | Ctrl + S | Cmd + S |
| Paste text | Ctrl + V | Cmd + V |
| Cut text | Ctrl + X | Cmd + X |
| Redo text | Ctrl + Y | Shift + Cmd + Z |
| Undo text | Ctrl + Z | Cmd + Z |

### Text Editing

| Description | Windows | Mac OS |
|---|---|---|
| **Cursor Movement** | | |
| Go to the right or to the beginning of next line break | Right Arrow | Right Arrow |
| Go to the left or to the end of previous line break | Left Arrow | Left Arrow |
| Go up one row | Up Arrow | Up Arrow |
| Go down one row | Down Arrow | Down Arrow |
| Go to the beginning of the current line | Home | Cmd + Left Arrow |
| Go to the end of the current line | End | Cmd + Right Arrow |
| Go to the beginning of the document | Ctrl + Home | Cmd + Up Arrow |
| Go to the end of the document | Ctrl + End | Cmd + Down Arrow |
| Move up one frame | Page Up | Fn + Up Arrow |
| Move down one frame | Page Down | Fn + Down Arrow |
| Go to beginning of previous word | Ctrl + Left Arrow | Option + Left Arrow |
| Go to beginning of next word | Ctrl + Right Arrow | Option + Right Arrow |
| Go to beginning of line break | Ctrl + Up Arrow | Cmd + Left Arrow |

| | | |
|---|---|---|
| Go to end of line break | Ctrl + Down Arrow | Cmd + Right Arrow |
| **Text Selection** | | |
| Select characters to the left | Shift + Left Arrow | Shift + Left Arrow |
| Select characters to the right | Shift + Right Arrow | Shift + Right Arrow |
| Select lines upwards | Shift + Up Arrow | Shift + Up Arrow |
| Select lines downwards | Shift + Down Arrow | Shift + Down Arrow |
| Select words to the left | Shift + Ctrl + Left | Shift + Opt + Left |
| Select words to the right | Shift + Ctrl + Right | Shift + Opt + Right |
| Select paragraphs to the left | Shift + Ctrl + Up | Shift + Opt + Up |
| Select paragraphs to the right | Shift + Ctrl + Down | Shift + Opt + Down |
| Select text between the cursor and the beginning of the current line | Shift + Home | Cmd + Shift + Left Arrow |
| Select text between the cursor and the end of the current line | Shift + End | Cmd + Shift + Right Arrow |
| Select text between the cursor and the beginning of the document | Shift + Ctrl + Home | Cmd + Shift + Up Arrow or Cmd + Shift + Fn + Left Arrow |
| Select text between the cursor and the end of the document | Shift + Ctrl + End | Cmd + Shift + Down Arrow or Cmd + Shift + Fn + Right Arrow |
| Select one frame at a time of text above the cursor | Shift + Page Up | Shift + Fn + Up Arrow |
| Select one frame at a time of text below the cursor | Shift + Page Down | Shift + Fn + Down Arrow |
| Select all text | Ctrl + A | Cmd + A |
| Find text | Ctrl + F | Cmd + F |
| **Text Formatting** | | |
| Make selected text bold | Ctrl + B | Cmd + B |
| Make selected text italic | Ctrl + I | Cmd + I |
| Underline selected text | Ctrl + U | Cmd + U |
| Make selected text superscript | Ctrl + Shift + = | Cmd + Shift + = |
| Make selected text subscript | Ctrl + = | Cmd + = |
| **Text Editing** | | |
| Delete characters to the left | Backspace | Backspace |
| Delete characters to the right | Delete | Fn + Backspace |
| Delete words to the right | Ctrl + Del | Cmd + Backspace |
| Delete words to the left | Ctrl + Backspace | Cmd + Fn + Backspace |

| | | |
|---|---|---|
| Indent | Tab | Tab |
| Outdent | Shift + Tab | Shift + Tab |
| Copy text | Ctrl + C | Cmd + C |
| Find and replace text | Ctrl + H | Cmd + F |
| Paste text | Ctrl + V | Cmd + V |
| Cut text | Ctrl + X | Cmd + X |
| Redo text | Ctrl + Y | Shift + Cmd + Z |
| Undo text | Ctrl + Z | Cmd + Z |

**Web Browsers**

| **Description** | **Windows** | **Mac OS** |
|---|---|---|
| **Navigation** | | |
| Scroll down a frame | Space or Page Down | Space or Fn + Down Arrow |
| Scroll up a frame | Shift + Space or Page Up | Shift + Space or Fn + Up Arrow |
| Go to bottom of the page | End | Cmd + Down Arrow |
| Go to top of the page | Home | Cmd + Up Arrow |
| Go back | Alt + Left Arrow or Backspace | Cmd + Left Arrow |
| Go forward | Alt + Right Arrow or Shift + Backspace | Cmd + Right Arrow |
| Refresh a webpage | F5 | Cmd + R |
| Refresh a webpage (no cache) | Ctrl + F5 | Cmd + Shift + R |
| Stop | Esc | Esc |
| Toggle full-screen | F11 | Cmd + Shift + F |
| Zoom in | Ctrl + + | Cmd + + |
| Zoom out | Ctrl + - | Cmd + - |
| Zoom 100% (default) | Ctrl + 0 | Cmd + 0 |
| Open homepage | Alt + Home | Option + Home |
| Find text | Ctrl + F | Cmd + F |
| **Tab / Window Management** | | |
| Open a new tab | Ctrl + T | Cmd + T |
| Close current tab | Ctrl + W | Cmd + W |
| Close all tabs | Ctrl + Shift + W | Cmd + Q |
| Close all tabs except the current tab | Ctrl + Alt + F4 | Cmd + Opt + W |
| Go to next tab | Ctrl + Tab | Control + Tab |
| Go to previous tab | Ctrl + Shift + Tab | Shift + Control + Tab |
| Go to a specific tab number | Ctrl + 1-8 | Cmd + 1-8 |

| | | |
|---|---|---|
| Go to the last tab | Ctrl + 9 | Cmd + 9 |
| Reopen the last closed tab | Ctrl + Shift + T | Cmd + Shift + T |
| Open a new window | Ctrl + N | Cmd + N |
| Close current window | Alt + F4 | Cmd + W |
| Go to next window | Alt + Tab | Cmd + Tab |
| Go to previous window | Alt + Shift + Tab | Cmd + Shift + Tab |
| Reopen the last closed window | Ctrl + Shift + N | |
| Open links in a new tab in the background | Ctrl + Click | Cmd + Click |
| Open links in a new tab in the foreground | Ctrl + Shift + Click | Cmd + Shift + Click |
| Print current webpage | Ctrl + P | Cmd + P |
| Save current webpage | Ctrl + S | Cmd + S |
| **Address Bar** | | |
| Cycle between toolbar, search bar, and page elements | Tab | Tab |
| Go to browser's address bar | Ctrl + L or Alt + D | Cmd + L |
| Focus and select the browser's search bar | Ctrl + E | Cmd + E / Cmd + K |
| Open the address bar location in a new tab | Alt + Enter | Opt + Enter |
| Display a list of previously typed addresses | F4 | |
| Add "www." to the beginning and ".com" to the end of the text typed in the address bar | Ctrl + Enter | Cmd + Enter or Control + Enter |
| **Bookmarks** | | |
| Open the bookmarks menu | Ctrl + B | Cmd + B |
| Add bookmark for current page | Ctrl + D | Cmd + Opt + B or Cmd + Shift + B |
| Open browsing history | Ctrl + H | Cmd + Shift + H or Cmd + Y |
| Open download history | Ctrl + J | Cmd + J or Cmd + Shift + J |

**Screenshots**

| **Description** | **Windows** | **Mac OS** |
|---|---|---|
| Save screenshot of the whole screen as file | | Cmd + Shift + 3 |
| Copy screenshot of the whole screen to the clipboard | PrtScr (Print Screen) or Ctrl + PrtScr | Cmd + Ctrl + Shift + 3 |
| Save screenshot of window as file | | Cmd + Shift + 4, then Space |
| Copy screenshot of window to the clipboard | Alt + PrtScr | Cmd + Ctrl + Shift + 4, then Space |
| Copy screenshot of wanted area to the clipboard | | Cmd + Ctrl + Shift + 4 |
| Save screenshot of wanted area as file | | Cmd + Shift + 4 |

**Note:** Due to different keyboard setups, some shortcuts may not be compatible for all users.

## Color Examples

| Color | Color HEX | Color RGB |
|---|---|---|
| ⬛ | #000000 | rgb(0,0,0) |
| 🟥 | #FF0000 | rgb(255,0,0) |
| 🟩 | #00FF00 | rgb(0,255,0) |
| 🟦 | #0000FF | rgb(0,0,255) |
| 🟨 | #FFFF00 | rgb(255,255,0) |
| 🟦 | #00FFFF | rgb(0,255,255) |
| 🟪 | #FF00FF | rgb(255,0,255) |
| ▨ | #C0C0C0 | rgb(192,192,192) |
| ⬜ | #FFFFFF | rgb(255,255,255) |

## 16 Million Different Colors

The combination of Red, Green and Blue values from 0 to 255 gives a total of more than 16 million different colors to play with (256 x 256 x 256).

## Web Safe Colors:

Some years ago, when computers supported max 256 different colors, a list of 216 "Web Safe Colors" was suggested as a Web standard, reserving 40 fixed system colors. The 216 cross-browser color palette was created to ensure that all computers would display the colors correctly when running a 256 color palette:

| | | | | | |
|---|---|---|---|---|---|
| 000000 | 000033 | 000066 | 000099 | 0000CC | 0000FF |
| 003300 | 003333 | 003366 | 003399 | 0033CC | 0033FF |
| 006600 | 006633 | 006666 | 006699 | 0066CC | 0066FF |
| 009900 | 009933 | 009966 | 009999 | 0099CC | 0099FF |
| 00CC00 | 00CC33 | 00CC66 | 00CC99 | 00CCCC | 00CCFF |
| 00FF00 | 00FF33 | 00FF66 | 00FF99 | 00FFCC | 00FFFF |
| 330000 | 330033 | 330066 | 330099 | 3300CC | 3300FF |
| 333300 | 333333 | 333366 | 333399 | 3333CC | 3333FF |
| 336600 | 336633 | 336666 | 336699 | 3366CC | 3366FF |
| 339900 | 339933 | 339966 | 339999 | 3399CC | 3399FF |
| 33CC00 | 33CC33 | 33CC66 | 33CC99 | 33CCCC | 33CCFF |
| 33FF00 | 33FF33 | 33FF66 | 33FF99 | 33FFCC | 33FFFF |

| | | | | | |
|---|---|---|---|---|---|
| 660000 | 660033 | 660066 | 660099 | 6600CC | 6600FF |
| 663300 | 663333 | 663366 | 663399 | 6633CC | 6633FF |
| 666600 | 666633 | 666666 | 666699 | 6666CC | 6666FF |
| 669900 | 669933 | 669966 | 669999 | 6699CC | 6699FF |
| 66CC00 | 66CC33 | 66CC66 | 66CC99 | 66CCCC | 66CCFF |
| 66FF00 | 66FF33 | 66FF66 | 66FF99 | 66FFCC | 66FFFF |
| 990000 | 990033 | 990066 | 990099 | 9900CC | 9900FF |
| 993300 | 993333 | 993366 | 993399 | 9933CC | 9933FF |
| 996600 | 996633 | 996666 | 996699 | 9966CC | 9966FF |
| 999900 | 999933 | 999966 | 999999 | 9999CC | 9999FF |
| 99CC00 | 99CC33 | 99CC66 | 99CC99 | 99CCCC | 99CCFF |
| 99FF00 | 99FF33 | 99FF66 | 99FF99 | 99FFCC | 99FFFF |
| CC0000 | CC0033 | CC0066 | CC0099 | CC00CC | CC00FF |
| CC3300 | CC3333 | CC3366 | CC3399 | CC33CC | CC33FF |
| CC6600 | CC6633 | CC6666 | CC6699 | CC66CC | CC66FF |
| CC9900 | CC9933 | CC9966 | CC9999 | CC99CC | CC99FF |
| CCCC00 | CCCC33 | CCCC66 | CCCC99 | CCCCCC | CCCCFF |
| CCFF00 | CCFF33 | CCFF66 | CCFF99 | CCFFCC | CCFFFF |
| FF0000 | FF0033 | FF0066 | FF0099 | FF00CC | FF00FF |
| FF3300 | FF3333 | FF3366 | FF3399 | FF33CC | FF33FF |
| FF6600 | FF6633 | FF6666 | FF6699 | FF66CC | FF66FF |
| FF9900 | FF9933 | FF9966 | FF9999 | FF99CC | FF99FF |
| FFCC00 | FFCC33 | FFCC66 | FFCC99 | FFCCCC | FFCCFF |
| FFFF00 | FFFF33 | FFFF66 | FFFF99 | FFFFCC | FFFFFF |

## Color Names Supported by All Browsers

140 color names are defined in the HTML and CSS color specification (17 standard colors plus 123 more). The table below lists them all, along with their hexadecimal values.

## Sorted by Color Name

Colors sorted by HEX values

| **Color Name** | **HEX** | **Color** |
|---|---|---|
| Alice Blue | #F0F8FF | ⬜ |
| Antique White | #FAEBD7 | ⬜ |
| Aqua | #00FFFF | 🟦 |

| | | |
|---|---|---|
| Aquamarine | #7FFFD4 | 🟩 |
| Azure | #F0FFFF | ⬜ |
| Beige | #F5F5DC | ⬜ |
| Bisque | #FFE4C4 | ⬜ |
| Black | #000000 | ⬛ |
| Blanched Almond | #FFEBCD | ⬜ |
| Blue | #0000FF | 🟦 |
| BlueViolet | #8A2BE2 | 🟪 |
| Brown | #A52A2A | 🟫 |
| BurlyWood | #DEB887 | 🟫 |
| CadetBlue | #5F9EA0 | 🟦 |
| Chartreuse | #7FFF00 | 🟩 |
| Chocolate | #D2691E | 🟧 |
| Coral | #FF7F50 | 🟧 |
| CornflowerBlue | #6495ED | 🟦 |
| Cornsilk | #FFF8DC | ⬜ |
| Crimson | #DC143C | 🟥 |
| Cyan | #00FFFF | 🟦 |
| DarkBlue | #00008B | 🟦 |
| DarkCyan | #008B8B | 🟦 |
| DarkGoldenRod | #B8860B | 🟨 |
| DarkGray | #A9A9A9 | ⬜ |
| DarkGreen | #006400 | 🟩 |
| DarkKhaki | #BDB76B | 🟨 |
| DarkMagenta | #8B008B | 🟪 |
| DarkOliveGreen | #556B2F | 🟩 |
| DarkOrange | #FF8C00 | 🟧 |
| DarkOrchid | #9932CC | 🟪 |
| DarkRed | #8B0000 | 🟥 |
| DarkSalmon | #E9967A | 🟧 |
| DarkSeaGreen | #8FBC8F | 🟩 |
| DarkSlateBlue | #483D8B | 🟦 |
| DarkSlateGray | #2F4F4F | ⬛ |
| DarkTurquoise | #00CED1 | 🟦 |
| DarkViolet | #9400D3 | 🟪 |
| DeepPink | #FF1493 | 🟥 |
| DeepSkyBlue | #00BFFF | 🟦 |
| DimGray | #696969 | ⬛ |
| DodgerBlue | #1E90FF | 🟦 |

| | | |
|---|---|---|
| FireBrick | #B22222 | 🟥 |
| FloralWhite | #FFFAF0 | ⬜ |
| ForestGreen | #228B22 | 🟩 |
| Fuchsia | #FF00FF | 🟪 |
| Gainsboro | #DCDCDC | ⬜ |
| GhostWhite | #F8F8FF | ⬜ |
| Gold | #FFD700 | 🟨 |
| GoldenRod | #DAA520 | 🟨 |
| Gray | #808080 | ⬛ |
| Green | #008000 | 🟩 |
| GreenYellow | #ADFF2F | 🟩 |
| HoneyDew | #F0FFF0 | ⬜ |
| HotPink | #FF69B4 | 🟪 |
| IndianRed | #CD5C5C | 🟥 |
| Indigo | #4B0082 | 🟪 |
| Ivory | #FFFFF0 | ⬜ |
| Khaki | #F0E68C | 🟨 |
| Lavender | #E6E6FA | ⬜ |
| LavenderBlush | #FFF0F5 | ⬜ |
| LawnGreen | #7CFC00 | 🟩 |
| LemonChiffon | #FFFACD | 🟨 |
| LightBlue | #ADD8E6 | 🟦 |
| LightCoral | #F08080 | 🟥 |
| LightCyan | #E0FFFF | 🟦 |
| LightGoldenRodYellow | #FAFAD2 | 🟨 |
| LightGray | #D3D3D3 | ⬜ |
| LightGreen | #90EE90 | 🟩 |
| LightPink | #FFB6C1 | 🟥 |
| LightSalmon | #FFA07A | 🟧 |
| LightSeaGreen | #20B2AA | 🟩 |
| LightSkyBlue | #87CEFA | 🟦 |
| LightSlateGray | #778899 | ⬛ |
| LightSteelBlue | #B0C4DE | 🟦 |
| LightYellow | #FFFFE0 | 🟨 |
| Lime | #00FF00 | 🟩 |
| LimeGreen | #32CD32 | 🟩 |
| Linen | #FAF0E6 | ⬜ |
| Magenta | #FF00FF | 🟪 |
| Maroon | #800000 | 🟫 |

| | | |
|---|---|---|
| MediumAquaMarine | #66CDAA | 🟩 |
| MediumBlue | #0000CD | 🟦 |
| MediumOrchid | #BA55D3 | 🟪 |
| MediumPurple | #9370DB | 🟪 |
| MediumSeaGreen | #3CB371 | 🟩 |
| MediumSlateBlue | #7B68EE | 🟦 |
| MediumSpringGreen | #00FA9A | 🟩 |
| MediumTurquoise | #48D1CC | 🟦 |
| MediumVioletRed | #C71585 | 🟥 |
| MidnightBlue | #191970 | 🟦 |
| MintCream | #F5FFFA | ⬜ |
| MistyRose | #FFE4E1 | ⬜ |
| Moccasin | #FFE4B5 | 🟨 |
| NavajoWhite | #FFDEAD | 🟨 |
| Navy | #000080 | 🟦 |
| OldLace | #FDF5E6 | ⬜ |
| Olive | #808000 | 🟩 |
| OliveDrab | #6B8E23 | 🟩 |
| Orange | #FFA500 | 🟧 |
| OrangeRed | #FF4500 | 🟧 |
| Orchid | #DA70D6 | 🟪 |
| PaleGoldenRod | #EEE8AA | 🟨 |
| PaleGreen | #98FB98 | 🟩 |
| PaleTurquoise | #AFEEEE | 🟦 |
| PaleVioletRed | #DB7093 | 🟥 |
| PapayaWhip | #FFEFD5 | ⬜ |
| PeachPuff | #FFDAB9 | 🟧 |
| Peru | #CD853F | 🟫 |
| Pink | #FFC0CB | 🟥 |
| Plum | #DDA0DD | 🟪 |
| PowderBlue | #B0E0E6 | 🟦 |
| Purple | #800080 | 🟪 |
| Red | #FF0000 | 🟥 |
| RosyBrown | #BC8F8F | 🟫 |
| RoyalBlue | #4169E1 | 🟦 |
| SaddleBrown | #8B4513 | 🟫 |
| Salmon | #FA8072 | 🟥 |
| SandyBrown | #F4A460 | 🟧 |
| SeaGreen | #2E8B57 | 🟩 |

| | | |
|---|---|---|
| SeaShell | #FFF5EE | ⬜ |
| Sienna | #A0522D | 🟫 |
| Silver | #C0C0C0 | ⬜ |
| SkyBlue | #87CEEB | 🟦 |
| SlateBlue | #6A5ACD | 🟪 |
| SlateGray | #708090 | ⬛ |
| Snow | #FFFAFA | ⬜ |
| SpringGreen | #00FF7F | 🟩 |
| SteelBlue | #4682B4 | 🟦 |
| Tan | #D2B48C | 🟫 |
| Teal | #008080 | 🟦 |
| Thistle | #D8BFD8 | 🟪 |
| Tomato | #FF6347 | 🟥 |
| Turquoise | #40E0D0 | 🟦 |
| Violet | #EE82EE | 🟪 |
| Wheat | #F5DEB3 | 🟨 |
| White | #FFFFFF | ⬜ |
| WhiteSmoke | #F5F5F5 | ⬜ |
| Yellow | #FFFF00 | 🟨 |
| YellowGreen | #9ACD32 | 🟩 |