

The **World Wide Web Consortium (W3C)** is an international community where Member organizations, a full-time staff, and the public work together to develop Web standards.

Design Principles

The following design principles guide W3C's work.

Web for All

The social value of the Web is that it enables human communication, commerce, and opportunities to share knowledge. One of W3C's primary goals is to make these benefits available to all people, whatever their hardware, software, network infrastructure, native language, culture, geographical location, or physical or mental ability.

Web on Everything

The number of different kinds of devices that can access the Web has grown immensely. Mobile phones, smart phones, personal digital assistants, interactive television systems, voice response systems, kiosks and even certain domestic appliances can all access the Web.

Vision

W3C's vision for the Web involves participation, sharing knowledge, and thereby building trust on a global scale.

Web for Rich Interaction

The Web was invented as a communications tool intended to allow anyone, anywhere to share information. For many years, the Web was a "read-only" tool for many. Blogs and wikis brought more authors to the Web, and social networking emerged from the flourishing market for content and personalized Web experiences. W3C standards have supported this evolution thanks to strong architecture and design principles.

Web of Data and Services

Some people view the Web as a giant repository of linked data while others as a giant set of services that exchange messages. The two views are complementary, and which to use often depends on the application.

Web of Trust

The Web has transformed the way we communicate with each other. In doing so, it has also modified the nature of our social relationships. People now "meet on the Web" and carry out commercial and personal relationships, in some cases without ever meeting in person. W3C recognizes that trust is a social phenomenon, but technology design can foster trust and confidence. As more activity moves on-line, it will become even more important to support complex interactions among parties around the globe.

What is HTML?

[HTML](#) is the language for ***describing the structure of Web pages***. HTML gives authors the means to:

- Publish online documents with headings, text, tables, lists, photos, etc.
- Retrieve online information via hypertext links, at the click of a button.
- Design forms for conducting transactions with remote services, for use in searching for information, making reservations, ordering products, etc.
- Include spread-sheets, video clips, sound clips, and other applications directly in their documents.

With HTML, authors describe the structure of pages using *markup*. The *elements* of the language label pieces of content such as "paragraph," "list," "table," and so on.

HTML Element Reference

HTML Tags Ordered Alphabetically

☺New in HTML5. ☹Not supported in HTML5.

| Tag | Category | Description |
|--------------|----------------------|--|
| <!--...--> | Basic | Defines a comment |
| <!DOCTYPE> | Basic | Defines the document type |
| <a> | Links | Defines a hyperlink |
| <abbr> | Formatting | Defines an abbreviation or an acronym |
| <acronym> | Formatting | Defines an acronym ☹ Use <abbr> instead. |
| <address> | Formatting | Defines contact information for the author/owner of a document |
| <applet> | Programming | Defines an embedded applet ☹Use <embed> or <object> instead. |
| <area> | Images | Defines an area inside an image-map |
| <article> | Styles and Semantics | ☺Defines an article |
| <aside> | Styles and Semantics | ☺Defines content aside from the page content |
| <audio> | Audio / Video | ☺Defines sound content |
| | Formatting | Defines bold text |
| <base> | Meta Info | Specifies the base URL/target for all relative URLs in a document |
| <basefont> | Meta Info | Specifies a default color, size, and font for all text in a document ☹Use CSS instead. |
| <bdi> | Formatting | ☺Isolates a part of text that might be formatted in a different direction from other text outside it |
| <bdo> | Formatting | Overrides the current text direction |
| <big> | Formatting | Defines big text ☹ Use CSS instead. |
| <blockquote> | Formatting | Defines a section that is quoted from another source |
| <body> | Basic | Defines the document's body |
| | Basic | Defines a single line break |
| <button> | Forms and Input | Defines a clickable button |
| <canvas> | Images | ☺Used to draw graphics, on the fly, via scripting (usually JavaScript) |
| <caption> | Tables | Defines a table caption |
| <center> | Formatting | Defines centered text ☹ Use CSS instead. |
| <cite> | Formatting | Defines the title of a work |
| <code> | Formatting | Defines a piece of computer code |
| <col> | Tables | Specifies column properties for each column within a <colgroup> element |
| <colgroup> | Tables | Specifies a group of one or more columns in a table for formatting |
| <datalist> | Forms and Input | ☺Specifies a list of pre-defined options for input controls |
| <dd> | Lists | Defines a description/value of a term in a description list |
| | Formatting | Defines text that has been deleted from a document |
| <details> | Styles and Semantics | ☺Defines additional details that the user can view or hide |
| <dfn> | Formatting | Represents the defining instance of a term |
| <dialog> | Styles and Semantics | ☺Defines a dialog box or window |
| <dir> | Lists | Defines a directory list ☹ Use instead. |
| <div> | Styles and Semantics | Defines a section in a document |
| <dl> | Lists | Defines a description list |
| <dt> | Lists | Defines a term/name in a description list |
| | Formatting | Defines emphasized text |
| <embed> | Programming | ☺Defines a container for an external (non-HTML) application |
| <fieldset> | Forms and Input | Groups related elements in a form |
| <figcaption> | Images | ☺Defines a caption for a <figure> element |
| <figure> | Images | ☺Specifies self-contained content |

| | | | |
|---------------------------------|----------------------|--|--------------------|
| | Formatting | Defines font, color, and size for text | ☹ Use CSS instead. |
| <footer> | Styles and Semantics | ☺Defines a footer for a document or section | |
| <form> | Forms and Input | Defines an HTML form for user input | |
| <frame> | Frames | Defines a window (a frame) in a frameset | ☹ |
| <frameset> | Frames | Defines a set of frames | ☹ |
| <h1> to <h6> | Basic | Defines HTML headings | |
| <head> | Basic, Meta Info | Defines information about the document | |
| <header> | Styles and Semantics | Defines a header for a document or section | |
| <hr> | Basic | Defines a thematic change in the content | |
| <html> | Basic | Defines the root of an HTML document | |
| <i> | Formatting | Defines a part of text in an alternate voice or mood | |
| <iframe> | Frames | Defines an inline frame | |
| | Images | Defines an image | |
| <input> | Forms and Input | Defines an input control | |
| <ins> | Formatting | Defines a text that has been inserted into a document | |
| <kbd> | Formatting | Defines keyboard input | |
| <label> | Forms and Input | Defines a label for an <input> element | |
| <legend> | Forms and Input | Defines a caption for a <fieldset> element | |
| | Lists | Defines a list item | |
| <link> | Links | Defines the relationship between a document and an external resource (most used to link to style sheets) | |
| <main> | Styles and Semantics | ☺Specifies the main content of a document | |
| <map> | Images | Defines a client-side image-map | |
| <mark> | Formatting | ☺Defines marked/highlighted text | |
| <menu> | Lists | Defines a list/menu of commands | |
| <menuitem> | Lists | ☺Defines a command/menu item that the user can invoke from a popup menu | |
| <meta> | Meta Info | Defines metadata about an HTML document | |
| <meter> | Formatting | ☺Defines a scalar measurement within a known range (a gauge) | |
| <nav> | Links | ☺Defines navigation links | |
| <noframes> | Frames | Defines an alternate content for users that do not support frames | ☹ |
| <noscript> | Programming | Defines an alternate content for users that do not support client-side scripts | |
| <object> | Programming | Defines an embedded object | |
| | Lists | Defines an ordered list | |
| <optgroup> | Forms and Input | Defines a group of related options in a drop-down list | |
| <option> | Forms and Input | Defines an option in a drop-down list | |
| <output> | Forms and Input | ☺Defines the result of a calculation | |
| <p> | Basic | Defines a paragraph | |
| <param> | Programming | Defines a parameter for an object | |
| <picture> | Images | ☺Defines a container for multiple image resources | |
| <pre> | Formatting | Defines preformatted text | |
| <progress> | Formatting | ☺Represents the progress of a task | |
| <q> | Formatting | Defines a short quotation | |
| <rp> | Formatting | ☺Defines what to show in browsers that do not support ruby annotations | |
| <rt> | Formatting | ☺Defines an explanation/pronunciation of characters (for East Asian typography) | |
| <ruby> | Formatting | ☺Defines a ruby annotation (for East Asian typography) | |
| <s> | Formatting | Defines text that is no longer correct | |
| <samp> | Formatting | Defines sample output from a computer program | |

| | | |
|------------|----------------------|--|
| <script> | Programming | Defines a client-side script |
| <section> | Styles and Semantics | ☺ Defines a section in a document |
| <select> | Forms and Input | Defines a drop-down list |
| <small> | Formatting | Defines smaller text |
| <source> | Audio / Video | ☺ Defines multiple media resources for media elements (<video> and <audio>) |
| | Styles and Semantics | Defines a section in a document |
| <strike> | Formatting | Defines strikethrough text ☹️Use or <s> instead. |
| | Formatting | Defines important text |
| <style> | Styles and Semantics | Defines style information for a document |
| <sub> | Formatting | Defines subscripted text |
| <summary> | Styles and Semantics | ☺ Defines a visible heading for a <details> element |
| <sup> | Formatting | Defines superscripted text |
| <table> | Tables | Defines a table |
| <tbody> | Tables | Groups the body content in a table |
| <td> | Tables | Defines a cell in a table |
| <textarea> | | Defines a multiline input control (text area) |
| <tfoot> | Tables | Groups the footer content in a table |
| <th> | Tables | Defines a header cell in a table |
| <thead> | Tables | Groups the header content in a table |
| <time> | Formatting | ☺ Defines a date/time |
| <title> | Basic | Defines a title for the document |
| <tr> | Tables | Defines a row in a table |
| <track> | Audio / Video | ☺ Defines text tracks for media elements (<video> and <audio>) |
| <tt> | Formatting | Defines teletype text ☹️Use CSS instead. |
| <u> | Formatting | Defines text that should be stylistically different from normal text |
| | Lists | Defines an unordered list |
| <var> | Formatting | Defines a variable |
| <video> | Audio / Video | ☺ Defines a video or movie |
| <wbr> | Formatting | ☺ Defines a possible line-break |

URL Encoding Reference

Your browser will encode input, according to the character-set used in your page.

The default character-set in HTML5 is UTF-8.



























| Character | UTF-8 | Character | UTF-8 | Character | UTF-8 | Character | UTF-8 |
|-----------|-------|-----------|-------|-----------|-----------|-----------|--------|
| space | %20 | H | %48 | p | %70 | ˜ | %CB%9C |
| ! | %21 | I | %49 | q | %71 | ™ | %E2%84 |
| " | %22 | J | %4A | r | %72 | Š | %C5%A1 |
| # | %23 | K | %4B | s | %73 | › | %E2%80 |
| \$ | %24 | L | %4C | t | %74 | œ | %C5%93 |
| % | %25 | M | %4D | u | %75 | | %9D |
| & | %26 | N | %4E | v | %76 | ž | %C5%BE |
| ' | %27 | O | %4F | w | %77 | Ÿ | %C5%B8 |
| (| %28 | P | %50 | x | %78 | | %C2%A0 |
|) | %29 | Q | %51 | y | %79 | ı | %C2%A1 |
| * | %2A | R | %52 | z | %7A | ¢ | %C2%A2 |
| + | %2B | S | %53 | { | %7B | £ | %C2%A3 |
| , | %2C | T | %54 | | %7C | ¼ | %C2%A4 |
| - | %2D | U | %55 | } | %7D | ¥ | %C2%A5 |
| . | %2E | V | %56 | ~ | %7E | ı | %C2%A6 |
| / | %2F | W | %57 | | %7F | § | %C2%A7 |
| 0 | %30 | X | %58 | ` | %E2%82%AC | ¨ | %C2%A8 |
| 1 | %31 | Y | %59 | | %81 | © | %C2%A9 |
| 2 | %32 | Z | %5A | , | %E2%80%9A | ª | %C2%AA |
| 3 | %33 | [| %5B | f | %C6%92 | « | %C2%AB |
| 4 | %34 | \ | %5C | „ | %E2%80%9E | ¬ | %C2%AC |
| 5 | %35 |] | %5D | ... | %E2%80%A6 | | %C2%AD |
| 6 | %36 | ^ | %5E | † | %E2%80%A0 | ® | %C2%AE |
| 7 | %37 | _ | %5F | ‡ | %E2%80%A1 | ˆ | %C2%AF |
| 8 | %38 | ` | %60 | ˆ | %CB%86 | ° | %C2%B0 |
| 9 | %39 | a | %61 | ‰ | %E2%80%B0 | ± | %C2%B1 |
| : | %3A | b | %62 | Š | %C5%A0 | ² | %C2%B2 |
| ; | %3B | c | %63 | ‹ | %E2%80%B9 | ³ | %C2%B3 |
| < | %3C | d | %64 | ƒ | %C5%92 | ˘ | %C2%B4 |
| = | %3D | e | %65 | | %C5%8D | μ | %C2%B5 |
| > | %3E | f | %66 | Ž | %C5%BD | ¶ | %C2%B6 |
| ? | %3F | g | %67 | | %8F | · | %C2%B7 |
| @ | %40 | h | %68 | | %C2%90 | ˙ | %C2%B8 |
| A | %41 | i | %69 | ’ | %E2%80%98 | ¹ | %C2%B9 |
| B | %42 | j | %6A | ’ | %E2%80%99 | º | %C2%BA |
| C | %43 | k | %6B | “ | %E2%80%9C | » | %C2%BB |
| D | %44 | l | %6C | ” | %E2%80%9D | ¼ | %C2%BC |
| E | %45 | m | %6D | • | %E2%80%A2 | ½ | %C2%BD |
| F | %46 | n | %6E | – | %E2%80%93 | ¾ | %C2%BE |
| G | %47 | o | %6F | — | %E2%80%94 | More... | |

















































The ASCII control characters **%00-%1F** were originally designed to control hardware devices. Control characters have nothing to do inside a URL.

[illegible]

HTML Color Names

Color Names Supported by All Browsers

| Color Name | HEX | Color | Color Name | HEX | Color |
|----------------|---------|---|----------------------|---------|--|
| AliceBlue | #F0F8FF |  | GhostWhite | #F8F8FF |  |
| AntiqueWhite | #FAEBD7 |  | Gold | #FFD700 |  |
| Aqua | #00FFFF |  | GoldenRod | #DAA520 |  |
| Aquamarine | #7FFFD4 |  | Gray | #808080 |  |
| Azure | #F0FFFF |  | Grey | #808080 |  |
| Beige | #F5F5DC |  | Green | #008000 |  |
| Bisque | #FFE4C4 |  | GreenYellow | #ADFF2F |  |
| Black | #000000 |  | HoneyDew | #F0FFF0 |  |
| BlanchedAlmond | #FFEBCD |  | HotPink | #FF69B4 |  |
| Blue | #0000FF |  | IndianRed | #CD5C5C |  |
| BlueViolet | #8A2BE2 |  | Indigo | #4B0082 |  |
| Brown | #A52A2A |  | Ivory | #FFFFFF |  |
| BurlyWood | #DEB887 |  | Khaki | #F0E68C |  |
| CadetBlue | #5F9EA0 |  | Lavender | #E6E6FA |  |
| Chartreuse | #7FFF00 |  | LavenderBlush | #FFF0F5 |  |
| Chocolate | #D2691E |  | LawnGreen | #7CFC00 |  |
| Coral | #FF7F50 |  | LemonChiffon | #FFFACD |  |
| CornflowerBlue | #6495ED |  | LightBlue | #ADD8E6 |  |
| Cornsilk | #FFF8DC |  | LightCoral | #F08080 |  |
| Crimson | #DC143C |  | LightCyan | #E0FFFF |  |
| Cyan | #00FFFF |  | LightGoldenRodYellow | #FAFAD2 |  |
| DarkBlue | #00008B |  | LightGray | #D3D3D3 |  |
| DarkCyan | #008B8B |  | LightGrey | #D3D3D3 |  |
| DarkGoldenRod | #B8860B |  | LightGreen | #90EE90 |  |
| DarkGray | #A9A9A9 |  | LightPink | #FFB6C1 |  |
| DarkGrey | #A9A9A9 |  | LightSalmon | #FFA07A |  |
| DarkGreen | #006400 |  | LightSeaGreen | #20B2AA |  |
| DarkKhaki | #BDB76B |  | LightSkyBlue | #87CEFA |  |
| DarkMagenta | #8B008B |  | LightSlateGray | #778899 |  |
| DarkOliveGreen | #556B2F |  | LightSlateGrey | #778899 |  |
| DarkOrange | #FF8C00 |  | LightSteelBlue | #B0C4DE |  |
| DarkOrchid | #9932CC |  | LightYellow | #FFFFE0 |  |
| DarkRed | #8B0000 |  | Lime | #00FF00 |  |
| DarkSalmon | #E9967A |  | LimeGreen | #32CD32 |  |
| DarkSeaGreen | #8FBC8F |  | Linen | #FAF0E6 |  |
| DarkSlateBlue | #483D8B |  | Magenta | #FF00FF |  |
| DarkSlateGray | #2F4F4F |  | Maroon | #800000 |  |
| DarkSlateGrey | #2F4F4F |  | MediumAquaMarine | #66CDAA |  |
| DarkTurquoise | #00CED1 |  | MediumBlue | #0000CD |  |
| DarkViolet | #9400D3 |  | MediumOrchid | #BA55D3 |  |
| DeepPink | #FF1493 |  | MediumPurple | #9370DB |  |
| DeepSkyBlue | #00BFFF |  | MediumSeaGreen | #3CB371 |  |
| DimGray | #696969 |  | MediumSlateBlue | #7B68EE |  |
| DimGrey | #696969 |  | MediumSpringGreen | #00FA9A |  |
| DodgerBlue | #1E90FF |  | MediumTurquoise | #48D1CC |  |
| FireBrick | #B22222 |  | MediumVioletRed | #C71585 |  |
| FloralWhite | #FFFAF0 |  | MidnightBlue | #191970 |  |
| ForestGreen | #228B22 |  | MintCream | #F5FFFA |  |
| Fuchsia | #FF00FF |  | MistyRose | #FFE4E1 |  |
| Gainsboro | #DCDCDC |  | Moccasin | #FFE4B5 |  |

| Color Name | HEX | Color |
|---------------|---------|---|
| 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 |  |
| RebeccaPurple | #663399 |  |
| 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 |  |
| SlateGrey | #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 |  |

HTML Introduction

What is HTML?

HTML is the standard markup language for creating Web pages.

- HTML stands for Hyper Text Markup Language
- HTML describes the structure of Web pages using markup
- HTML elements are the building blocks of HTML pages
- HTML elements are represented by tags
- HTML tags label pieces of content such as "heading", "paragraph", "table", and so on
- Browsers do not display the HTML tags, but use them to render the content of the page

A Simple HTML Document

Example [[HTML Example 1.html](#)]

```
<!DOCTYPE html>
<html>
<head>
<title>Computing 2017 Web Application</title>
</head>
<body>
<h1>Web Application </h1>
<p>Project 1</p>
</body>
</html>
```

Explained

- The <!DOCTYPE html> **declaration** defines this document to be HTML5
- The <html> element is the **root element** of an HTML page
- The <head> element contains **meta information** about the document
- The <title> element specifies a **title** for the document
- The <body> element contains the **visible page content**
- The <h1> element defines a large heading
- The <p> element defines a paragraph

HTML Tags

HTML tags are **element names** surrounded by **angle brackets**:

<tagname>content goes here...</tagname>

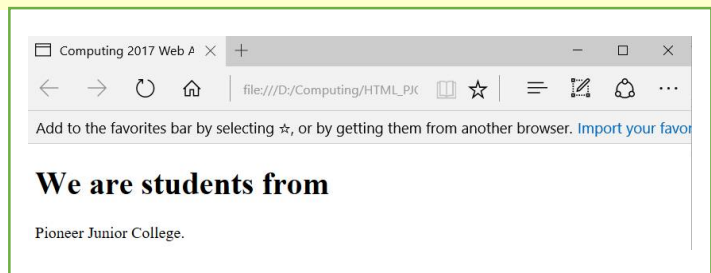
- HTML tags normally come **in pairs** like <p> and </p>
- The first tag in a pair is the **start tag**, the second tag is the **end tag**
- The end tag is written like the start tag, but with a **forward slash** inserted before the tag name

Tip: The start tag is also called the **opening tag**, and the end tag the **closing tag**.

Web Browsers

The purpose of a web browser (Chrome, IE, Firefox, Safari) is to read HTML documents and display them.

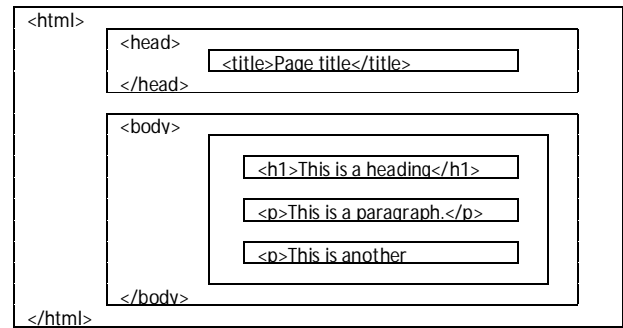
The browser does not display the HTML tags, but uses them to determine how to display the document:



HTML Page Structure

Below is a visualization of an HTML page structure:

Note: Only the content inside the `<body>` section (the white area above) is displayed in a browser.



The <!DOCTYPE> Declaration

The `<!DOCTYPE>` declaration represents the document type, and helps browsers to display web pages correctly. It must only appear once, at the top of the page (before any HTML tags).

The `<!DOCTYPE>` declaration is not case sensitive.

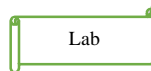
The `<!DOCTYPE>` declaration for HTML5 is:

`<!DOCTYPE html>`

HTML Versions

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

HTML Editors



| Version | Year |
|-----------|------|
| HTML | 1991 |
| HTML 4.01 | 1999 |
| XHTML | 2000 |
| HTML5 | 2014 |

HTML Documents

All HTML documents must start with a document type declaration: `<!DOCTYPE html>`.

The HTML document itself begins with `<html>` and ends with `</html>`.

The visible part of the HTML document is between `<body>` and `</body>`.

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

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

Example

`<h1>Mathematics</h1>`

`<h2>Physics</h2>`

`<h6>Computing</h6>`

HTML Paragraphs - HTML paragraphs are defined with the `<p>` tag:

Example

`<p>Feasibility paragraph.</p>`

`<p>Problem Definition paragraph.</p>`

HTML Links - HTML links are defined with the `<a>` tag:

Example

`<h4>Jurong Pioneer Junior College</h4>`

The link's destination is specified in the **href attribute**.

Attributes are used to provide additional information about HTML elements.

HTML Images

HTML images are defined with the `` tag.

The source file (`src`), alternative text (`alt`), width, and height are provided as attributes:

Example

``

HTML Elements

An **HTML** element usually consists of a **start** tag and **end** tag, with the content inserted in between:

`<tagname>Content goes here...</tagname>`

The HTML **element** is everything from the start tag to the end tag:

`<p> Computing</p>`

| Start tag | Element content | End tag |
|-------------------------|-----------------|--------------------------|
| <code><h1></code> | Science stream | <code></h1></code> |
| <code><p></code> | Computing. | <code></p></code> |
| <code> </code> | | |

HTML elements with no content are called **empty elements**. Empty elements do not have an end tag, such as the `
` element (which indicates a line break).

Nested HTML Elements

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

All HTML documents consist of nested HTML elements.

This example contains **four** HTML elements:

Example

```
<!DOCTYPE html>
<html>
<body>
<h1>Science Stream</h1>
<p>Computing.</p>
</body>
</html>
```

Example Explained [Self reading]

The `<html>` element defines the **whole document**.

It has a **start** tag `<html>` and an **end** tag `</html>`.

The element **content** is another HTML element (the `<body>` element).

The `<body>` element defines the **document body**.

It has a **start** tag `<body>` and an **end** tag `</body>`.

The element **content** is two other HTML elements (`<h1>` and `<p>`).

The `<h1>` element defines a **heading**.

It has a **start** tag `<h1>` and an **end** tag `</h1>`.

The element **content** is: Science Stream.

The `<p>` element defines a **paragraph**.

It has a **start** tag `<p>` and an **end** tag `</p>`.

The element **content** is: Computing.

Do Not Forget the End Tag

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

Example

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

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

Never rely on this. It might produce unexpected results and/or errors if you forget the end tag.

Empty HTML Elements

HTML elements with no content are called empty elements.

`
` is an empty element without a closing tag (the `
` tag defines a line break).

Empty elements can be "closed" in the opening tag like this: `
`.

Use Lowercase Tags

HTML tags are not case sensitive: `<P>` means the same as `<p>`.

W3C recommends lowercase in HTML, and **demands** lowercase for stricter document types like XHTML.

We always use lowercase tags.

HTML Attributes

Attributes provide additional information about HTML elements.

HTML Attributes

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

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

Example

```
<a href="https://jpjc.moe.edu.sg/"><h4>Jurong Pioneer Junior College</h4></a>
```

The src Attribute - HTML images are defined with the `` tag.

The filename of the image source is specified in the **src** attribute:

Example

```

```

The width and height Attributes - Images in HTML have a set of size attributes, which specifies the width and height of the image:

Example

```

```

The image size is specified in pixels: width="104" means 104 pixels wide.

The alt Attribute - The **alt** attribute specifies an alternative text to be used, when an image cannot be displayed.

The value of the attribute can be read by screen readers. This way, someone "listening" to the webpage, e.g. a blind person, can "hear" the element.

Example

```

```

The style Attribute - The style attribute is used to specify the styling of an element, like color, font, size etc.

Example

```
<p style="color:blue">System Analysis</p>
```

The lang Attribute - The language of the document can be declared in the `<html>` tag.

The language is declared with the **lang** attribute.

Declaring a language is important for accessibility applications (screen readers) and search engines:

```
<!DOCTYPE html>
```

```
<html lang="en-US">
```

```
<body>
```

```
...
```

```
</body>
```

```
</html>
```

The first two letters specify the language (en). If there is a dialect, use two more letters (US).

The title Attribute - Here, a **title** attribute is added to the **<p>** element. The value of the title attribute will be displayed as a tooltip when you mouse over the paragraph:

Example

```
<p title="9597">
```

Computing GCEA 2012-2019.

```
</p>
```

Use Lowercase Attributes

The title attribute can be written with uppercase or lowercase like **title** or **TITLE**.

W3C **recommends** lowercase in HTML, and **demand**s lowercase for stricter document types like XHTML.

Always use lowercase attribute names.

Quote Attribute Values

The **href** attribute, demonstrated above, can be written as:

Example

```
<a href="http://jpjc.moe.edu.sg"><h4>Jurong Pioneer Junior College</h4></a>
```

W3C **recommends** quotes in HTML, and **demand**s quotes for stricter document types like XHTML.

Sometimes it is **necessary** to use quotes. This example will not display the title attribute correctly, because it contains a space:

Example

```
<p title=Computer Science>
```

Using quotes are the most common. Omitting quotes can produce errors.

Always use quotes around attribute values.

Single or Double Quotes?

Double quotes around attribute values are the most common in HTML, but single quotes can also be used.

In some situations, when the attribute value itself contains double quotes, it is necessary to use single quotes:

```
<p title='I Love "Computing" Completely'>
```

Or vice versa:

```
<p title=""I Love 'Computing' completely "">
```

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>Physics</h1> <h5>Computing</h5>
```

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

Headings Are Important

Search engines use the headings to index the structure and content of your web pages.

Users skim your pages by its headings. It is important to use headings to show the document structure.

`<h1>` headings should be used for main headings, followed by `<h2>` headings, then the less important `<h3>`, and so on.

Note: Use HTML headings for headings only. Don't use headings to make text BIG or bold.

Bigger Headings - Each HTML heading has a **default size**. However, you can specify the size for any heading with the style attribute:

Example

```
<h1 style="font-size:60px;">Heading 1</h1>
```

HTML Horizontal Rules - The `<hr>` tag defines a thematic break in an HTML page, and is most often displayed as a horizontal rule.

The `<hr>` element is used to separate content (or define a change) in an HTML page:

Example

```
<h1>Science</h1>
<p>Computing</p>
<hr>
<h2>Art</h2>
<hr> 10/01/19-18$22
```

The HTML `<head>` Element - The HTML `<head>` element has nothing to do with HTML headings.

The `<head>` element is a container for metadata. HTML metadata is data about the HTML document.

Metadata is not displayed.

The `<head>` element is placed between the `<html>` tag and the `<body>` tag:

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>My First HTML</title>
  <meta charset="UTF-8">
</head>
<body>...
```

Note: Metadata typically define the document title, character set, styles, links, scripts, and other meta information.

How to View HTML Source? - View HTML Source Code:

To find out, right-click in the page and select "View Page Source" (in Chrome) or "View Source" (in IE), or similar in other browsers. This will open a window containing the HTML source code of the page.

Inspect an HTML Element: Right-click on an element (or a blank area), and choose "Inspect" or "Inspect Element" to see what elements are made up of (you will see both the HTML and the CSS). You can also edit the HTML or CSS on-the-fly in the Elements or Styles panel that opens.

HTML Paragraphs

The HTML `<p>` element defines a paragraph:

Example

```
<p>System Analysis.</p>
```

```
<p>System Design.</p>
```

Note: Browsers automatically add some white space (a margin) before and after a paragraph.

HTML Display

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 any extra spaces and extra lines when the page is displayed:

Example

```
<p>WAN (Wide Area Network) is a computer network that covers a broad area. </p>
```

```
<p>Computers connected to a wide-area network are often connected through public networks, such as the telephone system. They can also be connected through leased lines or satellites </p>
```

Don't Forget the End Tag

Most browsers will display HTML correctly even if you forget the end tag:

Example

```
<p> Internet is a good example of a WAN.
```

```
<p> Experiences more data transmission errors as compared to LAN.
```

The example above will work in most browsers, but do not rely on it.

Note: Dropping the end tag can produce unexpected results or errors.

HTML Line Breaks - The HTML `
` element defines a line break.

Use `
` if you want a line break (a new line) without starting a new paragraph:

Example

```
<p> Have a large geographical range generally spreading <br> across boundaries and <br> need leased telecommunication lines.</p>
```

The `
` tag is an empty tag, which means that it has no end tag.

The HTML `<pre>` Element - The HTML `<pre>` element defines preformatted text.

The text inside a `<pre>` element is displayed in a fixed-width font (usually Courier), and it preserves both spaces and line breaks:

Example

```
<pre>
```

```
Components:
```

```
Fault Tolerance:
```

```
Data Transmission Error:
```

```
Ownership:
```

```
Set-up costs:
```

```
</pre>
```

```
10/01/19-18S24
```

Example

I am Red

I am Blue

I am Big

The HTML Style Attribute - Setting the style of an HTML element, can be done with the style attribute.

The HTML style attribute has the following syntax:

```
<tagname style="property:value;">
```

The property is a CSS property. The value is a CSS value.

HTML Background Color - The **background-color** property defines the background color for an HTML element.

This example sets the background color for a page to darkblue:

Example

```
<body style="background-color: darkblue;">  
<h1>This is a heading</h1>  
<p>This is a paragraph.</p>  
</body>
```

HTML Text Color - The color property defines the text color for an HTML element:

Example

```
<h1 style="color:blue;">White box testing</h1>  
<p style="color:red;">Black box testing</p>
```

HTML Fonts - The font-family property defines the font to be used for an HTML element:

Example

```
<h1 style="font-family:verdana;"> Alfa testing </h1>  
<p style="font-family:courier;"> Beta testing</p>
```

HTML Text Size - The font-size property defines the text size for an HTML element:

Example

```
<h1 style="font-size:300%;">Top-down testing</h1>  
<p style="font-size:160%;">Bottom-up testing</p>
```

HTML Text Alignment - The text-align property defines the horizontal text alignment for an HTML element:

Example

```
<h1 style="text-align:center;">Unit testing</h1>  
<p style="text-align:center;">Acceptance test</p>
```

09/01/2019-18S23

HTML Text Formatting

Text Formatting - **This text is bold**, *This text is italic*, This is _{subscript} and ^{superscript}

HTML Formatting Elements - HTML defines special elements for defining text with a special meaning.

HTML uses elements like `` and `<i>` for formatting output, like bold or italic text.

Formatting elements were designed to display special types of text:

| HTML <code></code> Element | HTML <code></code> Element |
|--|--|
| The HTML <code></code> element defines bold text, without any extra importance. | The HTML <code></code> element defines strong text, with added semantic "strong" importance. |
| Example | Example |
| <code>This text is bold</code> | <code>This text is strong</code> |

| HTML <code><i></code> Element | HTML <code></code> Element |
|--|---|
| The HTML <code><i></code> element defines italic text, without any extra importance. | The HTML <code></code> element defines emphasized text, with added semantic importance. |
| Example | Example |
| <code><i>This text is italic</i></code> | <code>This text is emphasized</code> |

Note: Browsers display `` as ``, and `` as `<i>`. However, there is a difference in the meaning of these tags: `` and `<i>` defines bold and italic text, but `` and `` means that the text is "important".

| HTML <code><small></code> Element | HTML <code><mark></code> Element |
|--|---|
| The HTML <code><small></code> element defines smaller text: | The HTML <code><mark></code> element defines marked or highlighted text: |
| Example | Example |
| <code><h2>HTML <small>Small</small> Formatting</h2></code> | <code><h2>HTML <mark>Marked</mark> Formatting</h2></code> |
| HTML <code></code> Element | HTML <code><ins></code> Element |
| The HTML <code></code> element defines deleted (removed) text. | The HTML <code><ins></code> element defines inserted (added) text. |
| Example | Example |
| <code><p>My NRIC color is blue red.</p></code> | <code><p>Apple <ins>color</ins> is red.</p></code> |

| HTML <code><sub></code> Element | HTML <code><sup></code> Element |
|--|--|
| The HTML <code><sub></code> element defines subscripted text. | The HTML <code><sup></code> element defines superscripted text. |
| Example | Example |
| <code><p>This is <sub>subscripted</sub> text.</p></code> | <code><p>This is <sup>superscripted</sup> text.</p></code> |

HTML Quotation and Citation Elements

Quotation

Here is a quote from jpjc.moe.edu.sg's website:

The merger may be perceived as a destructive force that demolishes the achievements of both colleges by those whose response to the inevitable change is confined to achievements of the past. For those of us who recognise the transience of past glory and see our humble role in the evolution of education, we draw strength from our past, from our supporters, from our team members and most importantly, from our shared aspiration for the future.

HTML `<q>` for Short Quotations - The HTML `<q>` element defines a short quotation.

Browsers usually insert quotation marks around the `<q>` element.

Example

`<p> JPJC Staff <q> The Mission of the Education Service is to mould the future of the nation, by moulding the people who will determine the future of the nation. </q> </p>`

HTML `<blockquote>` for Quotations - The HTML `<blockquote>` element defines a section that is quoted from another source.

Browsers usually indent `<blockquote>` elements.

Example

`<p> Here is a quote from jpjc.moe.edu.sg's website: </p>`

`<blockquote cite=" http://jpjc.moe.edu.sg/about-jpjc/college-history">`

No roadmaps can guide us on this journey. But our shared beliefs will act as our compass as we work hard to retain the best of both colleges and from them, we fashion the *Best of the West* – the shared aspiration articulated by the staff of Jurong Pioneer Junior College at our inaugural Staff Meeting.

`</blockquote>`

| HTML <code><abbr></code> for Abbreviations | HTML <code><address></code> for Contact Information |
|---|---|
| The HTML <code><abbr></code> element defines an abbreviation or an acronym. | The HTML <code><address></code> element defines contact information (author/owner) of a document or an article. |
| Marking abbreviations can give useful information to browsers, translation systems and search-engines. | The <code><address></code> element is usually displayed in italic. Most browsers will add a line break before and after the element. |
| Example | Example |
| <code><p>The <code><abbr title="Jurong Pioneer Junior College"></code>JPJC<code></abbr></code> was founded in 2019.<code></p></code></code> | <code><address>21<code> </code> Teck Whye Walk<code> </code>Singapore 688258<code> </code><code></address></code></code> |

| HTML <code><cite></code> for Work Title | HTML <code><bdo></code> for Bi-Directional Override |
|--|--|
| The HTML <code><cite></code> element defines the title of a work. | The HTML <code><bdo></code> element defines bi-directional override. |
| Browsers usually display <code><cite></code> elements in italic. | The <code><bdo></code> element is used to override the current text direction. |
| Example | Example |
| <code><p><code><cite></code>The System Analysis<code></cite></code> by Paik Poh Leong.<code></p></code></code> | <code><bdo dir="rtl"> The System Analysis <code></bdo></code></code> |

HTML Comments

Comment tags are used to insert comments in the HTML source code.

HTML Comment Tags

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

```
<!-- Write your comments here -->
```

Notice that there is an exclamation point (!) in the opening tag, but not in the closing tag.

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

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

Example

```
<!-- This is a comment -->
```

```
<p>System Implementation.</p>
```

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

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

```
<!-- Do not display this at the moment
```

```

```

```
-->
```

HTML Colors

HTML colors are specified using predefined color names, or RGB, HEX, HSL, RGBA, HSLA values.

Color Names - In HTML, a color can be specified by using a color name:

Tomato, Orange, DarkBlue, LightGreen, Gray, Violet, LightGray

HTML supports 140 standard color names.

Background Color - You can set the background color for HTML elements: Testing Strategy

Example

```
<h1 style="background-color:LightBlue;">Problem Definition</h1>
<p style="background-color:Silver;">Problem Statement</p>
```

Text Color - You can set the color of text: Black Box testing

Example

```
<h1 style="color:Tomato;"> Problem Definition </h1>
<p style="color:DodgerBlue;"> Problem Statement </p>
```

Border Color - You can set the color of borders:

Example

```
<h1 style="border:2px solid Red;"> User</h1>
```

User

Color Values

In HTML, colors can also be specified using RGB values, HEX values, HSL values, RGBA values, and HSLA values:

Same as color name "Tomato":

rgb(255, 99, 71), #ff6347, hsl(9, 100%, 64%)

Same as color name "Tomato", but 50% transparent:

rgba(255, 99, 71, 0.5), hsla(9, 100%, 64%, 0.5)

Example

```
<h1 style="background-color:rgb(255, 99, 71);">...</h1>
<h1 style="background-color:#ff6347;">...</h1>
<h1 style="background-color:hsl(9, 100%, 64%;">...</h1>
<h1 style="background-color:rgba(255, 99, 71, 0.5);">...</h1>
<h1 style="background-color:hsla(9, 100%, 64%, 0.5);">...</h1>
```

RGB Value

In HTML, a color can be specified as an RGB value, using this formula: **rgb(red, green, blue)**

Each parameter (red, green, and blue) defines the intensity of the color between 0 and 255.

For example, rgb(255, 0, 0) is displayed as red, because red is set to its highest value (255) and the others are set to 0.

To display the color black, all color parameters must be set to 0, like this: rgb(0, 0, 0).

To display the color white, all color parameters must be set to 255, like this: rgb(255, 255, 255).

Example

rgb(255, 0, 0), rgb(0, 0, 255), rgb(60, 179, 113), rgb(238, 130, 238), rgb(255, 165, 0), rgb(106, 90, 205)

Shades of gray are often defined using equal values for all the 3 light sources:

Example

rgb(0, 0, 0), rgb(60, 60, 60), rgb(120, 120, 120), rgb(180, 180, 180), rgb(240, 240, 240), rgb(255, 255, 255)

HEX Value - In HTML, a color can be specified using a hexadecimal value in the form: **#rrggbb**

Where rr (red), gg (green) and bb (blue) are hexadecimal values between 00 and ff (same as decimal 0-255).

For example, #ff0000 is displayed as red, because red is set to its highest value (ff) and the others are set to the lowest value (00).

Example

#ff0000, #0000ff, #3cb371, #ee82ee, #ffa500, #6a5acd

Shades of gray are often defined using equal values for all the 3 light sources:

Example

#000000, #3c3c3c, #787878, #b4b4b4, #f0f0f0, #ffffff

HSL Value - In HTML, a color can be specified using hue, saturation, and lightness (HSL) in the form:

hsl(hue, saturation, lightness)

Hue is a degree on the color wheel from 0 to 360. 0 is red, 120 is green, and 240 is blue.

Saturation is a percentage value, 0% means a shade of gray, and 100% is the full color.

Lightness is also a percentage, 0% is black, 50% is neither light nor dark, 100% is white

Example

hsl(0, 100%, 50%), hsl(240, 100%, 50%), hsl(147, 50%, 47%), hsl(300, 76%, 72%), hsl(248, 53%, 58%)

Saturation - Saturation can be described as the intensity of a color.

100% is pure color, no shades of gray

50% is 50% gray, but you can still see the color.

0% is completely gray, you can no longer see the color.

Example

hsl(0, 100%, 50%), hsl(0, 80%, 50%), hsl(0, 60%, 50%), hsl(0, 40%, 50%), hsl(0, 20%, 50%), hsl(0, 0%, 50%)

Lightness - The lightness of a color can be described as how much light you want to give the color, where 0% means no light (black), 50% means 50% light (neither dark nor light) 100% means full lightness (white).

Example

hsl(0, 100%, 0%), hsl(0, 100%, 25%), hsl(0, 100%, 50%), hsl(0, 100%, 75%), hsl(0, 100%, 100%)

Shades of gray are often defined by setting the hue and saturation to 0, and adjust the lightness from 0% to 100% to get darker/lighter shades:

Example

hsl(0, 0%, 0%), hsl(0, 0%, 24%), hsl(0, 0%, 47%), hsl(0, 0%, 71%), hsl(0, 0%, 94%), hsl(0, 0%, 100%)

RGBA Value - RGBA color values are an extension of RGB color values with an alpha channel - which specifies the opacity for a color.

An RGBA color value is specified with: **rgba(red, green, blue, alpha)**

The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all):

Example

rgba(255, 99, 71, 0), rgba(255, 99, 71, 0.2), rgba(255, 99, 71, 0.4), rgba(255, 99, 71, 0.6)

HSLA Value - HSLA color values are an extension of HSL color values with an alpha channel - which specifies the opacity for a color.

An HSLA color value is specified with: **hsla(hue, saturation, lightness, alpha)**

The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all):

Example

**hsla(9, 100%, 64%, 0), hsla(9, 100%, 64%, 0.2), hsla(9, 100%, 64%, 0.4), hsla(9, 100%, 64%, 0.6)
hsla(9, 100%, 64%, 0.8), hsla(9, 100%, 64%, 1)**

HTML Styles - CSS

CSS = Styles and Colors

Manipulate Text, Colors, Boxes

Styling HTML with CSS

CSS stands for **C**ascading **S**tyle **S**heets.

CSS describes *how HTML elements are to be displayed on screen, paper, or in other media.*

CSS **saves a lot of work**. It can control the layout of multiple web pages all at once.

CSS can be added to HTML elements in 3 ways:

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

The most common way to add CSS, is to keep the styles in separate CSS files.

Inline CSS - An inline CSS is used to apply a unique style to a single HTML element.

An inline CSS uses the style attribute of an HTML element.

This example sets the text color of the <h1> element to blue:

Example

```
<h1 style="color:blue;">Checking which part is blue</h1>
```

Internal CSS - An internal CSS is used to define a style for a single HTML page.

An internal CSS is defined in the **<head>** section of an HTML page, within a <style> element:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
Body {
background-color: lightblue;
}
h1 {
color: blue;
}
p {
color: red;
}
</style>
</head>
<body>
<h1>Networking</h1>
<p>Packet Switching</p>
</body>
</html>
```

External CSS

An external style sheet is used to define the style for many HTML pages.

With an external style sheet, you can change the look of an entire web site, by changing one file!

To use an external style sheet, add a link to it in the **<head>** section of the HTML page:

Example

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
<h1>Networking</h1>
<p>Packet Switching</p>
</body>
</html>
```

An external style sheet can be **written in any text editor**. The file must not contain any HTML code, and must be saved with a **.css** extension.

Here is how the "styles.css" looks:

```
body {
  background-color: lightblue;
}
h1 {
  color: blue;
}
p {
  color: red;
}
```

CSS Fonts - The CSS color property defines the text color to be used.

The CSS font-family property defines the font to be used.

The CSS font-size property defines the text size to be used.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  color: blue;
  font-family: verdana;
  font-size: 300%;
}
p {
  color: red;
  font-family: courier;
  font-size: 160%;
}
</style>
</head>
<body>
<h1>Networking</h1>
<p>Packet Switching</p>
</body>
</html>
```

CSS Border - The CSS border property defines a border around an HTML element:

Example

```
p {  
  border: 1px solid lightblue;  
}
```

CSS Padding - The CSS padding property defines a padding (space) between the text and the border:

Example

```
p {  
  border: 1px solid lightblue;  
  padding: 30px;  
}
```

CSS Margin - The CSS margin property defines a margin (space) outside the border:

Example

```
p {  
  border: 1px solid lightblue;  
  margin: 50px;  
}
```

The id Attribute - To define a specific style for one special element, add an id attribute to the element:

```
<p id="p01">Circuit Switching</p>
```

then define a style for the element with the specific id:

Example

```
#p01 {  
  color: blue;  
}
```

Note: The id of an element should be unique within a page, so the id selector is used to select one unique element!

The class Attribute - To define a style for a special type of elements, add a **class attribute** to the element:

```
<p class="error"> Circuit Switching </p>
```

then define a style for the elements with the specific class:

Example

```
p.error {  
  color: red;  
}
```

External References - External style sheets can be referenced with a full URL or with a path relative to the current web page.

This example uses a full URL to link to a style sheet:

Example

```
<link rel="stylesheet" href="http://jpjc.moe.edu.sg /html/styles.css">
```

This example links to a style sheet located in the html folder on the current web site:

Example

```
<link rel="stylesheet" href="/html/styles.css">
```

This example links to a style sheet located in the same folder as the current page:

Example

```
<link rel="stylesheet" href="styles.css">
```

HTML Links

Links are found in nearly all web pages. Links allow users to click their way from page to page.

HTML Links – Hyperlinks - HTML links are hyperlinks.

You can click on a link and jump to another document.

When you move the mouse over a link, the mouse arrow will turn into a little hand.

Note: A link does not have to be text. It can be an image or any other HTML element.

HTML Links - Syntax

In HTML, links are defined with the `<a>` tag: `link text`

Example

```
<a href="http://jpic.moe.edu.sg/html/">Visit our School</a>
```

The href attribute specifies the destination address (`http://jpic.moe.edu.sg/html/`) of the link.

The link text is the visible part.

Clicking on the link text will send you to the specified address.

Note: Without a forward slash on subfolder addresses, you might generate two requests to the server. Many servers will automatically add a forward slash to the address, and then create a new request.

Local Links

The example above used an absolute URL (A full web address).

A local link (link to the same web site) is specified with a relative URL (without `http://www....`).

Example

```
<a href="Chapter_5.asp">Chapter 5 Asynchronous Transmission</a>
```

HTML Link Colors

By default, a link will appear like this (in all browsers):

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

You can change the default colors, by using styles:

Example

```
<style>
a:link {
  color: green;
  background-color: transparent;
  text-decoration: none;
}
a:visited {
  color: pink;
  background-color: transparent;
  text-decoration: none;
}
a:hover {
  color: red;
  background-color: transparent;
  text-decoration: underline;
}
a:active {
  color: yellow;
  background-color: transparent;
  text-decoration: underline;
}
</style>
```


HTML Links - The target Attribute - The target attribute specifies where to open the linked document.

The target attribute can have one of the following values:

- `_blank` - Opens the linked document in a new window or tab
- `_self` - Opens the linked document in the same window/tab as it was clicked (this is default)
- `_parent` - Opens the linked document in the parent frame
- `_top` - Opens the linked document in the full body of the window
- `framename` - Opens the linked document in a named frame

This example will open the linked document in a new browser window/tab:

Example

```
<a href="http://jpjc.moe.edu.sg/" target="_blank">Visit Parent Support Group! </a>
```

Tip: If your webpage is locked in a frame, you can use `target="_top"` to break out of the frame:

Example

```
<a href="http://jpjc.moe.edu.sg/html/" target="_top">Subject Combinations</a>
```

HTML Links - Image as Link

It is common to use images as links:

Example

```
<a href="default.asp">
  
</a>
```

Note: `border:0;` is added to prevent IE9 (and earlier) from displaying a border around the image (when the image is a link).

HTML Links - Create a Bookmark

HTML bookmarks are used to allow readers to jump to specific parts of a Web page.

Bookmarks can be useful if your webpage is very long.

To make a bookmark, you must first create the bookmark, and then add a link to it.

When the link is clicked, the page will scroll to the location with the bookmark.

Example

First, create a bookmark with the id attribute:

```
<h2 id="T4"> Computing Networking Tutorial 4</h2>
```

Then, add a link to the bookmark ("Jump to Tutorial 4"), from within the same page:

```
<a href="#T4">Jump to Tutorial 4</a>
```

Or, add a link to the bookmark ("Jump to Tutorial 4"), from another page:

Example

```
<a href="html_demo.html#T4">Jump to Tutorial 4</a>
```

External Paths

External pages can be referenced with a full URL or with a path relative to the current web page.

This example uses a full URL to link to a web page:

Example

```
<a href="http://www.jpjc.com/html/default.asp">Subject Combination</a>
```

This example links to a page located in the html folder on the current web site:

Example

```
<a href="/html/default.asp"> Subject Combination </a>
```

This example links to a page located in the same folder as the current page:

Example

```
<a href="default.asp"> Subject Combination </a>
```

HTML Images

Images can improve the design and the appearance of a web page.

Example

```

```

Example

```

```

HTML Images Syntax - In HTML, images are defined with the tag.

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

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

The alt Attribute - The alt attribute provides an alternate text for an image, if the user for some reason cannot view it (because of slow connection, an error in the src attribute, or if the user uses a screen reader). The value of the alt attribute should describe the image:

Example

```

```

If a browser cannot find an image, it will display the value of the alt attribute:

Example

```

```

Note: The alt attribute is required. A web page will not validate correctly without it.

Image Size - Width and Height

You can use the style attribute to specify the width and height of an image.

Example

```

```

Alternatively, you can use the width and height attributes:

Example

```

```

The width and height attributes always defines the width and height of the image in pixels.

Note: Always specify the width and height of an image. If width and height are not specified, the page might flicker while the image loads.

Width and Height, or Style?

Both the width, height, and style attributes are valid in HTML5.

However, we suggest using the style attribute. It prevents styles sheets from changing the size of images:

Example

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<style>
```

```
img {
```

```
  width:100%;
```

```
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```

```

```

```

```
</body>
```

```
</html>
```

Images in Another Folder - If not specified, the browser expects to find the image in the same folder as the web page. However, it is common to store images in a sub-folder. You must then include the folder name in the src attribute:

Example

```

```

Animated Images - HTML allows animated GIFs:

Example

```

```

Image as a Link - To use an image as a link, put the tag inside the <a> tag:

Example

```
<a href="default.asp">
  
</a>
```

Image Floating - Use the CSS float property to let the image float to the right or to the left of a text:

Example

```
<p>
The image will float to the right of the text.</p>
<p>
The image will float to the left of the text.</p>
```

Image Maps - Use the <map> tag to define an image-map. An image-map is an image with clickable areas. In the image below, click on the computer, a keyboard, or the mouse:

Example

Hardware

```

<map name="workmap">
  <area shape="rect" coords="34,44,270,350" alt="Computer" href="computer.htm">
  <area shape="rect" coords="290,172,333,250" alt="keyboard" href="phone.htm">
  <area shape="circle" coords="337,300,44" alt="mouse" href=" mouse.htm">
</map>
```

The name attribute of the <map> tag is associated with the 's usemap attribute and creates a relationship between the image and the map.

The <map> tag contains a number of <area> tags, that defines the clickable areas in the image-map.

Background Image - To add a background image on an HTML element, use the CSS property background-image:

Example

To add a background image on a web page, specify the background-image property on the BODY element:

```
<body style="background-image:url(17S23.jpg)">
<h2>Background Image of CTGP 17S23</h2>
</body>
```

Example

To add a background image on a paragraph, specify the background-image property on the P element:

```
<body>
<p style="background-image:url(17S23.jpg)">
...
</p>
</body>
```

The <picture> Element

HTML5 introduced the <picture> element to add more flexibility when specifying image resources.

The <picture> element contains a number of <source> elements, each referring to different image sources.

This way the browser can choose the image that best fit the current view and/or device.

Each <source> element have attributes describing when their image is the most suitable.

The browser will use the first <source> element with matching attribute values, and ignore any following <source> elements.

Example

Show one picture if the browser window (viewport) is a minimum of 650 pixels, and another image if not, but larger than 465 pixels.

<picture>

<source media="(min-width: 650px)" srcset="img_Circuit_Switching.jpg">

<source media="(min-width: 465px)" srcset="img_Packet_Switching.jpg">

</picture>

Note: Always specify an element as the last child element of the <picture> element. The element is used by browsers that do not support the <picture> element, or if none of the <source> tags matched.

HTML Screen Readers

A screen reader is a software program that reads the HTML code, converts the text, and allows the user to "listen" to the content. Screen readers are useful for people who are blind, visually impaired, or learning disabled.

HTML Table Example

| Subject | Teachers | Civic Tutor Group |
|-----------|----------------|-------------------|
| Computing | Lai Wai Liang | 17S22 |
| Physics | Chueng Jun Jie | 17S23 |

Defining an HTML Table - An HTML table is defined with the `<table>` tag.

Each table row is defined with the `<tr>` tag. A table header is defined with the `<th>` tag. By default, table headings are bold and centered. A table data/cell is defined with the `<td>` tag.

Example

```
<table style="width:100%">
  <tr>
    <th>Subject</th>
    <th>Teachers</th>
    <th>Civic Tutor Group</th>
  </tr>
  <tr>
    <td>Computing</td>
    <td>Lai Wai Liang</td>
    <td>17S22</td>
  </tr>
  <tr>
    <td>Physics</td>
    <td>Chueng Jun Jie</td>
    <td>17S23</td>
  </tr>
</table>
```

Note: The `<td>` elements are the data containers of the table.

They can contain all sorts of HTML elements; text, images, lists, other tables, etc.

HTML Table - Adding a Border

If you do not specify a border for the table, it will be displayed without borders.

A border is set using the CSS ***border*** property:

Example

```
table, th, td {
  border: 1px solid black;
}
```

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

HTML Table - Collapsed Borders

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

Example

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

HTML Table - Adding 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;
}
```

HTML Table - Left-align Headings - By default, table headings are bold and centered.

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

Example

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

HTML Table - Adding Border Spacing - Border spacing specifies the space between the cells.

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

Example

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

Note: If the table has collapsed borders, border-spacing has no effect.

HTML Table - Cells that Span Many Columns - To make a cell span more than one column, use the **colspan** attribute:

Example

```
<table style="width:100%">
  <tr>
    <th>Name</th>
    <th colspan="2">Telephone</th>
  </tr>
  <tr>
    <td>Paik Poh Leong</td>  <td>1800-9992418</td>  <td>9999012</td>
  </tr>
</table>
```

HTML Table - Cells that Span Many Rows - To make a cell span more than one row, use the **rowspan** attribute:

Example

```
<table style="width:100%">
  <tr>
    <th>Name:</th>
    <td> Paik Poh Leong </td>
  </tr>
  <tr>
    <th rowspan="2">Telephone:</th>
    <td>1800-9992418</td>
  </tr>
  <tr>
    <td>9999012</td>
  </tr>
</table>
```

HTML Table - Adding a Caption - To add a caption to a table, use the `<caption>` tag:

Example

```
<table style="width:100%">
  <caption>Monthly Expenses</caption>
  <tr>
    <th>Month</th>
    <th>Expenses </th>
  </tr>
  <tr>
    <td>January</td>
    <td>$1500</td>
  </tr>
  <tr>
    <td>February</td>
    <td>$5000</td>
  </tr>
</table>
```

Note: The `<caption>` tag must be inserted immediately after the `<table>` tag.

A Special Style for One Table - To define a special style for a special table, add an id attribute to the table:

Example

```
<table id="s01">
  <tr>
    <th>Subject</th>
    <th>Name</th>
    <th>CTGP</th>
  </tr>
  <tr>
    <td>Computing</td>
    <td>Lai Wai Liang</td>
    <td>17S22</td>
  </tr>
</table>
```

Now you can define a special style for this table:

```
table#s01 {
  width: 100%;
  background-color: #f1f1c1;
}
```

And add more styles:

```
table#s01 tr:nth-child(even) {
  background-color: #eee;
}
table#s01 tr:nth-child(odd) {
  background-color: #fff;
}
table#s01 th {
  color: white;
  background-color: black;
}
```

HTML List Example

| | |
|--|--|
| An Unordered List: <ul style="list-style-type: none"> • Mathematics • Physics • Computing • Economics | An Ordered List: <ol style="list-style-type: none"> 1. Mathematics 2. Physics 3. Computing 4. Economics |
|--|--|

Unordered HTML List - An unordered list starts with the `` tag. Each list item starts with the `` tag. The list items will be marked with bullets (small black circles) by default:

Example

```
<ul>
  <li>Mathematics</li>
  <li>Computing</li>
  <li>Economics </li>
</ul>
```

Unordered HTML List - Choose List Item Marker -The CSS `list-style-type` property is used to define the style of the list item marker:

| Value | Description |
|--------|---|
| disc | Sets the list item marker to a bullet (default) |
| circle | Sets the list item marker to a circle |
| square | Sets the list item marker to a square |
| none | The list items will not be marked |

Example - Disc

```
<ul style="list-style-type:disc">
  <li>Mathematics</li>
  <li>Computing</li>
  <li>Economics </li>
</ul>
```

Example - Circle

```
<ul style="list-style-type:circle">
  <li>Mathematics</li>
  <li>Computing</li>
  <li>Economics </li>
</ul>
```

Example - Square

```
<ul style="list-style-type:square">
  <li>Mathematics</li>
  <li>Computing</li>
  <li>Economics </li>
</ul>
```

Example - None

```
<ul style="list-style-type:none">
  <li>Mathematics</li>
  <li>Computing</li>
  <li>Economics </li>
</ul>
```


Ordered HTML List - An ordered list starts with the `` tag. Each list item starts with the `` tag. The list items will be marked with numbers by default:

Example

```
<ol>
  <li>Mathematics</li>
  <li> Computing</li>
  <li> Economics </li>
</ol>
```

Ordered HTML List - The Type Attribute - The type attribute of the `` tag, defines the type of the list item marker:

| Type | Description |
|----------|--|
| type="1" | The list items will be numbered with numbers (default) |
| type="A" | The list items will be numbered with uppercase letters |
| type="a" | The list items will be numbered with lowercase letters |
| type="I" | The list items will be numbered with uppercase roman numbers |
| type="i" | The list items will be numbered with lowercase roman numbers |

| Numbers | Uppercase Letters | Lowercase Letters | Uppercase Roman Numbers | Lowercase Roman Numbers |
|---|---|---|---|---|
| <code><ol type="1"></code> | <code><ol type="A"></code> | <code><ol type="a"></code> | <code><ol type="I"></code> | <code><ol type="i"></code> |
| <code>Mathematics</code> | <code>Mathematics</code> | <code>Mathematics</code> | <code>Mathematics</code> | <code>Mathematics</code> |
| <code> Computing</code> | <code> Computing</code> | <code> Computing</code> | <code> Computing</code> | <code> Computing</code> |
| <code> Economics </code> | <code> Economics </code> | <code> Economics </code> | <code> Economics </code> | <code> Economics </code> |
| <code></code> | <code></code> | <code></code> | <code></code> | <code></code> |

HTML Description Lists - HTML also supports description lists.

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

The `<dl>` tag defines the description list, the `<dt>` tag defines the term (name), and the `<dd>` tag describes each term:

Example

```
<dl>
  <dt>H2</dt>
  <dd>Mathematics </dd>
  <dt>H1</dt>
  <dd>- Economics </dd>
</dl>
```

Nested HTML Lists - List can be nested (lists inside lists):

Example

```
<ul>
  <li>JC2</li>
  <li>Science
    <ul>
      <li> Mathematics </li>
      <li> Computing </li>
    </ul>
  </li>
  <li>Art</li>
</ul>
```

Note: List items can contain new list, and other HTML elements, like images and links, etc.

Horizontal Lists - HTML lists can be styled in many different ways with CSS.

One popular way is to style a list horizontally, to create a menu:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
  background-color: #333333;
}
li {
  float: left;
}
li a {
  display: block;
  color: white;
  text-align: center;
  padding: 16px;
  text-decoration: none;
}
li a:hover {
  background-color: #111111;
}
</style>
</head>
<body>
<ul>
  <li><a href="#SubComp">Subject Combination</a></li>
  <li><a href="#DateOfRelease ">Date of Release</a></li>
  <li><a href="#contact">Contact</a></li>
  <li><a href="#FillForm">How to fill in the form</a></li>
</ul>
</body>
</html>
```

HTML Block and Inline Elements

Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is block or inline.

Block-level Elements -

A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

The **<div>** element is a block-level element.

Example

```
<div>Hello</div>
```

```
<div>Computing classes</div>
```

Block level elements in HTML:

| | | | | | | | |
|-----------|------------|--------------|--------------|------------|---------|-----------|----------|
| <address> | <article> | <aside> | <blockquote> | <canvas> | <dd> | <div> | <dl> |
| <dt> | <fieldset> | <figcaption> | <figure> | <footer> | <form> | <h1>-<h6> | <header> |
| <hr> | | <main> | <nav> | <noscript> | | <output> | <p> |
| <pre> | <section> | <table> | <tfoot> | | <video> | | |

Inline Elements - An inline element does not start on a new line and only takes up as much width as necessary.

This is an inline **** element inside a paragraph.

Example

```
<span>Hello</span>
```

```
<span> Computing classes </span>
```

Inline elements in HTML:

| | | | | | | | | |
|----------|------------|-----------|----------|----------|---------|--------|----------|--------|
| <a> | <abbr> | <acronym> | | <bdo> | <big> | | <button> | <cite> |
| <code> | <dfn> | | <i> | | <input> | <kbd> | <label> | <map> |
| <object> | <q> | <samp> | <script> | <select> | <small> | | | <sub> |
| <sup> | <textarea> | <time> | <tt> | <var> | | | | |

The <div> Element - The <div> element is often used as a container for other HTML elements.

The **<div>** element has no required attributes, but both style and class are common.

When used together with CSS, the <div> element can be used to style blocks of content:

Example

```
<div style="background-color: black; color: white; padding:20px;">
```

```
  <h2>Networking</h2>
```

```
  <p>Network topology</p>
```

```
</div>
```

The Element - The element is often used as a container for some text.

The **** element has no required attributes, but both style and class are common.

When used together with CSS, the element can be used to style parts of the text:

Example

```
<h1>Study <span style="color:red">Computing</span> is important</h1>
```

HTML Grouping Tags

| Tag | Description |
|--------|---|
| <div> | Defines a section in a document (block-level) |
| | Defines a section in a document (inline) |

HTML The class Attribute

Using The class Attribute - The class attribute specifies one or more class names for an HTML element.

The class name can be used by CSS and JavaScript to perform certain tasks for elements with the specified class name.

Example

Using CSS to style all elements with the class name "ctgp":

```
<style>
.ctgp {
  background-color: tomato;
  color: white;
  padding: 10px;
}
</style>

<h2 class="ctgp">17S09</h2>
<p>Mathematics, Chemistry, Computing. </p>
<h2 class="ctgp">17S22</h2>
<p>Mathematics, Physics, Computing. </p>
<h2 class="ctgp">17S23</h2>
<p>Mathematics, Physics, Computing. </p>
```

The class attribute can be used on any HTML element.

Using The class Attribute in JavaScript

JavaScript can access elements with a specified class name by using the `getElementsByClassName()` method:

Example

When a user clicks on a button, hide all elements with the class name "ctgp":

```
<script>
function ctgpFunction() {
  var CTGP = document.getElementsByClassName("ctgp");
  for (var i = 0; i < CTGP.length; i++) {
    CTGP[i].style.display = "none";
  }
}
</script>
```

Multiple Classes

HTML elements can have more than one class name, each class name must be separated by a space.

Example

Style elements with the class name "**ctgp**", also style elements with the class name "**stream**":

```
<h2 class="ctgp stream">Science</h2>
<h2 class="ctgp">17S09</h2>
<h2 class="ctgp">17S22</h2>
```

In the example above, the first h2 element belongs to both the "**ctgp**" class and the "**stream**" class.

Same Class, Different Tag

Different tags, like `<h2>` and `<p>`, can have the same class name and thereby share the same style:

Example

```
<h2 class="ctgp"> 17S09 </h2>
<p class="ctgp">17S09 is a chemistry class</p>
```

HTML Iframes

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

Iframe Syntax - An HTML iframe is defined with the <iframe> tag:

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

The src attribute specifies the URL (web address) of the inline frame page.

Iframe - Set Height and Width - Use the height and width attributes to specify the size of the iframe. The attribute values are specified in pixels by default, but they can also be in percent (like "80%").

Example

```
<iframe src="subject_iframe.htm" height="200" width="300"></iframe>
```

Iframe - Remove the Border - By default, an iframe has a border around it.

To remove the border, add the style attribute and use the CSS border property:

Example

```
<iframe src="subject_iframe.htm" style="border:none;"></iframe>
```

With CSS, you can also change the size, style and color of the iframe's border:

Example

```
<iframe src="subject_iframe.htm" style="border:2px solid grey;"></iframe>
```

Iframe - Target for a Link - An iframe can be used as the target frame for a link.

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

Example

```
<iframe src="subject_iframe.htm" name="iframe_a"></iframe>
```

```
<p><a href="https://www.jpjc.moe.edu.sg" target="iframe_a">Subject Combinations</a></p>
```

JavaScript makes HTML pages more dynamic and interactive.

Example

```
<!DOCTYPE html>
<html>
<body>
<h1>My First JavaScript</h1>
<button type="button"
onclick="document.getElementById('testing').innerHTML = Date()">
Click me to display Date and Time.</button>
<p id="testing"></p>
</body>
</html>
```

The HTML <script> Tag - The <script> tag is used to define a client-side script (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.

To select an HTML element, JavaScript very often use the document.getElementById(id) method.

This JavaScript example writes "Hello JavaScript!" into an HTML element with id="demo":

Example

```
<script>
document.getElementById("testing").innerHTML = "Hello Welcome!";
</script>
```

A Taste of JavaScript

Here are some examples of what JavaScript can do:

JavaScript can change HTML content

```
document.getElementById("testing").innerHTML = "Hello Computing!";
```

JavaScript can change HTML styles

```
document.getElementById("testing").style.fontSize = "45px";
document.getElementById("testing").style.color = "silver";
document.getElementById("testing").style.backgroundColor = "blue";
```

JavaScript can change HTML attributes

```
document.getElementById("image").src = "picture.gif";
```

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 scripts:

Example

```
<script>
document.getElementById("demo").innerHTML = "Hello Computing!";
</script>
<noscript>Sorry, your browser does not support JavaScript!</noscript>
```

HTML File Paths

| Path | Description |
|--|--|
| <code> picture.jpg</code> | is located in the same folder as the current page |
| <code> picture.jpg</code> | is located in the images folder in the current folder |
| <code> picture.jpg</code> | is located in the images folder at the root of the current web |
| <code> picture.jpg</code> | is located in the folder one level up from the current folder |

HTML File Paths - A file path describes the location of a file in a web site's folder structure.

File paths are used when linking to external files like:

- Web pages
- Images
- Style sheets
- JavaScripts

Absolute File Paths - An absolute file path is the full URL to an internet file:

Example

```

```

The `` tag and the `src` and `alt` attributes are explained in the chapter about HTML Images.

Relative File Paths - A relative file path points to a file relative to the current page.

In this example the file path points to a file in the images folder located at the root of the current web:

Example

```

```

In this example the file path points to a file in the images folder located in the current folder:

Example

```

```

In this example the file path points to a file in the images folder located in the folder one level above the current folder:

Example

```

```

Best Practice

It is a best practice to use relative file paths (if possible).

When using relative file paths, your web pages will not be bound to your current base URL. All links will work on your own computer (localhost) as well as on your current public domain and your future public domains.

HTML Head

The HTML <head> Element - The <head> element is a container for metadata (data about data) and is placed between the <html> tag and the <body> tag.

HTML metadata is data about the HTML document. Metadata is not displayed.

Metadata typically define the document title, character set, styles, links, scripts, and other meta information.

The following tags describe metadata: <title>, <style>, <meta>, <link>, <script>, and <base>.

The HTML <title> Element - The <title> element defines the title of the document, and is required in all HTML/XHTML documents.

The <title> element:

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

A simple HTML document:

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Introduction Networking</title>
</head>
<body>
  Networking is.....
</body>
</html>
```

The HTML <style> Element - The <style> element is used to define style information for a single HTML page:

Example

```
<style>
  body {background-color: powderblue;}
  h1 {color: red;}
  p {color: blue;}
</style>
```

The HTML <link> Element - The <link> element is used to link to external style sheets

Example

```
<link rel="stylesheet" href="mystyle.css">
```

The HTML <meta> Element

The <meta> element is used to specify which character set is used, page description, keywords, author, and other metadata.

Metadata is used by browsers (how to display content), by search engines (keywords), and other web services.

Define the character set used:

```
<meta charset="UTF-8">
```

Define a description of your web page:

```
<meta name="description" content="Free Web tutorials">
```

Define keywords for search engines:


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

Define the author of a page:

```
<meta name="author" content="John Doe">
```

Refresh document every 30 seconds:

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

Example of <meta> tags:

Example

```
<meta charset="UTF-8">
<meta name="description" content="Computing tutorial 4">
<meta name="keywords" content="HTML,CSS,XML,JavaScript">
<meta name="author" content="Paik Poh Leong">
```

The HTML <script> Element

The <script> element is used to define client-side JavaScripts.

This JavaScript writes "Hello JavaScript!" into an HTML element with id="ctgp":

Example**<script>**

```
function myFunction {
    document.getElementById("ctgp").innerHTML = "Hello Computing classes!";
}
```

</script>

The HTML <base> Element

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

Example

```
<base href=" http://jpjc.moe.edu.sg/images/" target="_blank">
```

Reserved characters in HTML must be replaced with character entities.
 Characters that are not present on your keyboard can also be replaced by entities.

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: **<** or **<**;

Advantage of using an entity name: An entity name is easy to remember.

Disadvantage of using an entity name: 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: ** **;

A non-breaking space is a space that will not break into a new line.

Two words separated by a non-breaking space will stick together (not break into a new line). This is handy when breaking the words might be disruptive.

Examples:

- § 10
- 10 km/h
- 10 PM

Another common use of the non-breaking space is to prevent that browsers 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 ** ** character entity.

The non-breaking hyphen ([‑](#)) lets you use a hyphen character (-) that won't break.

Some Other Useful HTML Character Entities

| Result | Description | Entity Name | Entity Number |
|--------|------------------------------------|--------------------------|--------------------------|
| | non-breaking space | <code>&nbsp;</code> | <code>&#160;</code> |
| < | less than | <code>&lt;</code> | <code>&#60;</code> |
| > | greater than | <code>&gt;</code> | <code>&#62;</code> |
| & | ampersand | <code>&amp;</code> | <code>&#38;</code> |
| " | double quotation mark | <code>&quot;</code> | <code>&#34;</code> |
| ' | single quotation mark (apostrophe) | <code>&apos;</code> | <code>&#39;</code> |
| ¢ | cent | <code>&cent;</code> | <code>&#162;</code> |
| £ | pound | <code>&pound;</code> | <code>&#163;</code> |
| ¥ | yen | <code>&yen;</code> | <code>&#165;</code> |
| € | euro | <code>&euro;</code> | <code>&#8364;</code> |
| © | copyright | <code>&copy;</code> | <code>&#169;</code> |
| ® | registered trademark | <code>&reg;</code> | <code>&#174;</code> |

Note: Entity names are case sensitive.

🔄HTML Encoding (Character Sets)

To display an HTML page correctly, a web browser must know which character set (character encoding) to use.

What is Character Encoding?

ASCII was the first character encoding standard (also called character set). ASCII defined 128 different alphanumeric characters that could be used on the internet: numbers (0-9), English letters (A-Z), and some special characters like ! \$ + - () @ < > .

Because ANSI was so limited, HTML 4 also supported UTF-8.

UTF-8 (Unicode) covers almost all of the characters and symbols in the world.

The default character encoding for HTML5 is 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">`

The ASCII Character Set

ASCII uses the values from 0 to 31 (and 127) for control characters.

ASCII uses the values from 32 to 126 for letters, digits, and symbols.

ASCII does not use the values from 128 to 255.

The UTF-8 Character Set

UTF-8 is identical to ASCII for the values from 0 to 127.

UTF-8 does not use the values from 128 to 159.

UTF-8 is identical to both ANSI and 8859-1 for the values from 160 to 255.

UTF-8 continues from the value 256 with more than 10 000 different characters.

HTML Form Example

Name:

Address:

The `<form>` Element - The HTML `<form>` element defines a form that is used to collect user input:
`<form>`

form elements

`</form>`

An HTML form contains form elements.

Form elements are different types of input elements, like text fields, checkboxes, radio buttons, submit buttons, and more.

The `<input>` Element

The `<input>` element is the most important form element.

The `<input>` element can be displayed in several ways, depending on the type attribute.

Here are some examples:

| Type | Description |
|--|--|
| <code><input type="text"></code> | Defines a one-line text input field |
| <code><input type="radio"></code> | Defines a radio button (for selecting one of many choices) |
| <code><input type="submit"></code> | Defines a submit button (for submitting the form) |

Text Input - `<input type="text">` defines a one-line input field for text input:

Example

`<form>`

Name:
`
`

`<input type="text" name="name">`
`
`

Address:
`
`

`<input type="text" name="address">`

`</form>`

This is how it will look like in a browser:

Name:

Address:

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

The Name Attribute - Each input field must have a name attribute to be submitted.

If the name attribute is omitted, the data of that input field will not be sent at all.

This example will only submit the " address" input field:

Example

```
<form action="/action_page.asp">
  Name:<br>
  <input type="text" value="Paik Poh Leong"><br>
  Address:<br>
  <input type="text" name="address" value="144 Bukit Timak"><br><br>
  <input type="submit" value="Submit">
</form>
```

Grouping Form Data with <fieldset>

The <**fieldset**> element is used to group related data in a form.

The <**legend**> element defines a caption for the <**fieldset**> element.

Example

```
<form action="/action_page.php">
  <fieldset>
    <legend>Personal information:</legend>
      Name:<br>
      <input type="text" name="name" value="Paik Poh Leong"><br>
      Address:<br>
      <input type="text" name="address" value="144 Bukit Timak"><br><br>
      <input type="submit" value="Submit">
  </fieldset>
</form>
```

Radio Button Input - <input type="radio"> defines a radio button.

Radio buttons let a user select ONE of a limited number of choices:

Example

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

This is how the HTML code above will be displayed in a browser:

☐ Male
☐ Female

The Submit Button - <input type="submit"> defines a button for submitting the form data to a form-handler.

The form-handler is typically a **server page** with a script for processing input data.

The form-handler is specified in the form's action attribute:

Example

```
<form action="/action_page.asp">
  Name:<br>
  <input type="text" name="name" value="Paik Poh Leong"><br>
  Address:<br> <input type="text" name="address" value="144 Bukit Timak"><br><br>
  <input type="submit" value="Submit">
</form>
```

The Action Attribute - The action attribute defines the action to be performed when the form is submitted.

Normally, the form data is sent to a web page on the server when the user clicks on the submit button. In the example above, the form data is **sent to a page on the server** called "/action_page.asp". This page contains a **server-side script** that handles the form data:

```
<form action="/action_page.asp">
```

If the action attribute is omitted, the action is set to the current page.

The Target Attribute

The target attribute specifies if the submitted result will open in a new browser tab, a frame, or in the current window.

The default value is "_self" which means the form will be submitted in the current window.

To make the form result open in a new browser tab, use the value "_blank":

Example

```
<form action="/action_page.asp" target="_blank">
```

Other legal values are "_parent", "_top", or a name representing the name of an iframe.

The Method Attribute

The method attribute specifies the HTTP method (GET or POST) to be used when submitting the form data:

Example

```
<form action="/action_page.asp" method="get">
```

or:

Example

```
<form action="/action_page.asp" method="post">
```

When to Use GET?

The default method when submitting form data is GET.

However, when GET is used, the submitted form data will be visible in the page address field:

/action_page.asp?name=Paik%20Poh%20Leong&address=144%20Bukit%20Timak

Notes on GET:

- Appends form-data into the URL in name/value pairs
- The length of a URL is limited (about 3000 characters)
- Never use GET to send sensitive data! (will be visible in the URL)
- Useful for form submissions where a user wants to bookmark the result
- GET is better for non-secure data, like query strings in Google

When to Use POST?

Always use POST if the form data contains sensitive or personal information. The POST method does not display the submitted form data in the page address field.

Notes on POST:

- POST has no size limitations, and can be used to send large amounts of data.
- Form submissions with POST cannot be bookmarked

HTTP Methods: GET vs. POST

The two most used HTTP methods are: GET and POST.

What is HTTP?

The Hypertext Transfer **Protocol** (HTTP) is designed to **enable communications** between **clients** and **servers**. HTTP works as a **request-response protocol** between a client and server.

A web browser may be the client, and an application on a computer that hosts a web site may be the server.

Example: A client (browser) submits an HTTP request to the server; then the server returns a response to the client. The response contains status information about the request and may also contain the requested content.

Two HTTP Request Methods: GET and POST

Two commonly used methods for a request-response between a client and server are: GET and POST.

- GET - Requests data from a specified resource
- POST - Submits data to be processed to a specified resource

The GET Method

Note that the query string (**name/value pairs**) is sent in the URL of a GET request:
/test/demo_form.asp?name1=value1&name2=value2

Some other notes on GET requests:

- GET requests can be cached
- GET requests remain in the browser history
- GET requests can be bookmarked
- GET requests should never be used when dealing with sensitive data
- GET requests have length restrictions
- GET requests should be used only to retrieve data

The POST Method

Note that the query string (name/value pairs) is sent in the HTTP message body of a POST request:

POST /test/demo_form.asp HTTP/1.1

Host: jpjc.moe.edu.sg

name1=value1&name2=value2

Some other notes on POST requests:

- POST requests are never cached
- POST requests do not remain in the browser history
- POST requests cannot be bookmarked
- POST requests have no restrictions on data length

Compare GET vs. POST

The following table compares the two HTTP methods: GET and POST.

| GET | | POST |
|-----------------------------|--|--|
| BACK button/Reload | Harmless | Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted) |
| Bookmarked | Can be bookmarked | Cannot be bookmarked |
| Cached | Can be cached | Not cached |
| Encoding type | application/x-www-form-urlencoded | application/x-www-form-urlencoded or multipart/form-data. Use multipart encoding for binary data |
| History | Parameters remain in browser history | Parameters are not saved in browser history |
| Restrictions on data length | Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL length is 2048 characters) | No restrictions |
| Restrictions on data type | Only ASCII characters allowed | No restrictions. Binary data is also allowed |
| Security | GET is less secure compared to POST because data sent is part of the URL Never use GET when sending passwords or other sensitive information! | POST is a little safer than GET because the parameters are not stored in browser history or in web server logs |
| Visibility | Data is visible to everyone in the URL | Data is not displayed in the URL |

Other HTTP Request Methods

The following table lists some other HTTP request methods:

| Method | Description |
|---------|--|
| HEAD | Same as GET but returns only HTTP headers and no document body |
| PUT | Uploads a representation of the specified URI |
| DELETE | Deletes the specified resource |
| OPTIONS | Returns the HTTP methods that the server supports |
| CONNECT | Converts the request connection to a transparent TCP/IP tunnel |

HTML URL Encoding Reference

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

URL - Uniform Resource Locator

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

The URL is the address of a web page, like: <http://www.jpjc.moe.edu.sg>.

URL Encoding (% Percent 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 replaces unsafe ASCII characters with a "%" followed by two hexadecimal digits.

URLs cannot contain spaces. URL encoding normally replaces a space with a plus (+) sign or with %20.

URL Encoding Functions

In JavaScript, PHP, and ASP there are functions that can be used to URL encode a string.

In JavaScript you can use the **encodeURIComponent()** function.

```
<!--
URL encoding test.html
-->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
    <title>untitled</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script>
    <!--
    function urlencode()
    {
        var txtReceived=document.getElementById("txtCaptured").value;
        //var encodetxt=encodeURIComponent(txtReceived);
        var encodetxt=encodeURIComponent(txtReceived);
        document.getElementById("encodingresult").innerHTML=encodetxt;
    }
    -->
    </script>
</head>
<body>
<input type="text" name=" txtCaptured" id="txtCaptured" value="Welcome back JC2" size="30" />
<input type="button" value="URL Encode" onclick="urlencode()" />
<div id="encodingresult"></div>
</body>
</html>
```

Note: The JavaScript function encodes space as %20.

HTTP Status Messages

When a browser requests a service from a web server, an error might occur. This is a list of HTTP status messages that might be returned:

1xx: Information

| Message: | Description: |
|-------------------------|---|
| 100 Continue | The server has received the request headers, and the client should proceed to send the request body |
| 101 Switching Protocols | The requester has asked the server to switch protocols |
| 103 Checkpoint | Used in the resumable requests proposal to resume aborted PUT or POST requests |

2xx: Successful

| Message: | Description: |
|-----------------------------------|--|
| 200 OK | The request is OK (this is the standard response for successful HTTP requests) |
| 201 Created | The request has been fulfilled, and a new resource is created |
| 202 Accepted | The request has been accepted for processing, but the processing has not been completed |
| 203 Non-Authoritative Information | The request has been successfully processed, but is returning information that may be from another source |
| 204 No Content | The request has been successfully processed, but is not returning any content |
| 205 Reset Content | The request has been successfully processed, but is not returning any content, and requires that the requester reset the document view |
| 206 Partial Content | The server is delivering only part of the resource due to a range header sent by the client |

3xx: Redirection

| Message: | Description: |
|------------------------|---|
| 300 Multiple Choices | A link list. The user can select a link and go to that location. Maximum five addresses |
| 301 Moved Permanently | The requested page has moved to a new URL |
| 302 Found | The requested page has moved temporarily to a new URL |
| 303 See Other | The requested page can be found under a different URL |
| 304 Not Modified | Indicates the requested page has not been modified since last requested |
| 306 Switch Proxy | <i>No longer used</i> |
| 307 Temporary Redirect | The requested page has moved temporarily to a new URL |
| 308 Resume Incomplete | Used in the resumable requests proposal to resume aborted PUT or POST requests |

4xx: Client Error

| Message: | Description: |
|-----------------------------------|---|
| 400 Bad Request | The request cannot be fulfilled due to bad syntax |
| 401 Unauthorized | The request was a legal request, but the server is refusing to respond to it. For use when authentication is possible but has failed or not yet been provided |
| 402 Payment Required | <i>Reserved for future use</i> |
| 403 Forbidden | The request was a legal request, but the server is refusing to respond to it |
| 404 Not Found | The requested page could not be found but may be available again in the future |
| 405 Method Not Allowed | A request was made of a page using a request method not supported by that page |
| 406 Not Acceptable | The server can only generate a response that is not accepted by the client |
| 407 Proxy Authentication Required | The client must first authenticate itself with the proxy |
| 408 Request Timeout | The server timed out waiting for the request |
| 409 Conflict | The request could not be completed because of a conflict in the request |
| 410 Gone | The requested page is no longer available |
| 411 Length Required | The "Content-Length" is not defined. The server will not accept the request without it |
| 412 Precondition Failed | The precondition given in the request evaluated to false by the server |
| 413 Request Entity Too Large | The server will not accept the request, because the request entity is too large |

| | |
|-------------------------------------|--|
| 414 Request-URI Too Long | The server will not accept the request, because the URL is too long. Occurs when you convert a POST request to a GET request with a long query information |
| 415 Unsupported Media Type | The server will not accept the request, because the media type is not supported |
| 416 Requested Range Not Satisfiable | The client has asked for a portion of the file, but the server cannot supply that portion |
| 417 Expectation Failed | The server cannot meet the requirements of the Expect request-header field |

5xx: Server Error

| Message: | Description: |
|-------------------------------------|--|
| 500 Internal Server Error | A generic error message, given when no more specific message is suitable |
| 501 Not Implemented | The server either does not recognize the request method, or it lacks the ability to fulfill the request |
| 502 Bad Gateway | The server was acting as a gateway or proxy and received an invalid response from the upstream server |
| 503 Service Unavailable | The server is currently unavailable (overloaded or down) |
| 504 Gateway Timeout | The server was acting as a gateway or proxy and did not receive a timely response from the upstream server |
| 505 HTTP Version Not Supported | The server does not support the HTTP protocol version used in the request |
| 511 Network Authentication Required | The client needs to authenticate to gain network access |

HTML Form Elements

The <input> Element

The most important form element is the <input> element.

The **<input>** element can be displayed in several ways, depending on the type attribute.

Example

```
<input name="address" type="text">
```

If the type attribute is omitted, the input field gets the default type: "text".

The <select> Element

The **<select>** element defines a drop-down list:

Example

```
<select name="CivicTutorGroup">  
  <option value="17S09">17S09</option>  
  <option value="17S22">17S22</option>  
  <option value="17S23">17S23</option>  
</select>
```

The **<option>** elements defines an option that can be selected.

By default, the first item in the drop-down list is selected.

To define a pre-selected option, add the selected attribute to the option:

Example

```
<option value="17S09" selected>17S09</option>
```

Visible Values:

Use the **size** attribute to specify the number of visible values:

Example

```
<select name=" CivicTutorGroup " size="2">  
  <option value="17S09">17S09</option>  
  <option value="17S22">17S22</option>  
  <option value="17S23">17S23</option>  
</select>
```

Allow Multiple Selections:

Use the **multiple** attribute to allow the user to select more than one value:

Example

```
<select name="cars" size="3" multiple>  
  <option value="17S09">17S09</option>  
  <option value="17S22">17S22</option>  
  <option value="17S23">17S23</option>  
</select>
```

The <textarea> Element

The **<textarea>** element defines a multi-line input field (a text area):

Example

```
<textarea name="message" rows="10" cols="30">Congratulation you are now in JC2!  
</textarea>
```

The rows attribute specifies the visible number of lines in a text area.

The cols attribute specifies the visible width of a text area.

You can also define the size of the text area by using CSS:

Example

```
<textarea name="message" style="width:200px; height:600px">  
Congratulation you are now in JC2!  
</textarea>
```

The `<button>` Element - The `<button>` element defines a clickable button:

Example

```
<button type="button" onclick="alert('Hello Computing class!')">Click Me!</button>
```

Input Type Text

`<input type="text">` defines a one-line text input field:

Example

```
<form>  
  Name:<br>  
  <input type="text" name="Paik Poh Leong"><br>  
  Address:<br>  
  <input type="text" address="144 Bukit Timak ">  
</form>
```

Input Type Password

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

Example

```
<form>  
  User name:<br>  
  <input type="text" name="username"><br>  
  User password:<br>  
  <input type="password" name="psw">  
</form>
```

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

Input Type Submit

`<input type="submit">` defines a button for submitting form data to a form-handler.

The form-handler is typically a server page with a script for processing input data.

The form-handler is specified in the form's action attribute:

Example

```
<form action="/action_page.php">  
  Name:<br>  
  <input type="text" name="Paik Poh Leong"><br>  
  Address:<br>  
  <input type="text" address="144 Bukit Timak "><br>  
  <input type="submit" value="Submit">  
</form>
```

If you omit the submit button's value attribute, the button will get a default text:

Example

```
<form action="/action_page.php">  
  Name:<br>  
  <input type="text" name="Paik Poh Leong"><br>  
  Address:<br>  
  <input type="text" address="144 Bukit Timak "><br>  
  <input type="submit">  
</form>
```

Input Type Reset

`<input type="reset">` defines a reset button that will reset all form values to their default values:

Example

```
<form action="/action_page.php">
  Name:<br>
  <input type="text" name="Paik Poh Leong"><br>
  Address:<br>
  <input type="text" address="144 Bukit Timak "><br>
  <input type="submit" value="Submit">
  <input type="reset">
</form>
```

If you change the input values and then click the "Reset" button, the form-data will be reset to the default values.

Input Type Radio

`<input type="radio">` defines a radio button.

Radio buttons let a user select ONLY ONE of a limited number of choices:

Example

```
<form>
  <input type="radio" name="gender" value="male" checked> Male<br>
  <input type="radio" name="gender" value="female"> Female<br>
  <input type="radio" name="gender" value="other"> Other
</form>
```

Input Type Checkbox

`<input type="checkbox">` defines a checkbox.

Checkboxes let a user select ZERO or MORE options of a limited number of choices.

Example

```
<form>
  <input type="checkbox" name="H21" value="Physics"> Physics <br>
  <input type="checkbox" name="H22" value="Chemistry"> Chemistry
</form>
```

Input Type Button

`<input type="button">` defines a button:

Example

```
<input type="button" onclick="alert('Hello Computing classes!')" value="Click Me!">
```

The value Attribute - The value attribute specifies the initial value for an input field:

Example

```
<form action="">
  Name:<br><input type="text" name="name" value="Paik Poh Leong">
</form>
```

The readonly Attribute - The readonly attribute specifies that the input field is read only (cannot be changed):

Example

```
<form action="">
  Name:<br>
  <input type="text" name="name" value=" Paik Poh Leong " readonly>
</form>
```

The disabled Attribute - The disabled attribute specifies that the input field is disabled.

A disabled input field is unusable and un-clickable, and its value will not be sent when submitting the form:

Example

```
<form action="">  
Name:<br>  
<input type="text" name="name" value=" Paik Poh Leong " disabled>  
</form>
```

The size Attribute - The size attribute specifies the size (in characters) for the input field:

Example

```
<form action="">  
Name:<br>  
<input type="text" name="name" value=" Paik Poh Leong " size="40">  
</form>
```

The maxlength Attribute - The maxlength attribute specifies the maximum allowed length for the input field:

Example

```
<form action="">  
Name:<br>  
<input type="text" name="name" value=" Paik Poh Leong " maxlength="10">  
</form>
```

With a maxlength attribute, the input field will not accept more than the allowed number of characters. The maxlength attribute does not provide any feedback. If you want to alert the user, you must write JavaScript code.

Note: Input restrictions are not foolproof, and JavaScript provides many ways to add illegal input. To safely restrict input, it must be checked by the receiver (the server) as well!

The height and width Attributes

The **height** and **width** attributes specify the height and width of an <input type="image"> element.

Always specify the size of images. If the browser does not know the size, the page will flicker while images load.

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">
```

The multiple Attribute - The multiple attribute specifies that the user is allowed to enter more than one value in the <input> element.

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**>

HTML5 Input Types - HTML5 added several new input types:

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

New input types that are not supported by older web browsers, will behave as `<input type="text">`.

Input Type Color

The `<input type="color">` is used for input fields that should contain a color.

Depending on browser support, a color picker can show up in the input field. Opera Safari Chrome Firefox Internet Explorer

Example

```
<form>
  Select your favorite color: <input type="color" name="favcolor">
</form>
```

Input Type Date

The `<input type="date">` is used for input fields that should contain a date.

Example

```
<form>
  Birthday: <input type="date" name="bday">
</form>
```

You can also add restrictions to dates:

Example

```
<form>
  Enter a date before 1980-01-01:
  <input type="date" name="bday" max="1979-12-31"><br>
  Enter a date after 2000-01-01:
  <input type="date" name="bday" min="2000-01-02"><br>
</form>
```

Input Type Datetime-local

The `<input type="datetime-local">` specifies a date and time input field, with no time zone.

Example

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

Input Type Email

The `<input type="email">` is used for input fields that should contain an e-mail address.

Depending on browser support, the e-mail address can be automatically validated when submitted.

Some smartphones recognize the email type, and adds ".com" to the keyboard to match email input.

Example

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


Input Type Month

The `<input type="month">` allows the user to select a month and year.

Example

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

Input Type Number

The `<input type="number">` defines a numeric input field.

You can also set restrictions on what numbers are accepted.

The following example displays a numeric input field, where you can enter a value from 1 to 5:

Example

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

Input Restrictions

Here is a list of some common input restrictions (some are new in HTML5):

| Attribute | Description |
|-----------|---|
| disabled | Specifies that an input field should be disabled |
| max | Specifies the maximum value for an input field |
| maxlength | Specifies the maximum number of character for an input field |
| min | Specifies the minimum value for an input field |
| pattern | Specifies a regular expression to check the input value against |
| readonly | Specifies that an input field is read only (cannot be changed) |
| required | Specifies that an input field is required (must be filled out) |
| size | Specifies the width (in characters) of an input field |
| step | Specifies the legal number intervals for an input field |
| value | Specifies the default value for an input field |

The following example displays a numeric input field, where you can enter a value from 0 to 100, in steps of 10. The default value is 30:

Opera Safari Chrome Firefox Internet Explorer

Example

```
<form>
  Quantity: <input type="number" name="points" min="0" max="100" step="10" value="30">
</form>
```

Input Type Range

The `<input type="range">` defines a control for entering a number whose exact value is not important (like a slider control). Default range is 0 to 100. However, you can set restrictions on what numbers are accepted with the min, max, and step attributes:

Example

```
<form>
  <input type="range" name="points" min="0" max="10">
</form>
```

Input Type Search

The `<input type="search">` is used for search fields (a search field behaves like a regular text field).

Example

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

Input Type Time

The `<input type="time">` allows the user to select a time (no time zone).

Example

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

Input Type Url

The `<input type="url">` is used for input fields that should contain a URL address.

Some smartphones recognize the url type, and adds ".com" to the keyboard to match url input.

Example

```
<form>
  Add your homepage:
  <input type="url" name="homepage">
</form>
```

Input Type Week

The `<input type="week">` allows the user to select a week and year.

Example

```
<form>
  Select a week:
  <input type="week" name="week_year">
</form>
```