HTML5 Introduction

|  |  |
| --- | --- |
| **Year** | **Version** |
| 1989 | Tim Berners-Lee invented www |
| 1991 | Tim Berners-Lee invented HTML |
| 1993 | Dave Raggett drafted HTML+ |
| 1995 | HTML Working Group defined HTML 2.0 |
| 1997 | W3C Recommendation: HTML 3.2 |
| 1999 | W3C Recommendation: HTML 4.01 |
| 2000 | W3C Recommendation: XHTML 1.0 |
| 2008 | WHATWG HTML5 First Public Draft |
| 2012 | WHATWG HTML5 Living Standard |
| 2014 | W3C Recommendation: HTML5 |
| 2016 | W3C Candidate Recommendation: HTML 5.1 |
| 2017 | W3C Recommendation: HTML5.1 2nd Edition |
| 2017 | W3C Recommendation: HTML5.2 |

From 1991 to 1999, HTML developed from version 1 to version 4.

In year 2000, the World Wide Web Consortium (W3C) recommended XHTML 1.0. The XHTML syntax was strict, and the developers were forced to write valid and "well-formed" code.

In 2004, W3C's decided to close down the development of HTML, in favor of XHTML.

In 2004, WHATWG (Web Hypertext Application Technology Working Group) was formed. The WHATWG wanted to develop HTML, consistent with how the web was used, while being backward compatible with older versions of HTML.

In 2004 - 2006, the WHATWG gained support by the major browser vendors.

In 2006, W3C announced that they would support WHATWG.

In 2008, the first HTML5 public draft was released.

In 2012, WHATWG and W3C decided on a separation:

# HTML5 Browser Support

</section>  
  
</body>  
</html>

# HTML5 New Elements

|  |
| --- |
| Defines multiple media resources for media elements (<video> and <audio>) |
| [<track>](https://www.w3schools.com/tags/tag_track.asp) | Defines text tracks for media elements (<video> and <audio>) |
| [<video>](https://www.w3schools.com/tags/tag_video.asp) | Defines video or movie |

# HTML5 Semantic Elements

Semantics is the study of the meanings of words and phrases in a language.

Semantic elements = elements with a meaning.

## **What are Semantic Elements?**

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

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

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

## **Browser Support**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| Yes | Yes | Yes | Yes | Yes |

HTML5 semantic elements are supported in all modern browsers.

In addition, you can "teach" older browsers how to handle "unknown elements".

Read about it in [HTML5 Browser Support](https://www.w3schools.com/html/html5_browsers.asp).

## **New Semantic Elements in HTML5**

Many web sites contain HTML code like: <div id="nav"> <div class="header"> <div id="footer">  
to indicate navigation, header, and footer.

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

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



## **HTML5 <section> Element**

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

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

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

### **Example**

<section>  
  <h1>WWF</h1>  
  <p>The World Wide Fund for Nature (WWF) is....</p>  
</section>

[Try it Yourself »](https://www.w3schools.com/html/tryit.asp?filename=tryhtml5_section)

## **HTML5 <article> Element**

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

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

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

* Forum post
* Blog post
* Newspaper article

### **Example**

<article>  
  <h1>What Does WWF Do?</h1>  
  <p>WWF's mission is to stop the degradation of our planet's natural environment,  
  and build a future in which humans live in harmony with nature.</p>  
</article>

[Try it Yourself »](https://www.w3schools.com/html/tryit.asp?filename=tryhtml5_article)

## **Nesting <article> in <section> or Vice Versa?**

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

The <section> element defines section in a document.

Can we use the definitions to decide how to nest those elements? No, we cannot!

So, on the Internet, you will find HTML pages with <section> elements containing <article> elements, and <article> elements containing <section> elements.

You will also find pages with <section> elements containing <section> elements, and <article> elements containing <article> elements.

Example for a newspaper: The sport <article> in the sport **section**, may have a technical **section** in each <article>.

## **HTML5 <header> Element**

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

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

You can have several <header> elements in one document.

The following example defines a header for an article:

### **Example**

<article>  
  <header>  
    <h1>What Does WWF Do?</h1>  
    <p>WWF's mission:</p>  
  </header>  
  <p>WWF's mission is to stop the degradation of our planet's natural environment,  
  and build a future in which humans live in harmony with nature.</p>  
</article>

[Try it Yourself »](https://www.w3schools.com/html/tryit.asp?filename=tryhtml5_header)

## **HTML5 <footer> Element**

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

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

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

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

### **Example**

<footer>  
  <p>Posted by: Hege Refsnes</p>  
  <p>Contact information: <a href="mailto:someone@example.com">  
  someone@example.com</a>.</p>  
</footer>

[Try it Yourself »](https://www.w3schools.com/html/tryit.asp?filename=tryhtml5_footer)

## **HTML5 <nav> Element**

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

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

### **Example**

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

[Try it Yourself »](https://www.w3schools.com/html/tryit.asp?filename=tryhtml5_nav)

## **HTML5 <aside> Element**

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

The <aside> content should be related to the surrounding content.

### **Example**

<p>My family and I visited The Epcot center this summer.</p>  
  
<aside>  
  <h4>Epcot Center</h4>  
  <p>The Epcot Center is a theme park in Disney World, Florida.</p>  
</aside>

[Try it Yourself »](https://www.w3schools.com/html/tryit.asp?filename=tryhtml5_aside)

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

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

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

### **Example**

<figure>  
  <img src="pic\_trulli.jpg" alt="Trulli">  
  <figcaption>Fig1. - Trulli, Puglia, Italy.</figcaption>  
</figure>

[Try it Yourself »](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_figcaption)

The <img> element defines the image, the <figcaption> element defines the caption.

## **Why Semantic Elements?**

With HTML4, developers used their own id/class names to style elements: header, top, bottom, footer, menu, navigation, main, container, content, article, sidebar, topnav, etc.

This made it impossible for search engines to identify the correct web page content.

With the new HTML5 elements (<header> <footer> <nav> <section> <article>), this will become easier.

According to the W3C, a Semantic Web: "Allows data to be shared and reused across applications, enterprises, and communities."

## **Semantic Elements in HTML5**

Below is an alphabetical list of the new semantic elements in HTML5.

The links go to our complete [HTML5 Reference](https://www.w3schools.com/tags/default.asp).

|  |  |
| --- | --- |
| **Tag** | **Description** |
| [<article>](https://www.w3schools.com/tags/tag_article.asp) | Defines an article |
| [<aside>](https://www.w3schools.com/tags/tag_aside.asp) | Defines content aside from the page content |
| [<details>](https://www.w3schools.com/tags/tag_details.asp) | Defines additional details that the user can view or hide |
| [<figcaption>](https://www.w3schools.com/tags/tag_figcaption.asp) | Defines a caption for a <figure> element |
| [<figure>](https://www.w3schools.com/tags/tag_figure.asp) | Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc. |
| [<footer>](https://www.w3schools.com/tags/tag_footer.asp) | Defines a footer for a document or section |
| [<header>](https://www.w3schools.com/tags/tag_header.asp) | Specifies a header for a document or section |
| [<main>](https://www.w3schools.com/tags/tag_main.asp) | Specifies the main content of a document |
| [<mark>](https://www.w3schools.com/tags/tag_mark.asp) | Defines marked/highlighted text |
| [<nav>](https://www.w3schools.com/tags/tag_nav.asp) | Defines navigation links |
| [<section>](https://www.w3schools.com/tags/tag_section.asp) | Defines a section in a document |
| [<summary>](https://www.w3schools.com/tags/tag_summary.asp) | Defines a visible heading for a <details> element |
| [<time>](https://www.w3schools.com/tags/tag_time.asp) | Defines a date/time |

# HTML5 Migration

## **Migration from HTML4 to HTML5**

This chapter is entirely about how to **migrate** from **HTML4** to **HTML5**.

This chapter demonstrates how to convert an HTML4 page into an HTML5 page, without destroying anything of the original content or structure.

You can migrate from XHTML to HTML5, using the same recipe.

|  |  |
| --- | --- |
| **Typical HTML4** | **Typical HTML5** |
| <div id="header"> | <header> |
| <div id="menu"> | <nav> |
| <div id="content"> | <section> |
| <div class="article"> | <article> |
| <div id="footer"> | <footer> |

## **A Typical HTML4 Page**

### **Example**

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  
<html lang="en">  
<head>  
<meta http-equiv="Content-Type" content="text/html;charset=utf-8">  
<title>HTML4</title>  
<style>  
body {  
    font-family: Verdana,sans-serif;  
    font-size: 0.9em;  
}  
  
div#header, div#footer {  
    padding: 10px;  
    color: white;  
    background-color: black;  
}  
  
div#content {  
    margin: 5px;  
    padding: 10px;  
    background-color: lightgrey;  
}  
  
div.article {  
    margin: 5px;  
    padding: 10px;  
    background-color: white;  
}  
  
div#menu ul {  
    padding: 0;  
}  
  
div#menu ul li {  
    display: inline;  
    margin: 5px;  
}  
</style>  
</head>  
<body>  
  
<div id="header">  
  <h1>Monday Times</h1>  
</div>  
  
<div id="menu">  
  <ul>  
    <li>News</li>  
    <li>Sports</li>  
    <li>Weather</li>  
  </ul>  
</div>  
  
<div id="content">  
  <h2>News Section</h2>  
  <div class="article">  
    <h2>News Article</h2>  
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque in porta lorem. Morbi condimentum est nibh, et consectetur tortor feugiat at.</p>  
  </div>  
  <div class="article">  
    <h2>News Article</h2>  
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque in porta lorem. Morbi condimentum est nibh, et consectetur tortor feugiat at.</p>  
  </div>  
</div>  
  
<div id="footer">  
  <p>&amp;copy; 2016 Monday Times. All rights reserved.</p>  
</div>  
  
</body>  
</html>

## **Change to HTML5 Doctype**

Change the **doctype**:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

to the HTML5 doctype:

### **Example**

<!DOCTYPE html>

## **Change to HTML5 Encoding**

Change the **encoding** information:

<meta http-equiv="Content-Type" content="text/html;charset=utf-8">

to HTML5 encoding:

### **Example**

<meta charset="utf-8">

## **Add The HTML5Shiv**

The new HTML5 semantic elements are supported in all modern browsers. In addition, you can "teach" older browsers how to handle "unknown elements".

However, IE8 and earlier, does not allow styling of unknown elements. So, the HTML5Shiv is a JavaScript workaround to enable styling of HTML5 elements in versions of Internet Explorer prior to version 9.

Add the HTML5Shiv:

### **Example**

<!--[if lt IE 9]>  
  <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>  
<![endif]-->

[Try it Yourself »](https://www.w3schools.com/html/tryit.asp?filename=tryhtml5_migrate_4)

Read more about the**HTML5Shiv** in [HTML5 Browser Support](https://www.w3schools.com/html/html5_browsers.asp).

## **Change to HTML5 Semantic Elements**

The existing CSS contains id's and classes for styling the elements:

body {  
    font-family: Verdana,sans-serif;  
    font-size: 0.9em;  
}  
  
div#header, div#footer {  
    padding: 10px;  
    color: white;  
    background-color: black;  
}  
  
div#content {  
    margin: 5px;  
    padding: 10px;  
    background-color: lightgrey;  
}  
  
div.article {  
    margin: 5px;  
    padding: 10px;  
    background-color: white;  
}  
  
div#menu ul {  
    padding: 0;  
}  
  
div#menu ul li {  
    display: inline;  
    margin: 5px;  
}

Replace with equal CSS styles for HTML5 semantic elements:

body {  
    font-family: Verdana,sans-serif;  
    font-size: 0.9em;  
}  
  
header, footer {  
    padding: 10px;  
    color: white;  
    background-color: black;  
}  
  
section {  
    margin: 5px;  
    padding: 10px;  
    background-color: lightgrey;  
}  
  
article {  
    margin: 5px;  
    padding: 10px;  
    background-color: white;  
}  
  
nav ul {  
    padding: 0;  
}  
  
nav ul li {  
    display: inline;  
    margin: 5px;  
}

Finally, change the elements to HTML5 semantic elements:

### **Example**

<body>  
  
<header>  
<h1>Monday Times</h1>  
</header>  
  
<nav>  
<ul>  
<li>News</li>  
<li>Sports</li>  
<li>Weather</li>  
</ul>  
</nav>  
  
<section>  
<h2>News Section</h2>  
<article>  
<h2>News Article</h2>  
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque in porta lorem. Morbi condimentum est nibh, et consectetur tortor feugiat at.</p>  
</article>  
<article>  
<h2>News Article</h2>  
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque in porta lorem. Morbi condimentum est nibh, et consectetur tortor feugiat at.</p>  
</article>  
</section>  
  
<footer>  
<p>&copy; 2014 Monday Times. All rights reserved.</p>  
</footer>  
  
</body>

[Try it Yourself »](https://www.w3schools.com/html/tryit.asp?filename=tryhtml5_migrate_5)

## **The Difference Between <article> <section> and <div>**

There is a confusing (lack of) difference in the HTML5 standard, between <article> <section> and <div>.

In the HTML5 standard, the <section> element is defined as a block of related elements.

The <article> element is defined as a complete, self-contained block of related elements.

The <div> element is defined as a block of children elements.

How to interpret that?

In the example above, we have used <section> as a container for related <articles>.

But, we could have used <article> as a container for articles as well.

Here are some different examples:

### **<article> in <article>:**

<article>  
  
<h2>Famous Cities</h2>  
  
<article>  
<h2>London</h2>  
<p>London is the capital city of England. It is the most populous city in the United Kingdom,  
with a metropolitan area of over 13 million inhabitants.</p>  
</article>  
  
<article>  
<h2>Paris</h2>  
<p>Paris is the capital and most populous city of France.</p>  
</article>  
  
<article>  
<h2>Tokyo</h2>  
<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area,  
and the most populous metropolitan area in the world.</p>  
</article>  
  
</article>

### **<div> in <article>:**

<article>  
  
<h2>Famous Cities</h2>  
  
<div class="city">  
<h2>London</h2>  
<p>London is the capital city of England. It is the most populous city in the United Kingdom,  
with a metropolitan area of over 13 million inhabitants.</p>  
</div>  
  
<div class="city">  
<h2>Paris</h2>  
<p>Paris is the capital and most populous city of France.</p>  
</div>  
  
<div class="city">  
<h2>Tokyo</h2>  
<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area,  
and the most populous metropolitan area in the world.</p>  
</div>  
  
</article>

[Try it Yourself »](https://www.w3schools.com/html/tryit.asp?filename=tryhtml5_migrate_7)

### **<div> in <section> in <article>:**

<article>  
  
<section>  
<h2>Famous Cities</h2>  
  
<div class="city">  
<h2>London</h2>  
<p>London is the capital city of England. It is the most populous city in the United Kingdom,  
with a metropolitan area of over 13 million inhabitants.</p>  
</div>  
  
<div class="city">  
<h2>Paris</h2>  
<p>Paris is the capital and most populous city of France.</p>  
</div>  
  
<div class="city">  
<h2>Tokyo</h2>  
<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area,  
and the most populous metropolitan area in the world.</p>  
</div>  
</section>  
  
<section>  
<h2>Famous Countries</h2>  
  
<div class="country">  
<h2>England</h2>  
<p>London is the capital city of England. It is the most populous city in the United Kingdom,  
with a metropolitan area of over 13 million inhabitants.</p>  
</div>  
  
<div class="country">  
<h2>France</h2>  
<p>Paris is the capital and most populous city of France.</p>  
</div>  
  
<div class="country">  
<h2>Japan</h2>  
<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area,  
and the most populous metropolitan area in the world.</p>  
</div>  
</section>  
  
</article>

# HTML5 Style Guide and Coding Conventions

## **HTML Coding Conventions**

Web developers are often uncertain about the coding style and syntax to use in HTML.

Between 2000 and 2010, many web developers converted from HTML to XHTML.

With XHTML, developers were forced to write valid and "well-formed" code.

HTML5 is a bit more sloppy when it comes to code validation.

## **Be Smart and Future Proof**

A consistent use of style makes it easier for others to understand your HTML.

In the future, programs like XML readers may want to read your HTML.

Using a well-formed-"close to XHTML" syntax can be smart.

Always keep your code tidy, clean and well-formed.

## **Use Correct Document Type**

Always declare the document type as the first line in your document:

<!DOCTYPE html>

If you want consistency with lower case tags, you can use:

<!doctype html>

## **Use Lower Case Element Names**

HTML5 allows mixing uppercase and lowercase letters in element names.

We recommend using lowercase element names because:

* Mixing uppercase and lowercase names is bad
* Developers normally use lowercase names (as in XHTML)
* Lowercase look cleaner
* Lowercase are easier to write

### **Bad:**

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

### **Very Bad:**

<Section>   
  <p>This is a paragraph.</p>  
</SECTION>

### **Good:**

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

## **Close All HTML Elements**

In HTML5, you don't have to close all elements (for example the <p> element).

We recommend closing all HTML elements.

### **Bad:**

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

### **Good:**

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

## **Close Empty HTML Elements**

In HTML5, it is optional to close empty elements.

### **Allowed:**

<meta charset="utf-8">

### **Also Allowed:**

<meta charset="utf-8" />

However, the closing slash (/) is REQUIRED in XHTML and XML.

If you expect XML software to access your page, it is a good idea to keep the closing slash!

## **Use Lower Case Attribute Names**

HTML5 allows mixing uppercase and lowercase letters in attribute names.

We recommend using lowercase attribute names because:

* Mixing uppercase and lowercase names is bad
* Developers normally use lowercase names (as in XHTML)
* Lowercase look cleaner
* Lowercase are easier to write

### **Bad:**

<div CLASS="menu">

### **Good:**

<div class="menu">

## **Quote Attribute Values**

HTML5 allows attribute values without quotes.

We recommend quoting attribute values because:

* Mixing uppercase and lowercase values is bad
* Quoted values are easier to read
* You MUST use quotes if the value contains spaces

### **Very bad:**

This will not work, because the value contains spaces:

<table class=table striped>

### **Bad:**

<table class=striped>

### **Good:**

<table class="striped">

## **Image Attributes**

Always add the alt attribute to images. This attribute is important when the image for some reason cannot be displayed. Also, always define image width and height. It reduces flickering because the browser can reserve space for the image before loading.

### **Bad:**

<img src="html5.gif">

### **Good:**

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

## **Spaces and Equal Signs**

HTML5 allows spaces around equal signs. But space-less is easier to read and groups entities better together.

### **Bad:**

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

### **Good:**

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

## **Avoid Long Code Lines**

When using an HTML editor, it is inconvenient to scroll right and left to read the HTML code.

Try to avoid code lines longer than 80 characters.

## **Blank Lines and Indentation**

Do not add blank lines without a reason.

For readability, add blank lines to separate large or logical code blocks.

For readability, add two spaces of indentation. Do not use the tab key.

Do not use unnecessary blank lines and indentation. It is not necessary to indent every element:

### **Unnecessary:**

<body>  
  
  <h1>Famous Cities</h1>  
  
  <h2>Tokyo</h2>  
  
  <p>  
    Tokyo is the capital of Japan, the center of the Greater Tokyo Area,  
    and the most populous metropolitan area in the world.  
    It is the seat of the Japanese government and the Imperial Palace,  
    and the home of the Japanese Imperial Family.  
  </p>  
  
</body>

### **Better:**

<body>  
  
<h1>Famous Cities</h1>  
  
<h2>Tokyo</h2>  
<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area,  
and the most populous metropolitan area in the world.  
It is the seat of the Japanese government and the Imperial Palace,  
and the home of the Japanese Imperial Family.</p>  
  
</body>

### **Table Example:**

<table>  
  <tr>  
    <th>Name</th>  
    <th>Description</th>  
  </tr>  
  <tr>  
    <td>A</td>  
    <td>Description of A</td>  
  </tr>  
  <tr>  
    <td>B</td>  
    <td>Description of B</td>  
  </tr>  
</table>

### **List Example:**

<ol>  
  <li>London</li>  
  <li>Paris</li>  
  <li>Tokyo</li>  
</ol>

## **Omitting <html> and <body>?**

In HTML5, the <html> tag and the <body> tag can be omitted.

The following code will validate as HTML5:

### **Example**

<!DOCTYPE html>  
<head>  
  <title>Page Title</title>  
</head>  
  
<h1>This is a heading</h1>  
<p>This is a paragraph.</p>

[Try it Yourself »](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_syntax_nobody)

**However, we do not recommend omitting the <html> and the <body> tag.**

The <html> element is the document root. It is the recommended place for specifying the page language:

<!DOCTYPE html>  
<html lang="en-US">

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

Omitting <html> or <body> can crash DOM and XML software.

Omitting <body> can produce errors in older browsers (IE9).

## **Omitting <head>?**

In HTML5, the <head> tag can also be omitted.

By default, browsers will add all elements before <body> to a default <head> element.

You can reduce the complexity of HTML by omitting the <head> tag:

### **Example**

<!DOCTYPE html>  
<html>  
<title>Page Title</title>  
  
<body>  
  <h1>This is a heading</h1>  
  <p>This is a paragraph.</p>  
</body>  
  
</html>

[Try it Yourself »](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_syntax_nohead)

**However, we do not recommend omitting the <head> tag.**

Omitting tags is unfamiliar to web developers. It needs time to be established as a guideline.

## **Meta Data**

The <title> element is required in HTML5. Make the title as meaningful as possible:

<title>HTML5 Syntax and Coding Style</title>

To ensure proper interpretation and correct search engine indexing, both the language and the character encoding should be defined as early as possible in a document:

<!DOCTYPE html>  
<html lang="en-US">  
<head>  
  <meta charset="UTF-8">  
  <title>HTML5 Syntax and Coding Style</title>  
</head>

## **Setting The Viewport**

HTML5 introduced a method to let web designers take control over the viewport, through the <meta> tag.

The viewport is the user's visible area of a web page. It varies with the device, and will be smaller on a mobile phone than on a computer screen.

You should include the following <meta> viewport element in all your web pages:

<meta name="viewport" content="width=device-width, initial-scale=1.0">

A <meta> viewport element gives the browser instructions on how to control the page's dimensions and scaling.

The width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).

The initial-scale=1.0 part sets the initial zoom level when the page is first loaded by the browser.

Here is an example of a web page without the viewport meta tag, and the same web page with the viewport meta tag:

**Tip:** If you are browsing this page with a phone or a tablet, you can click on the two links below to see the difference.

[[](https://www.w3schools.com/html/example_withoutviewport.htm)  
  
**Without the viewport meta tag**](https://www.w3schools.com/html/example_withoutviewport.htm) 

[[](https://www.w3schools.com/html/example_withviewport.htm)  
  
**With the viewport meta tag**](https://www.w3schools.com/html/example_withviewport.htm) 

## **HTML Comments**

Short comments should be written on one line, like this:

<!-- This is a comment -->

Comments that spans more than one line, should be written like this:

<!--   
  This is a long comment example. This is a long comment example.  
  This is a long comment example. This is a long comment example.  
-->

Long comments are easier to observe if they are indented two spaces.

## **Style Sheets**

Use simple syntax for linking to style sheets (the type attribute is not necessary):

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

Short rules can be written compressed, like this:

p.intro {font-family: Verdana; font-size: 16em;}

Long rules should be written over multiple lines:

body {  
  background-color: lightgrey;  
  font-family: "Arial Black", Helvetica, sans-serif;  
  font-size: 16em;  
  color: black;  
}

* Place the opening bracket on the same line as the selector
* Use one space before the opening bracket
* Use two spaces of indentation
* Use semicolon after each property-value pair, including the last
* Only use quotes around values if the value contains spaces
* Place the closing bracket on a new line, without leading spaces
* Avoid lines over 80 characters

## **Loading JavaScript in HTML**

Use simple syntax for loading external scripts (the type attribute is not necessary):

<script src="myscript.js">

## **Accessing HTML Elements with JavaScript**

A consequence of using "untidy" HTML styles can result in JavaScript errors.

These two JavaScript statements will produce different results:

### **Example**

var obj = getElementById("Demo")  
  
var obj = getElementById("demo")

[Try it Yourself »](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_syntax_javascript)

[Visit the JavaScript Style Guide](https://www.w3schools.com/js/js_conventions.asp).

## **Use Lower Case File Names**

Some web servers (Apache, Unix) are case sensitive about file names: "london.jpg" cannot be accessed as "London.jpg".

Other web servers (Microsoft, IIS) are not case sensitive: "london.jpg" can be accessed as "London.jpg" or "london.jpg".

If you use a mix of upper and lower case, you have to be extremely consistent.

If you move from a case insensitive to a case sensitive server, even small errors will break your web!

To avoid these problems, always use lower case file names.

## **File Extensions**

HTML files should have a **.html** or **.htm** extension.

CSS files should have a **.css** extension.

JavaScript files should have a **.js** extension.

## **Differences Between .htm and .html**

There is no difference between the .htm and .html extensions. Both will be treated as HTML by any web browser or web server.

The differences are cultural:

.htm "smells" of early DOS systems where the system limited the extensions to 3 characters.

.html "smells" of Unix operating systems that did not have this limitation.

## **Technical Differences**

When a URL does not specify a filename (like https://www.w3schools.com/css/), the server returns a default filename. Common default filenames are index.html, index.htm, default.html and default.htm.

If your server is configured only with "index.html" as default filename, your file must be named "index.html", not "index.htm."

However, servers can be configured with more than one default filename, and normally you can set up as many default filenames as needed.

Anyway, the full extension for HTML files is .html, and there's no reason it should not be used.

# HTML5 Canvas

The HTML <canvas> element is used to draw graphics on a web page.

The graphic to the left is created with <canvas>. It shows four elements: a red rectangle, a gradient rectangle, a multicolor rectangle, and a multicolor text.

## **What is HTML Canvas?**

The HTML <canvas> element is used to draw graphics, on the fly, via JavaScript.

The <canvas> element is only a container for graphics. You must use JavaScript to actually draw the graphics.

Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

## **Browser Support**

The numbers in the table specify the first browser version that fully supports the <canvas> element.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Element |  |  |  |  |  |
| <canvas> | 4.0 | 9.0 | 2.0 | 3.1 | 9.0 |

## **Canvas Examples**

A canvas is a rectangular area on an HTML page. By default, a canvas has no border and no content.

The markup looks like this:

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

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

Here is an example of a basic, empty canvas:

### **Example**

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

[Try it Yourself »](https://www.w3schools.com/html/tryit.asp?filename=tryhtml5_canvas_empty)

### **Draw a Line**

### **Example**

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

[Try it Yourself »](https://www.w3schools.com/html/tryit.asp?filename=tryhtml5_canvas_tut_path)

### **Draw a Circle**

### **Example**

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

### **Draw a Text**

### **Example**

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

### **Stroke Text**

### **Example**

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

### **Draw Linear Gradient**

### **Example**

var c = document.getElementById("myCanvas");  
var ctx = c.getContext("2d");  
  
// Create gradient  
var grd = ctx.createLinearGradient(0, 0, 200, 0);  
grd.addColorStop(0, "red");  
grd.addColorStop(1, "white");  
  
// Fill with gradient  
ctx.fillStyle = grd;  
ctx.fillRect(10, 10, 150, 80);

### **Draw Circular Gradient**

### **Example**

var c = document.getElementById("myCanvas");  
var ctx = c.getContext("2d");  
  
// Create gradient  
var grd = ctx.createRadialGradient(75, 50, 5, 90, 60, 100);  
grd.addColorStop(0, "red");  
grd.addColorStop(1, "white");  
  
// Fill with gradient  
ctx.fillStyle = grd;  
ctx.fillRect(10, 10, 150, 80);

### **Draw Image**

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

[Try it Yourself »](https://www.w3schools.com/html/tryit.asp?filename=tryhtml5_canvas_tut_img)

# HTML5 SVG

## **What is SVG?**

* SVG stands for Scalable Vector Graphics
* SVG is used to define graphics for the Web
* SVG is a W3C recommendation

## **The HTML <svg> Element**

The HTML <svg> element is a container for SVG graphics.

SVG has several methods for drawing paths, boxes, circles, text, and graphic images.

## **Browser Support**

The numbers in the table specify the first browser version that fully supports the <svg> element.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Element |  |  |  |  |  |
| <svg> | 4.0 | 9.0 | 3.0 | 3.2 | 10.1 |

## **SVG Circle**

### **Example**

<!DOCTYPE html>  
<html>  
<body>  
  
<svg width="100" height="100">  
  <circle cx="50" cy="50" r="40" stroke="green" stroke-width="4" fill="yellow" />  
</svg>  
  
</body>  
</html>

[Try it Yourself »](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_svg_circle)

## **SVG Rectangle**

### **Example**

<svg width="400" height="100">  
  <rect width="400" height="100" style="fill:rgb(0,0,255);stroke-width:10;stroke:rgb(0,0,0)"/>  
</svg>

[Try it Yourself »](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_svg_rect)

## **SVG Rounded Rectangle**

### **Example**

<svg width="400" height="180">  
  <rect x="50" y="20" rx="20" ry="20" width="150" height="150"  
  style="fill:red;stroke:black;stroke-width:5;opacity:0.5" />  
</svg>

[Try it Yourself »](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_svg_rect_round)

## **SVG Star**

### **Example**

<svg width="300" height="200">  
  <polygon points="100,10 40,198 190,78 10,78 160,198"  
  style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;" />  
</svg>

[Try it Yourself »](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_svg_star)

## **SVG Logo**

SVG

### **Example**

<svg height="130" width="500">  
  <defs>  
    <linearGradient id="grad1" x1="0%" y1="0%" x2="100%" y2="0%">  
      <stop offset="0%" style="stop-color:rgb(255,255,0);stop-opacity:1" />  
      <stop offset="100%" style="stop-color:rgb(255,0,0);stop-opacity:1" />  
    </linearGradient>  
  </defs>  
  <ellipse cx="100" cy="70" rx="85" ry="55" fill="url(#grad1)" />  
  <text fill="#ffffff" font-size="45" font-family="Verdana" x="50" y="86">Super </text>  
  Sorry, your browser does not support inline SVG.  
</svg>

[Try it Yourself »](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_svg_logo)

## **Differences Between SVG and Canvas**

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

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

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

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

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

## **Comparison of Canvas and SVG**

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

|  |  |
| --- | --- |
| **Canvas** | **SVG** |
| * Resolution dependent * No support for event handlers * Poor text rendering capabilities * You can save the resulting image as .png or .jpg * Well suited for graphic-intensive games | * Resolution independent * Support for event handlers * Best suited for applications with large rendering areas (Google Maps) * Slow rendering if complex * (anything that uses the DOM alot will be slow) * Not suited for game applications |

# HTML Google Maps

## **A Basic Web Page**

To demonstrate how to add a Google Map to a web page, we will use a basic HTML page:

<!DOCTYPE html>

<html lang="en">

<head>

<title>Example of HTML image map</title>

</head>

<body>

<h1>Click on a shape to see how it works:</h1>

<img src="/examples/images/shapes.png" alt="Geometrical Shapes" usemap="#shapes">

<map name="shapes">

<area shape="circle" coords="40,40,40" href="/examples/html/circle.html" alt="Circle">

<area shape="poly" coords="148,2,104,80,193,80" href="/examples/html/triangle.html" alt="Triangle">

<area shape="rect" coords="226,2,323,80" href="/examples/html/rectangle.html" alt="Rectangle">

<area shape="poly" coords="392,2,352,32,366,80,418,80,432,32" href="/examples/html/pentagon.html" alt="Pentagon">

</map>

</body>

</html>

### **Example**

<!DOCTYPE html>  
<html>  
<body>  
  
<h1>My First Google Map</h1>  
  
<div id="map">My map will go here</div>  
  
</body>  
<html>

[Try it Yourself »](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_google_map_1)

## **Set the Map Size**

Set the size of the map:

### **Example**

<div id="map" style="width:400px;height:400px">

[Try it Yourself »](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_google_map_2)

## **Create a Function to Set The Map Properties**

This example defines a Google Map centered in London, England:

### **Example**

function myMap() {  
    var mapOptions = {  
        center: new google.maps.LatLng(51.5, -0.12),  
        zoom: 10,  
        mapTypeId: google.maps.MapTypeId.HYBRID  
    }  
var map = new google.maps.Map(document.getElementById("map"), mapOptions);  
}

[Try it Yourself »](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_google_map_3)

### **Example Explained**

The **mapOptions** variable defines the properties for the map.

The **center** property specifies where to center the map (using latitude and longitude coordinates).

The **zoom** property specifies the zoom level for the map (try to experiment with the zoom level).

The **mapTypeId** property specifies the map type to display. The following map types are supported: ROADMAP, SATELLITE, HYBRID, and TERRAIN.

The line: **var map=new google.maps.Map(document.getElementById("map"), mapOptions);**creates a new map inside the <div> element with id="map", using the parameters that are passed (mapOptions).

## **Add the Google Maps API**

Finally, show the map on the page!

The functionality of the map is provided by a JavaScript library located at Google. Add a script to refer to the Google Maps API with a callback to the myMap function:

### **Example**

<script src="https://maps.googleapis.com/maps/api/js?callback=myMap"></script>

# HTML Multimedia

Multimedia on the web is sound, music, videos, movies, and animations.

## **What is Multimedia?**

Multimedia comes in many different formats. It can be almost anything you can hear or see.

Examples: Images, music, sound, videos, records, films, animations, and more.

Web pages often contain multimedia elements of different types and formats.

In this chapter you will learn about the different multimedia formats.

## **Browser Support**

The first web browsers had support for text only, limited to a single font in a single color.

Later came browsers with support for colors and fonts, and images!

Audio, video, and animation have been handled differently by the major browsers. Different formats have been supported, and some formats require extra helper programs (plug-ins) to work.

Hopefully this will become history. HTML5 multimedia promises an easier future for multimedia.

## **Multimedia Formats**

Multimedia elements (like audio or video) are stored in media files.

The most common way to discover the type of a file, is to look at the file extension.

Multimedia files have formats and different extensions like: .swf, .wav, .mp3, .mp4, .mpg, .wmv, and .avi.

## **Common Video Formats**

|  |  |
| --- | --- |
| Videoformats | MP4 is the new and upcoming format for internet video.  MP4 is recommended by YouTube.  MP4 is supported by Flash Players.  MP4 is supported by HTML5. |

|  |  |  |
| --- | --- | --- |
| **Format** | **File** | **Description** |
| MPEG | .mpg .mpeg | MPEG. Developed by the Moving Pictures Expert Group. The first popular video format on the web. Used to be supported by all browsers, but it is not supported in HTML5 (See MP4). |
| AVI | .avi | AVI (Audio Video Interleave). Developed by Microsoft. Commonly used in video cameras and TV hardware. Plays well on Windows computers, but not in web browsers. |
| WMV | .wmv | WMV (Windows Media Video). Developed by Microsoft. Commonly used in video cameras and TV hardware. Plays well on Windows computers, but not in web browsers. |
| QuickTime | .mov | QuickTime. Developed by Apple. Commonly used in video cameras and TV hardware. Plays well on Apple computers, but not in web browsers. (See MP4) |
| RealVideo | .rm .ram | RealVideo. Developed by Real Media to allow video streaming with low bandwidths. It is still used for online video and Internet TV, but does not play in web browsers. |
| Flash | .swf .flv | Flash. Developed by Macromedia. Often requires an extra component (plug-in) to play in web browsers. |
| Ogg | .ogg | Theora Ogg. Developed by the Xiph.Org Foundation. Supported by HTML5. |
| WebM | .webm | WebM. Developed by the web giants, Mozilla, Opera, Adobe, and Google. Supported by HTML5. |
| MPEG-4 or MP4 | .mp4 | MP4. Developed by the Moving Pictures Expert Group. Based on QuickTime. Commonly used in newer video cameras and TV hardware. Supported by all HTML5 browsers. Recommended by YouTube. |

Only MP4, WebM, and Ogg video are supported by the HTML5 standard.

## **Audio Formats**

MP3 is the newest format for compressed recorded music. The term MP3 has become synonymous with digital music.

If your website is about recorded music, MP3 is the choice.

|  |  |  |
| --- | --- | --- |
| **Format** | **File** | **Description** |
| MIDI | .mid .midi | MIDI (Musical Instrument Digital Interface). Main format for all electronic music devices like synthesizers and PC sound cards. MIDI files do not contain sound, but digital notes that can be played by electronics. Plays well on all computers and music hardware, but not in web browsers. |
| RealAudio | .rm .ram | RealAudio. Developed by Real Media to allow streaming of audio with low bandwidths. Does not play in web browsers. |
| WMA | .wma | WMA (Windows Media Audio). Developed by Microsoft. Commonly used in music players. Plays well on Windows computers, but not in web browsers. |
| AAC | .aac | AAC (Advanced Audio Coding). Developed by Apple as the default format for iTunes. Plays well on Apple computers, but not in web browsers. |
| WAV | .wav | WAV. Developed by IBM and Microsoft. Plays well on Windows, Macintosh, and Linux operating systems. Supported by HTML5. |
| Ogg | .ogg | Ogg. Developed by the Xiph.Org Foundation. Supported by HTML5. |
| MP3 | .mp3 | MP3 files are actually the sound part of MPEG files. MP3 is the most popular format for music players. Combines good compression (small files) with high quality. Supported by all browsers. |
| MP4 | .mp4 | MP4 is a video format, but can also be used for audio. MP4 video is the upcoming video format on the internet. This leads to automatic support for MP4 audio by all browsers. |

Only MP3, WAV, and Ogg audio are supported by the HTML5 standard.

# HTML5 Video

## **Playing Videos in HTML**

Before HTML5, a video could only be played in a browser with a plug-in (like flash).

The HTML5 <video> element specifies a standard way to embed a video in a web page.

## **Browser Support**

The numbers in the table specify the first browser version that fully supports the <video> element.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Element |  |  |  |  |  |
| <video> | 4.0 | 9.0 | 3.5 | 4.0 | 10.5 |

## **The HTML <video> Element**

To show a video in HTML, use the <video> element:

### **Example**

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

[Try it Yourself »](https://www.w3schools.com/html/tryit.asp?filename=tryhtml5_video_all)

## **How it Works**

The controls attribute adds video controls, like play, pause, and volume.

It is a good idea to always include width and height attributes. If height and width are not set, the page might flicker while the video loads.

The <source> element allows you to specify alternative video files which the browser may choose from. The browser will use the first recognized format.

The text between the <video> and </video> tags will only be displayed in browsers that do not support the <video>element.

## **HTML <video> Autoplay**

To start a video automatically use the autoplay attribute:

### **Example**

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

[Try it Yourself »](https://www.w3schools.com/html/tryit.asp?filename=tryhtml5_video_autoplay)

The autoplay attribute does not work in mobile devices like iPad and iPhone.

## **HTML Video - Browser Support**

In HTML5, there are 3 supported video formats: MP4, WebM, and Ogg.

The browser support for the different formats is:

|  |  |  |  |
| --- | --- | --- | --- |
| **Browser** | **MP4** | **WebM** | **Ogg** |
| Internet Explorer | YES | NO | NO |
| Chrome | YES | YES | YES |
| Firefox | YES | YES | YES |
| Safari | YES | NO | NO |
| Opera | YES (from Opera 25) | YES | YES |

## **HTML Video - Media Types**

|  |  |
| --- | --- |
| **File Format** | **Media Type** |
| MP4 | video/mp4 |
| WebM | video/webm |
| Ogg | video/ogg |

## **HTML Video - Methods, Properties, and Events**

HTML5 defines DOM methods, properties, and events for the <video> element.

This allows you to load, play, and pause videos, as well as setting duration and volume.

There are also DOM events that can notify you when a video begins to play, is paused, etc.

### **Example: Using JavaScript**

Play/Pause Big Small Normal 

Video courtesy of [Big Buck Bunny](https://www.bigbuckbunny.org/).

[Try it Yourself »](https://www.w3schools.com/html/tryit.asp?filename=tryhtml5_video_js_prop)

For a full DOM reference, go to our [HTML5 Audio/Video DOM Reference](https://www.w3schools.com/tags/ref_av_dom.asp).

## **HTML5 Video Tags**

|  |  |
| --- | --- |
| **Tag** | **Description** |
| [<video>](https://www.w3schools.com/tags/tag_video.asp) | Defines a video or movie |
| [<source>](https://www.w3schools.com/tags/tag_source.asp) | Defines multiple media resources for media elements, such as <video> and <audio> |
| [<track>](https://www.w3schools.com/tags/tag_track.asp) | Defines text tracks in media players |

# HTML5 Audio

## **Audio on the Web**

Before HTML5, audio files could only be played in a browser with a plug-in (like flash).

The HTML5 <audio> element specifies a standard way to embed audio in a web page.

## **The HTML <audio> Element**

To play an audio file in HTML, use the <audio> element:

### **Example**

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

[Try it Yourself »](https://www.w3schools.com/html/tryit.asp?filename=tryhtml5_audio_all)

## **HTML Audio - How It Works**

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

The <source> element allows you to specify alternative audio files which the browser may choose from. The browser will use the first recognized format.

The text between the <audio> and </audio> tags will only be displayed in browsers that do not support the <audio>element.

## **HTML Audio - Browser Support**

In HTML5, there are 3 supported audio formats: MP3, WAV, and OGG.

The browser support for the different formats is:

|  |  |  |  |
| --- | --- | --- | --- |
| **Browser** | **MP3** | **WAV** | **OGG** |
| Internet Explorer | YES | NO | NO |
| Chrome | YES | YES | YES |
| Firefox | YES | YES | YES |
| Safari | YES | YES | NO |
| Opera | YES | YES | YES |

## **HTML Audio - Media Types**

|  |  |
| --- | --- |
| **File Format** | **Media Type** |
| MP3 | audio/mpeg |
| OGG | audio/ogg |
| WAV | audio/wav |

## **HTML Audio - Methods, Properties, and Events**

HTML5 defines DOM methods, properties, and events for the <audio> element.

This allows you to load, play, and pause audios, as well as set duration and volume.

There are also DOM events that can notify you when an audio begins to play, is paused, etc.

For a full DOM reference, go to our [HTML5 Audio/Video DOM Reference](https://www.w3schools.com/tags/ref_av_dom.asp).

## **HTML5 Audio Tags**

|  |  |
| --- | --- |
| **Tag** | **Description** |
| [<audio>](https://www.w3schools.com/tags/tag_audio.asp) | Defines sound content |
| [<source>](https://www.w3schools.com/tags/tag_source.asp) | Defines multiple media resources for media elements, such as <video> and <audio> |

# HTML Plug-ins

The purpose of a plug-in is to extend the functionality of a web browser.

## **HTML Helpers (Plug-ins)**

Helper applications (plug-ins) are computer programs that extend the standard functionality of a web browser.

Examples of well-known plug-ins are Java applets.

Plug-ins can be added to web pages with the <object> tag or the <embed> tag.

Plug-ins can be used for many purposes: display maps, scan for viruses, verify your bank id, etc.

To display video and audio: Use the <video> and <audio> tags.

## **The <object> Element**

The <object> element is supported by all browsers.

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

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

### **Example**

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

The <object> element can also be used to include HTML in HTML:

### **Example**

<object width="100%" height="500px" data="snippet.html"></object>

Or images if you like:

### **Example**

<object data="audi.jpeg"></object>

## **The <embed> Element**

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

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

Web browsers have supported the <embed> element for a long time. However, it has not been a part of the HTML specification before HTML5.

### **Example**

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

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

The <embed> element can also be used to include HTML in HTML:

### **Example**

<embed width="100%" height="500px" src="snippet.html">

Or images if you like:

### **Example**

<embed src="audi.jpeg">

# HTML <!DOCTYPE> Declaration

### **Example**

<!DOCTYPE html>  
<html>  
<head>  
<title>Title of the document</title>  
</head>  
  
<body>  
The content of the document......  
</body>  
  
</html>

[Try it Yourself »](https://www.w3schools.com/tags/tryit.asp?filename=tryhtml_doctype)

## **Definition and Usage**

The <!DOCTYPE> declaration must be the very first thing in your HTML document, before the <html> tag.

The <!DOCTYPE> declaration is not an HTML tag; it is an instruction to the web browser about what version of HTML the page is written in.

In HTML 4.01, the <!DOCTYPE> declaration refers to a DTD, because HTML 4.01 was based on SGML. The DTD specifies the rules for the markup language, so that the browsers render the content correctly.

HTML5 is not based on SGML, and therefore does not require a reference to a DTD.

**Tip:** Always add the <!DOCTYPE> declaration to your HTML documents, so that the browser knows what type of document to expect.

## **Differences Between HTML 4.01 and HTML5**

There are three different <!DOCTYPE> declarations in HTML 4.01. In HTML5 there is only one:

<!DOCTYPE html>

## **HTML Elements and Doctypes**

Look at our table of all [HTML elements, and what Doctype each element appears in](https://www.w3schools.com/tags/ref_html_dtd.asp).

## **Tips and Notes**

**Tip:** The <!DOCTYPE> declaration is NOT case sensitive.

**Tip:** To check if the HTML of your Web documents is valid, go to [W3C's validation service](http://validator.w3.org/).

## **Common DOCTYPE Declarations**

### **HTML 5**

<!DOCTYPE html>

### **HTML 4.01 Strict**

This DTD contains all HTML elements and attributes, but does NOT INCLUDE presentational or deprecated elements (like font). Framesets are not allowed.

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">

### **HTML 4.01 Transitional**

This DTD contains all HTML elements and attributes, INCLUDING presentational and deprecated elements (like font). Framesets are not allowed.

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

### **HTML 4.01 Frameset**

This DTD is equal to HTML 4.01 Transitional, but allows the use of frameset content.

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">

### **XHTML 1.0 Strict**

This DTD contains all HTML elements and attributes, but does NOT INCLUDE presentational or deprecated elements (like font). Framesets are not allowed. The markup must also be written as well-formed XML.

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

### **XHTML 1.0 Transitional**

This DTD contains all HTML elements and attributes, INCLUDING presentational and deprecated elements (like font). Framesets are not allowed. The markup must also be written as well-formed XML.

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

### **XHTML 1.0 Frameset**

This DTD is equal to XHTML 1.0 Transitional, but allows the use of frameset content.

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">

### **XHTML 1.1**

This DTD is equal to XHTML 1.0 Strict, but allows you to add modules (for example to provide Ruby support for East-Asian languages).

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

# ---------------------------

# HTML <a> Tag

### **Example**

A link to W3Schools.com:

<a href="https://www.w3schools.com">Visit W3Schools.com!</a>

[Try it Yourself »](https://www.w3schools.com/tags/tryit.asp?filename=tryhtml_link_test)

More "Try it Yourself" examples below.

## **Definition and Usage**

The <a> tag defines a hyperlink, which is used to link from one page to another.

The most important attribute of the <a> element is the href attribute, which indicates the link's destination.

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

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

## **Tips and Notes**

**Tip:** The following attributes: download, hreflang, media, rel, target, and type cannot be present if the href attribute is not present.

**Tip:** A linked page is normally displayed in the current browser window, unless you specify another target.

**Tip:** Use CSS to style links: [CSS Links Tutorial](https://www.w3schools.com/css/css_link.asp) and [CSS Buttons Tutorial](https://www.w3schools.com/css/css3_buttons.asp)

## **Differences Between HTML 4.01 and HTML5**

In HTML 4.01, the <a> tag could be either a hyperlink or an anchor. In HTML5, the <a> tag is always a hyperlink, but if it has no href attribute, it is only a placeholder for a hyperlink.

HTML5 has some new attributes, and some HTML 4.01 attributes are no longer supported.

## **Attributes**

= New in HTML5.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Value** | **Description** |
| [charset](https://www.w3schools.com/tags/att_a_charset.asp) | *char\_encoding* | Not supported in HTML5. Specifies the character-set of a linked document |
| [coords](https://www.w3schools.com/tags/att_a_coords.asp) | *coordinates* | Not supported in HTML5. Specifies the coordinates of a link |
| [download](https://www.w3schools.com/tags/att_a_download.asp) | *filename* | Specifies that the target will be downloaded when a user clicks on the hyperlink |
| [href](https://www.w3schools.com/tags/att_a_href.asp) | *URL* | Specifies the URL of the page the link goes to |
| [hreflang](https://www.w3schools.com/tags/att_a_hreflang.asp) | *language\_code* | Specifies the language of the linked document |
| [media](https://www.w3schools.com/tags/att_a_media.asp) | *media\_query* | Specifies what media/device the linked document is optimized for |
| [name](https://www.w3schools.com/tags/att_a_name.asp) | *section\_name* | Not supported in HTML5. Use the global [id attribute](https://www.w3schools.com/tags/ref_standardattributes.asp) instead. Specifies the name of an anchor |
| [ping](https://www.w3schools.com/tags/att_a_ping.asp) | list\_of\_URLs | Specifies a space-separated list of URLs to which, when the link is followed, post requests with the body ping will be sent by the browser (in the background). Typically used for tracking. |
| [rel](https://www.w3schools.com/tags/att_a_rel.asp) | alternate author bookmark external help license next nofollow noreferrer noopener prev search tag | Specifies the relationship between the current document and the linked document |
| [rev](https://www.w3schools.com/tags/att_a_rev.asp) | *text* | Not supported in HTML5. Specifies the relationship between the linked document and the current document |
| [shape](https://www.w3schools.com/tags/att_a_shape.asp) | default rect circle poly | Not supported in HTML5. Specifies the shape of a link |
| [target](https://www.w3schools.com/tags/att_a_target.asp) | \_blank \_parent \_self \_top framename | Specifies where to open the linked document |
| [type](https://www.w3schools.com/tags/att_a_type.asp) | *media\_type* | Specifies the media type of the linked document |

## **Default CSS Settings**

Most browsers will display the <a> element with the following default values:

a:link, a:visited {   
    color: (internal value);  
    text-decoration: underline;  
    cursor: auto;  
}  
  
a:link:active, a:visited:active {   
    color: (internal value);  
}

# HTML 5 <map> Tag

The HTML <map> tag is used for declaring an image map.

The <map> tag is used in conjunction with the [<area>](https://www.quackit.com/html_5/tags/html_area_tag.cfm) tag and [<img>](https://www.quackit.com/html_5/tags/html_img_tag.cfm) tag to specify clickable areas (sometimes referred to as "hot spots") on an image.

To create an image map, you use the <map> tag to declare the image map, and the [<area>](https://www.quackit.com/html_5/tags/html_area_tag.cfm) tag (nested within the <map> tag) to define the clickable areas. The [<img>](https://www.quackit.com/html_5/tags/html_img_tag.cfm) tag can be defined elsewhere on the page, and is linked to the <map>element using the name attribute.

<!DOCTYPE html>

<title>My Example</title>

<img src ="/pix/mueller\_hut/mueller\_hut\_t.jpg"

width="225" height="151" border="0"

alt="Mueller Hut, Mount Cook, and I"

usemap ="#muellermap" />

<map id ="muellermap"

name="muellermap">

<area shape ="rect" coords ="90,80,120,151"

href ="javascript:alert('Me');"

alt="Me" />

<area shape ="poly" coords ="55,55,120,80,90,80,90,100,70,100,20,80,55,55"

href ="http://en.wikipedia.org/wiki/Mount\_Cook" target="\_blank"

alt="Mount Cook" />

<area shape ="poly" coords ="145,80,145,100,215,90,215,80,180,60,145,80"

href ="http://www.natural-environment.com/places/mueller\_hut.php" target="\_blank"

alt="Mueller Hut" />

</map>