**1.** HTML 5 features which are useful *right now* include:

* Web Workers: Certain web applications use heavy scripts to perform functions. Web Workers use separate background threads for processing and it does not effect the performance of a web page.
* Video: You can embed video without third-party proprietary plug-ins or codec. Video becomes as easy as embedding an image.
* Canvas: This feature allows a web developer to render graphics on the fly. As with video, there is no need for a plug in.
* Application caches: Web pages will start storing more and more information locally on the visitor's computer. It works like cookies, but where cookies are small, the new feature allows for much larger files. Google Gears is an excellent example of this in action.
* Geolocation: Best known for use on mobile devices, geolocation is coming with HTML5

# **2.HTML <!DOCTYPE> Declaration**

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.

# **HTML Tag: head**

Where **metadata** — information about the document — is placed. It should be the first element inside an [html](http://htmldog.com/references/html/tags/html/) element.

A [title](http://htmldog.com/references/html/tags/title/) element is required within the head element. [meta](http://htmldog.com/references/html/tags/meta/), [style](http://htmldog.com/references/html/tags/style/), [base](http://htmldog.com/references/html/tags/base/), [link](http://htmldog.com/references/html/tags/link/), and [script](http://htmldog.com/references/html/tags/script/) can also be used.

head is required and it should be used just once. It should start immediately after the opening [html](http://htmldog.com/references/html/tags/html/) tag and end directly before the opening [body](http://htmldog.com/references/html/tags/body/) tag.

4. <html>

<head>

<title>HTML in 10 Simple Steps or Less</title>

<meta http-equiv="refresh" content="5" >

</head>

<body>

</body>

</html>

## 5. 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 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="I'm a tooltip">  
This is a paragraph.  
</p>

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

## The href Attribute

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

### **Example**

<a href="https://www.w3schools.com">This is a link</a>

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

You will learn more about links and the <a> tag later in this tutorial.

## Size Attributes

HTML images are defined with the **<img>** tag.

The filename of the source (**src**), and the size of the image (**width** and **height**) are all provided as **attributes**:

### **Example**

<img src="w3schools.jpg" width="104" height="142">

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

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

You will learn more about images and the <img> tag later in this tutorial.

## 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**

<img src="w3schools.jpg" alt="W3Schools.com" width="104" height="142">

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

## We Suggest: Use Lowercase Attributes

The HTML5 standard does not require lowercase attribute names.

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

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

At W3Schools we always use lowercase attribute names.

## We Suggest: Quote Attribute Values

The HTML5 standard does not require quotes around attribute values.

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

### **Example**

<a href=https://www.w3schools.com>

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

W3C **recommends** quotes in HTML, and **demands** 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=About W3Schools>

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

Using quotes are the most common. Omitting quotes can produce errors.   
At W3Schools we 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='John "ShotGun" Nelson'>

Or vice versa:

<p title="John 'ShotGun' Nelson">

## Chapter Summary

* All HTML elements can have **attributes**
* The **title** attribute provides additional "tool-tip" information
* The **href** attribute provides address information for links
* The **width** and **height** attributes provide size information for images
* The **alt** attribute provides text for screen readers
* At W3Schools we always use **lowercase** attribute names
* At W3Schools we always **quote** attribute values with double quotes

## Test Yourself with Exercises!

[Exercise 1 »](https://www.w3schools.com/html/exercise.asp?filename=exercise_attributes1) [Exercise 2 »](https://www.w3schools.com/html/exercise.asp?filename=exercise_attributes2) [Exercise 3 »](https://www.w3schools.com/html/exercise.asp?filename=exercise_attributes3) [Exercise 4 »](https://www.w3schools.com/html/exercise.asp?filename=exercise_attributes4) [Exercise 5 »](https://www.w3schools.com/html/exercise.asp?filename=exercise_attributes5)

## HTML Attributes

Below is an alphabetical list of some attributes often used in HTML:

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| alt | Specifies an alternative text for an image, when the image cannot be displayed |
| disabled | Specifies that an input element should be disabled |
| href | Specifies the URL (web address) for a link |
| id | Specifies a unique id for an element |
| src | Specifies the URL (web address) for an image |
| style | Specifies an inline CSS style for an element |
| title | Specifies extra information about an element (displayed as a tool tip) |

# 6. Empty element

An **empty element** is an [element](https://developer.mozilla.org/en-US/docs/Glossary/element) from HTML, SVG, or MathML that **cannot** have any child nodes (i.e., nested elements or text nodes).

The [HTML](https://html.spec.whatwg.org/multipage/), [SVG](https://www.w3.org/TR/SVG2/), and [MathML](https://www.w3.org/TR/MathML3/) specifications define very precisely what each element can contain. Many combinations have no semantic meaning, for example an [<audio>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/audio) element nested inside an [<hr>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/hr) element.

In HTML, using a closing tag on an empty element is usually invalid. For example, <input type="text"></input> is invalid HTML.

The empty elements in HTML are as follows:

* [<area>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/area)
* [<base>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/base)
* [<br>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/br)
* [<col>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/col)
* [<embed>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/embed)
* [<hr>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/hr)
* [<img>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/img)
* [<input>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input)
* [<keygen>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/keygen)(HTML 5.2 Draft removed)
* [<link>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/link)
* [<meta>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/meta)
* [<param>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/param)
* [<source>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/source)
* [<track>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/track)
* [<wbr>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/wbr)

## 7. Unordered HTML List

An unordered list starts with the **<ul>** tag. Each list item starts with the **<li>** tag.

The list items will be marked with bullets (small black circles) by default:

### **Example**

<ul>  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ul>

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

## 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>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ul>

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

### **Example - Circle**

<ul style="list-style-type:circle">  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ul>

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

### **Example - Square**

<ul style="list-style-type:square">  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ul>

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

### **Example - None**

<ul style="list-style-type:none">  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ul>

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

## Ordered HTML List

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

The list items will be marked with numbers by default:

### **Example**

<ol>  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ol>

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

## Ordered HTML List - The Type Attribute

The **type** attribute of the <ol> 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:**

<ol type="1">  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ol>

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

### **Uppercase Letters:**

<ol type="A">  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ol>

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

### **Lowercase Letters:**

<ol type="a">  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ol>

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

### **Uppercase Roman Numbers:**

<ol type="I">  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ol>

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

### **Lowercase Roman Numbers:**

<ol type="i">  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ol>

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

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

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

## Nested HTML Lists

List can be nested (lists inside lists):

### **Example**

<ul>  
  <li>Coffee</li>  
  <li>Tea  
    <ul>  
      <li>Black tea</li>  
      <li>Green tea</li>  
    </ul>  
  </li>  
  <li>Milk</li>  
</ul>

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

**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.

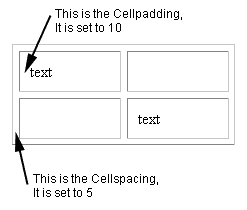
One popular way is to style a list horizontally, to create a menu:

### **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="#home">Home</a></li>  
  <li><a href="#news">News</a></li>  
  <li><a href="#contact">Contact</a></li>  
  <li><a href="#about">About</a></li>  
</ul>  
  
</body>  
</html>

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

8.



|  |  |
| --- | --- |
|  | **Cell padding**  is used for formatting purpose which is used to specify the space needed between the edges of the cells and also in the cell contents. The general format of specifying cell padding is as follows:  < table width="100" border="2" cellpadding="5">  The above adds 5 pixels of padding inside each cell . Cell Spacing: Cell spacing is one also used f formatting but there is a major difference between cell padding and cell spacing. It is as follows: Cell padding is used to set extra space which is used to separate cell walls from their contents. But in contrast cell spacing is used to set space between cells. |
|  | 9.  The required **alt** attribute specifies an alternate text for an **image**, if the **image**cannot be displayed. The **alt** attribute provides alternative information for an **image** if a 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)  10.  The short answer is: just add a target="\_blank" attribute to your links (anchor tags). Now when your visitors click that link, it will **open** in a **new** window or **tab**(depending on which web browser they are using and how they configured that browser). 11. BLOCK LEVEL ELEMENTS So what is a block-level element? A block-level element is an HTML element that begins a new line on a Web page and extends the full width of the available horizontal space of its parent element.   It creates large blocks of content like paragraphs or page divisions. In fact, most HTML elements are block-level elements.  Block-level elements are used within the HTML document's body. They can contain inline elements, as well as other block-level elements. INLINE ELEMENTS In contrast to a block-level element, an inline element can:   * Begin within a line * Does not start a new line * Its width only extends as far as it is defined by its tags.   ​An example of an inline element is the <strong> tag, which makes the font of the text content contained within boldface. An inline element generally only contains other inline elements, or it can contain nothing at all, such as the <br /> break tag.  There is also a third type of element in HTML: those that aren't displayed at all. These elements provide information about the page but aren't displayed when rendered in a Web browser.  For example:   * <style> defines styles and stylesheets. * <meta> defines meta data. * <head> is the HTML document element that holds these elements.   12. The <iframe> tag defines a rectangular region within the document in which the browser can **display** a separate document, including scrollbars and borders. The src attribute is used to specify the URL of the document that occupies the inline frame.  13. These are two different things. The anchor element is used to link to another page or to a certain part of the page if you use its ID.  <a href="index.html">Home</a>  And The link tag defines a link between a document and an external resource.  The link tag is used to link to external style sheets.  <head>  <link rel="stylesheet" type="text/css" href="theme.css">  </head>  FAQ 2) 1. ADVANTAGES OF HTML5 HTML5 has many new syntactical features. HTML5 comes with <audio>, <video> and <canvas> elements as well as integration of SVG content. With these new elements it will be very easy to integrate multimedia and graphical content to web without using flash and third party plugins.  HTML5 also have new elements like <section>, <article>, <header> and <nav> which enriches the semantic value of the document.  HTML5 advantages: The following are the advantages of HTML5.  **1)Cleaner markup/ Improved code:** HTML5 will enable web designers to use cleaner, neater code. We can remove div tags and replace them with semantic HTML5 elements.  **2)Elegant forms:** HTML5 enables designer to use fancier forms. There will be different type of inputs, search and different fields for different purpose.  **3)Consistency:** As websites will adopt the new HTML5 elements we will see more consistency in terms of HTML used to code a web page on one site compared to another. This will make it much easier for designers and developers to immediately understand how a web page is created.  **4)Supports rich media elements:** HTML5 has an inbuilt capability to play audio and video and so we can bid goodbye to those plugin tags.  **5)Offline Application Cache:** HTML5 offers an offline application cache facility which will load the page the user has visited even if the user is temporarily offline. This feature will help the files to load much faster and reduces load on server.  2.   |  |  | | --- | --- | | 174down voteaccepted | HTML5 has several goals which differentiate it from HTML4. Consistency in Handling Malformed Documents The primary one is consistent, defined error handling. As you know, HTML purposely supports 'tag soup', or the ability to write malformed code and have it corrected into a valid document. The problem is that the rules for doing this aren't written down anywhere. When a new browser vendor wants to enter the market, they just have to test malformed documents in various browsers (especially IE) and reverse-engineer their error handling. If they don't, then many pages won't display correctly (estimates place roughly 90% of pages on the net as being at least somewhat malformed).  So, HTML5 is attempting to discover and codify this error handling, so that browser developers can all standardize and greatly reduce the time and money required to display things consistently. As well, long in the future after HTML has died as a document format, historians may still want to read our documents, and having a completely defined parsing algorithm will greatly aid this. Better Web Application Features The secondary goal of HTML5 is to develop the ability of the browser to be an application platform, via HTML, CSS, and Javascript. Many elements have been added directly to the language that are currently (in HTML4) Flash or JS-based hacks, such as <canvas>, <video>, and <audio>. Useful things such as Local Storage (a js-accessible browser-built-in key-value database, for storing information beyond what cookies can hold), new input types such as date for which the browser can expose easy user interface (so that we don't have to use our js-based calendar date-pickers), and browser-supported form validation will make developing web applications much simpler for the developers, and make them much faster for the users (since many things will be supported natively, rather than hacked in via javascript). Improved Element Semantics There are many other smaller efforts taking place in HTML5, such as better-defined semantic roles for existing elements (<strong> and <em> now actually mean something different, and even <b>and <i> have vague semantics that should work well when parsing legacy documents) and adding new elements with useful semantics - <article>, <section>, <header>, <aside>, and <nav>should replace the majority of <div>s used on a web page, making your pages a bit more semantic, but more importantly, easier to read. No more painful scanning to see just what that random </div> is closing - instead you'll have an obvious </header>, or </article>, making the structure of your document much more intuitive. |  3. HTML Layout Elements Websites often display content in multiple columns (like a magazine or newspaper).  HTML5 offers new semantic elements that define the different parts of a web page:   |  |  | | --- | --- | |  | * <header> - Defines a header for a document or a section * <nav> - Defines a container for navigation links * <section> - Defines a section in a document * <article> - Defines an independent self-contained article * <aside> - Defines content aside from the content (like a sidebar) * <footer> - Defines a footer for a document or a section * <details> - Defines additional details * <summary> - Defines a heading for the <details> element |  HTML Layout Techniques There are four different ways to create multicolumn layouts. Each way has its pros and cons:   * HTML tables * CSS float property * CSS framework * CSS flexbox  Which One to Choose?**HTML Tables** The <table> element was not designed to be a layout tool! The purpose of the <table> element is to display tabular data. So, do not use tables for your page layout! They will bring a mess into your code. And imagine how hard it will be to redesign your site after a couple of months.  **Tip:** Do NOT use tables for your page layout! **CSS Frameworks** If you want to create your layout fast, you can use a framework, like [W3.CSS](https://www.w3schools.com/w3css/default.asp) or [Bootstrap](https://www.w3schools.com/bootstrap/default.asp). **CSS Floats** It is common to do entire web layouts using the CSS float property. Float is easy to learn - you just need to remember how the float and clear properties work. Disadvantages: Floating elements are tied to the document flow, which may harm the flexibility. Learn more about float in our [CSS Float and Clear](https://www.w3schools.com/css/css_float.asp) chapter. **CSS Flexbox** Flexbox is a new layout mode in CSS3.  Use of flexbox ensures that elements behave predictably when the page layout must accommodate different screen sizes and different display devices. Disadvantages: Does not work in IE10 and earlier. 4. The difference between <article> and <section> There’s been [a lot of confusion](http://adactio.com/journal/1654/) over the difference (or perceived lack of a difference) between the <article> and <section> elements in HTML5. The <article> element is a specialised kind of <section>; it has a more specific semantic meaning than <section> in that it is **an independent, self-contained** block of related content. We coulduse <section>, but using <article> gives more semantic meaning to the content.  By contrast <section> is only a block of related content, and <div> is only a block of content. Also as mentioned above the pubdateattribute doesn’t apply to <section>. To decide which of these three elements is appropriate, choose the first suitable option:   1. Would the content would make sense on its own in a feed reader? If so use <article> 2. Is the content related? If so use <section> 3. Finally if there’s no semantic relationship use <div>   [Dr Bruce](http://html5doctor.com/author/brucel) has written [HTML5 <article>s and <section>s, what’s the difference?](http://www.brucelawson.co.uk/2010/html5-articles-and-sections-whats-the-difference/), so we recommend reading that if you are still fuzzy on when to use <article>  5.refer details, summary  6. The <blockquote> tag specifies a section that is quoted from another source.  Browsers usually indent <blockquote> elements.  7.   |  |  | | --- | --- | | **Tag** | **Description** | | <audio> | Defines sound content | | <embed> | Defines a container for an external (non-HTML) application | | <source> | Defines multiple media resources for media elements (<video> and <audio>) | | <track> | Defines text tracks for media elements (<video> and <audio>) | | <video> | Defines video or movie | |

## 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.

## 8. 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 a lot will be slow) * Not suited for game applications |

|  |  |
| --- | --- |
| 1. | GET and POST are two different types of HTTP requests.  According to [Wikipedia](http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol):  **GET** requests a representation of the specified resource. Note that GET should not be used for operations that cause side-effects, such as using it for taking actions in web applications. One reason for this is that GET may be used arbitrarily by robots or crawlers, which should not need to consider the side effects that a request should cause.  and  **POST** submits data to be processed (e.g., from an HTML form) to the identified resource. The data is included in the body of the request. This may result in the creation of a new resource or the updates of existing resources or both.  So essentially GET is used to retrieve remote data, and POST is used to insert/update remote data.  HTTP/1.1 specification (RFC 2616) section 9 [**Method Definitions**](http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9) contains more information on GET and POST as well as the other HTTP methods, if you are interested.  In addition to explaining the intended uses of each method, the spec also provides at least one practical reason for why GET should only be used to retrieve data:  Authors of services which use the HTTP protocol SHOULD NOT use GET based forms for the submission of sensitive data, because this will cause this data to be encoded in the Request-URI. Many existing servers, proxies, and user agents will log the request URI in some place where it might be visible to third parties. Servers can use POST-based form submission instead  Finally, an important consideration when using GET for AJAX requests is that some browsers - IE in particular - will cache the results of a GET request. So if you, for example, poll using the same GET request you will always get back the same results, even if the data you are querying is being updated server-side. One way to alleviate this problem is to make the URL unique for each request by appending a timestamp. |

2. Two option.

1.

<input type="image" src="path/to/image">

2.

<input type="submit">

Then style with CSS

# 3. **HTML <input> pattern Attribute**

## Definition and Usage

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

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

**Tip:** Use the global [title](https://www.w3schools.com/tags/att_global_title.asp) attribute to describe the pattern to help the user.

**Tip:** Learn more about [regular expressions](https://www.w3schools.com/js/js_regexp.asp) in our JavaScript tutorial.

<form action="/action\_page.php">  
E-mail: <input type="email" name="email" pattern="[a-z0-9.\_%+-]+@[a-z0-9.-]+\.[a-z]{2,3}$">  
<input type="submit">  
</form>

|  |  |
| --- | --- |
|  |  |
|  | You can try this  <Label>ZIP Code</Label><input type="text" pattern="[0-9]{5}" title="Five digit zip code" /> |

# 4. **HTML <input> multiple Attribute**

<form action="/action\_page.php">  
  Select images: <input type="file" name="img" multiple>  
  <input type="submit">  
</form>

## Definition and Usage

The multiple attribute is a boolean attribute.

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

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

**Tip:** For <input type="file">: to select multiple files, hold down the CTRL or SHIFT key while selecting.

**Tip:** For <input type="email">: separate each email with a comma, like: mail@example.com, mail2@example.com, mail3@example.com in the email field.

### 5.

### url

There is currently some debate about whether the user has to enter http:// into a field using input="url". Browser vendors are discussing the possibility of pre-pending http:// to a URL following a check upon form submission. For up-to-date information, refer to the HTML5 specification.

The url input type, as you might expect, is for web addresses. You can use the multiple attribute to enter more than one URL. Like type="email", a browser will carry out simple validation on these fields and present an error message on form submission. This is likely to include looking for forward slashes, periods, and spaces, and possibly detecting a valid top-level domain (such as .com or .co.uk). Use the url input type like so:

<input type="url" name="url" required>

Again, we’ll take a look at how the iPhone renders type="url". As you can see in Figure 5, it has again updated the onscreen keyboard to ensure that completing the field is as simple as possible for the user by swapping the default space key for period, forward slash, and .com keys. (To access more endings like .org and .net, tap and hold the .com key.)

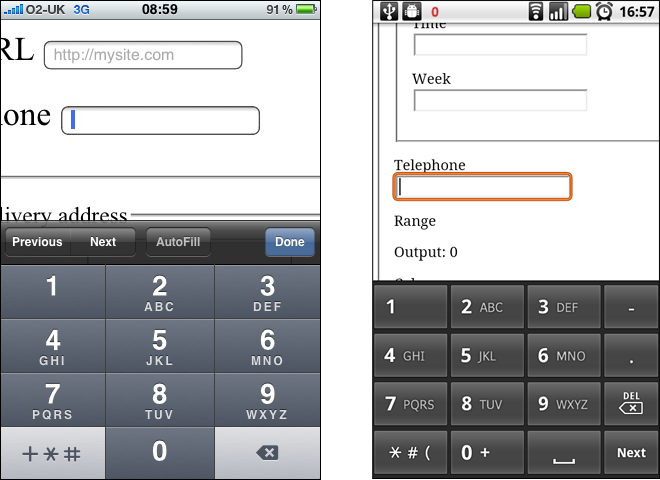
Figure 5. type=”url” activates a URL-specific keyboard on the iPhone.

### tel

tel differs from email and url in that no particular syntax is enforced. Phone numbers differ around the world, making it difficult to guarantee any type of specific notation except for allowing only numbers and perhaps a + symbol to be entered. It’s possible that you can validate specific phone numbers (if you can guarantee the format) using client-side validation. type="tel" is marked up as follows:

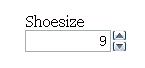
<input type="tel" name="tel" id="tel" required>

Once more, the iPhone recognises type="tel", only this time it goes one step further and completely changes the keyboard to the standard phone keyboard, as shown on the left in Figure 6. In addition to the iPhone, some Android devices (such as HTC Desire, shown on the right in Figure 6) also display a numeric keyboard for type="tel". That’s pretty handy, don’t you think? Nice, big keys for entering a phone number help you to get that big, nasty form completed quickly.

Figure 6. type="tel" on the iPhone and some Android devices dynamically changes the keyboard to a numeric keypad. (Android screenshot provided by Stuart Langridge).

### number

number, as you might expect, is used for specifying a numerical value. As with the majority of these new input types, Opera was the first to implement type="number". It, Safari, and Chrome render the input as a spinbox control (see Figure 7) whereby you can click the arrows to move up or down. Or if you prefer, you can type directly into the field. Firefox, on the other hand, renders the field like a standard text box. Support for type=”number” is in IE 10 also, although if you enter a non-numerical character the field empties when focus is lost and no feedback is provided to the user.

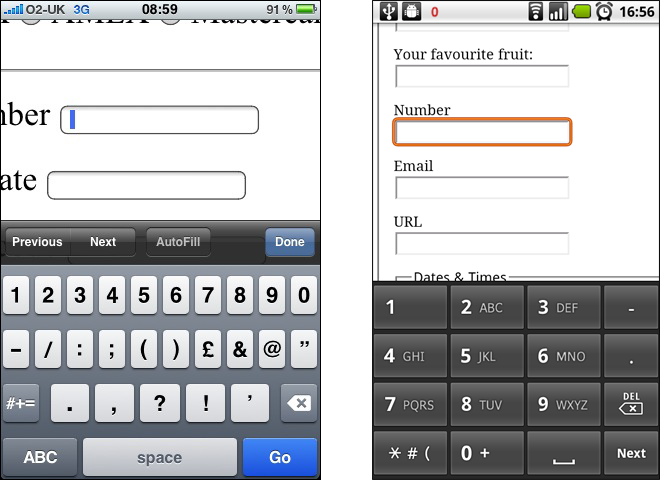
Figure 7. type="number" in Opera

With the additional attributes min, max, and step we can change the default step value of this spinbox control as well as set minimum, maximum, and starting values (using the standard HTML valueattribute). This example shows how these attributes work:

<input type="number" min="5" max="18" step="0.5"value="9" name="shoe-size">

In this example, min represents the minimum value the field will accept, and max represents the maximum value. If we reach the maximum or minimum value, the appropriate arrow on the spinbox control will be greyed out so you can no longer interact with it. step is the increment that the value should adjust up or down, with the default step value being 1. This means we can include negative values or step up in increments of 0.5 or 5. value is that attribute you’re used to from previous versions of HTML. Each of the attributes are optional, with defaults being set if they aren’t used.

In contrast to Opera’s implementation, the iPhone (Figure 8) and some Android devices (such as HTC Desire, shown on the right in Figure 6-13) simply render the field as a standard text box but optimize the keyboard for easy input.

Figure 8. type="number" on iPhone and Android HTC Desire (Android screenshot provided by Stuart Langridge)

To make the iPhone render with the standard telephone keypad as we saw for type="text" Chris Coyier, of CSS Tricks devised a little hoax you can use. Rather than using type=”number”, use a standard type="text" input and add a pattern attribute that accepts only numbers, as shown below. This solution isn’t ideal but if you think it could be useful, [Chris has put a short video together showing it in action](http://j.mp/numbertrick).

<input type="text" pattern="[0-9]\*" name="shoe-size">

Chris’ technique may soon become absolete though with the introduction of the inputmode attribute. The attribute, recently added to the specification will allow users to specify the type of input mechanism that is most useful for users. When implemented, you will be able to choose between numeric, latin, email, or kana input modes.

### novalidate and formnovalidate

The novalidate and formnovalidate attributes indicate that the form shouldn’t be validated when submitted. They are both Boolean attributes. formnovalidate can be applied to submit or image input types. The novalidate attribute can be set only on the form element.

An example use case for the formnovalidate attribute could be on a “save draft” button, where the form has fields that are required for submitting the draft but aren’t required for saving the draft. novalidate would be used in cases where you don’t want to validate the form but do want to take advantage of the more useful user interface enhancements that the new input types offer.

The following example shows how to use formnovalidate:

<form action="process.php">  
  <label for="email">Email:</label>  
  <input type="text" name="email"value="gordo@example.com">  
  <input type="submit" formnovalidate value="Submit">  
</form>

And this example shows how to use novalidate:

<form action="process.php" novalidate>  
  <label for="email">Email:</label>  
  <input type="text" name="email"value="gordo@example.com">  
  <input type="submit" value="Submit">  
</form>

### form

The form attribute is used to associate an input, select, or textarea element with a form (known as its form owner). Using form means that the element doesn’t need to be a child of the associated form and can be moved away from it in the source. The primary use case for this is that input buttons that are placed within tables can now be associated with a form.

<input type="button" name="sort-l-h" form="sort">

## formaction, formenctype, formmethod, and formtarget

The formaction, formenctype, formmethod, and formtargetattributes each have a corresponding attribute on the form element, which you’ll be familiar with from HTML4, so let’s run through each of them briefly. These new attributes have been introduced primarily because you may require alternative actions for different submit buttons, as opposed to having several forms in a document.

#### formaction

formaction specifies the file or application that will submit the form. It has the same effect as the action attribute on the form element and can only be used with a submit or image button (type="submit" or type="image"). When the form is submitted, the browser first checks for a formaction attribute; if that isn’t present, it proceeds to look for an action attribute on the form.

<input type="submit" value="Submit"formaction="process.php">

#### formenctype

formenctype details how the form data is encoded with the POST method type. It has the same effect as the enctype attribute on the form element and can only be used with a submit or image button (type="submit" or type="image"). The default value if not included is application/x-www-formurlencoded.

<input type="submit" value="Submit"formenctype="application/x-www-form-urlencoded">

#### formmethod

formmethod specifies which HTTP method (GET, POST, PUT, DELETE) will be used to submit the form data. It has the same effect as the method attribute on the form element and can only be used with a submit or image button (type="submit" or type="image").

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

#### formtarget

formtarget specifies the target window for the form results. It has the same effect as the target attribute on the form element and can only be used with a submit or image button (type="submit" or type="image").

<input type="submit" value="Submit" formtarget="\_self">