HTML Text Basics

One of HTML's main jobs is to give text structure and meaning (also known

as [semantics](https://developer.mozilla.org/en-US/docs/Glossary/semantics),) so that a browser can display it correctly. This article explains the way [HTML](https://developer.mozilla.org/en-US/docs/Glossary/HTML) can be used to structure a page of text by adding headings and paragraphs, emphasizing words, creating lists, and more.

|  |  |
| --- | --- |
| **Prerequisites:** | Basic HTML familiarity, as covered in [Getting started with HTML](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Getting_started). |
| **Objective:** | Learn how to mark up a basic page of text to give it structure and meaning — including paragraphs, headings, lists, emphasis, and quotations. |

The basics: Headings and Paragragh**Edit**

In HTML, each paragraph has to be wrapped in a [<p>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/p) element, like so:

<p>I am a paragraph, oh yes I am.</p>

Each heading has to be wrapped in a heading element:

<h1>I am the title of the story.</h1>

There are six heading elements — [<h1>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/h1), [<h2>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/h2), [<h3>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/h3), [<h4>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/h4), [<h5>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/h5), and [<h6>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/h6). Each element represents a different level of content in the document; <h1> represents the main heading, <h2> represents subheadings, <h3> represents sub-subheadings, and so on.

**Implementing structural hierarchy**

As an example, in a story, <h1> would represent the title of the story, <h2>'s would represent the title of each chapter and <h3>'s would represent sub-sections of each chapter, and so on.

Preferably you should just use a single <h1> per page — this is the top level heading, and all others sit below this in the hierarchy.

* Make sure you use the headings in the correct order in the hierarchy.  Don't use <h3>'s to represent subheadings, followed by <h2>'s to represent sub-subheadings — that doesn't make sense and will lead to weird results.
* Of the six heading levels available, you should aim to use no more than three per page, unless you feel it is necessary. Documents with many levels (i.e. a deep heading hierarchy) become unwieldy and difficult to navigate. On such occasions, it is advisable to spread the content over multiple pages if possible.

**Why do we need semantics?**

Semantics are relied on everywhere around us — we rely on previous experience to tell us what the function of everyday objects is; when we see something, we know what its function will be. So, for example, we expect a red traffic light to mean "stop", and a green traffic light to mean "go". Things can get tricky very quickly if the wrong semantics are applied (do any countries use red to mean "go"? I hope not.)

In a similar vein, we need to make sure we are using the correct elements, giving our content the correct meaning, function, or appearance. In this context the [<h1>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/h1) element is also a semantic element, which gives the text it wraps around the role (or meaning) of "a top level heading on your page."

<h1>This is a top level heading</h1>

By default, the browser will give it a large font size to make it look like a heading (although you could style it to look like anything you wanted using CSS). More importantly, its semantic value will be used in multiple ways, for example by search engines and screen readers (as mentioned above.)

On the other hand, you could make any element *look* like a top level heading. Consider the following:

<span style="font-size: 32px; margin: 21px 0;">Is this a top level heading?</span>

This is a [<span>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/span) element. It has no semantics. You use it to wrap content when you want to apply CSS to it (or do something to it with JavaScript) without giving it any extra meaning (You'll find out more about these later on in the course.) We've applied some CSS to it to make it look like a top level heading, but since it has no semantic value, it will not get any of the extra benefits described above. It is a good idea to use the relevant HTML element for the job.

Lists[**Edit**](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/HTML_text_fundamentals$edit#Lists)

Now let's turn our attention to lists. Lists are everywhere in life — from your shopping list to the list of directions you subconsciously follow to get to your house every day, to the lists of instructions you are following in these tutorials! Lists are everywhere on the Web too, and we've got three different types to worry about.

**Unordered**

Unordered lists are used to mark up lists of items for which the order of the items doesn't matter — let's take a shopping list as an example.

milk

eggs

bread

hummus

The last step is to wrap each list item in a [<li>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/li) (list item) element:

<ul>

<li>milk</li>

<li>eggs</li>

<li>bread</li>

<li>hummus</li>

</ul>

**Ordered**

Ordered lists are lists in which the order of the items *does* matter — let's take a set of directions as an example:

Drive to the end of the road

Turn right

Go straight across the first two roundabouts

Turn left at the third roundabout

The school is on your right, 300 meters up the road

The markup structure is the same as for unordered lists, except that you have to wrap the list items in an [<ol>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/ol) element, rather than <ul>:

<ol>

<li>Drive to the end of the road</li>

<li>Turn right</li>

<li>Go straight across the first two roundabouts</li>

<li>Turn left at the third roundabout</li>

<li>The school is on your right, 300 meters up the road</li>

</ol>

**Nesting lists**

It is perfectly ok to nest one list inside another one. You might want to have some sub-bullets sitting below a top level bullet. Let's take the second list from our recipe example:

<ol>

<li>Remove the skin from the garlic, and chop coarsely.</li>

<li>Remove all the seeds and stalk from the pepper, and chop coarsely.</li>

<li>Add all the ingredients into a food processor.</li>

<li>Process all the ingredients into a paste.</li>

<li>If you want a coarse "chunky" hummus, process it for a short time.</li>

<li>If you want a smooth hummus, process it for a longer time.</li>

</ol>

Since the last two bullets are very closely related to the one before them (they read like sub-instructions or choices that fit below that bullet), it might make sense to nest them inside their own unordered list, and put that list inside the current fourth bullet. This would look like so:

<ol>

<li>Remove the skin from the garlic, and chop coarsely.</li>

<li>Remove all the seeds and stalk from the pepper, and chop coarsely.</li>

<li>Add all the ingredients into a food processor.</li>

<li>Process all the ingredients into a paste.

<ul>

<li>If you want a coarse "chunky" hummus, process it for a short time.</li>

<li>If you want a smooth hummus, process it for a longer time.</li>

</ul>

</li>

</ol>

Try going back to the previous active learning example and updating the second list like this.

Emphasis and importance[**Edit**](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/HTML_text_fundamentals$edit#Emphasis_and_importance)

In human language, we often emphasise certain words to alter the meaning of a sentence, and we often want to mark certain words as important or different in some way. HTML provides various semantic elements to allow us to mark up textual content with such effects, and in this section, we'll look at a few of the most common ones.

**Emphasis**

When we want to add emphasis in spoken language, we *stress* certain words, subtly altering the meaning of what we are saying. Similarly, in written language we tend to stress words by putting them in italics. For example, the following two sentences have different meanings.

I am glad you weren't late.

I am *glad* you weren't *late*.

The first sentence sounds genuinely relieved that the person wasn't late. In contrast, the second one sounds sarcastic or passive-aggressive, expressing annoyance that the person arrived a bit late.

In HTML we use the [<em>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/em) (emphasis) element to mark up such instances. As well as making the document more interesting to read, these are recognised by screen readers and spoken out in a different tone of voice. Browsers style this as italic by default, but you shouldn't use this tag purely to get italic styling. To do that, you'd use a [<span>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/span) element and some CSS, or perhaps an [<i>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/i) element (see below.)

<p>I am <em>glad</em> you weren't <em>late</em>.</p>

**Strong importance**

To emphasize important words, we tend to stress them in spoken language and **bold** them in written language. For example:

This liquid is **highly toxic**.

I am counting on you. **Do not** be late!

In HTML we use the [<strong>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/strong) (strong importance) element to mark up such instances. As well as making the document more useful, again these are recognized by screen readers and spoken in a different tone of voice. Browsers style this as bold text by default, but you shouldn't use this tag purely to get bold styling. To do that, you'd use a [<span>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/span) element and some CSS, or perhaps a [<b>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/b) element (see below.)

<p>This liquid is <strong>highly toxic</strong>.</p>

<p>I am counting on you. <strong>Do not</strong> be late!</p>

You can nest strong and emphasis inside one another if desired:

<p>This liquid is <strong>highly toxic</strong> —

if you drink it, <strong>you may <em>die</em></strong>.</p>

**Italic, bold, underline...**

The elements we've discussed so far have clearcut associated semantics. The situation with [<b>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/b), [<i>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/i), and [<u>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/u) is somewhat more complicated. They came about so people could write bold, italics, or underlined text in an era when CSS was still supported poorly or not at all. Elements like this, which only affect presentation and not semantics, are known as **presentational elements** and should no longer be used, because as we've seen before, semantics is so important to accessibility, SEO, etc.

HTML5 redefined <b>, <i> and <u> with new, somewhat confusing, semantic roles.

Here's the best rule of thumb: it's likely appropriate to use <b>, <i>, or <u> to convey a meaning traditionally conveyed with bold, italics, or underline, provided there is no more suitable element. However, it always remains critical to keep an accessibility mindset. The concept of italics isn't very helpful to people using screen readers, or to people using a writing system other than the Latin alphabet.

* [<i>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/i) is used to convey a meaning traditionally conveyed by italic: Foreign words, taxonomic designation, technical terms, a thought...
* [<b>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/b) is used to convey a meaning traditionally conveyed by bold: Key words, product names, lead sentence...
* [<u>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/u) is used to convey a meaning traditionally conveyed by underline: Proper name, misspelling...

A kind warning about underline: **People strongly associate underlining with hyperlinks.** Therefore, on the Web, it's best to underline only links. Use the <u> element when it's semantically appropriate, but consider using CSS to change the default underline to something more appropriate on the Web. The example below illustrates how it can be done.

<!-- scientific names -->

<p>

The Ruby-throated Hummingbird (<i>Archilocus colubris</i>)

is the most common hummingbird in Eastern North America.

</p>

<!-- foreign words -->

<p>

The menu was a sea of exotic words like <i lang="uk-latn">vatrushka</i>,

<i lang="id">nasi goreng</i> and <i lang="fr">soupe à l'oignon</i>.

</p>

<!-- a known misspelling -->

<p>

Someday I'll learn how to <u>spel</u> better.

</p>

<!-- Highlight keywords in a set of instructions -->

<ol>

<li>

<b>Slice</b> two pieces of bread off the loaf.

</li>

<li>

<b>Insert</b> a tomato slice and a leaf of

lettuce between the slices of bread.

</li>

</ol>

Summary[**Edit**](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/HTML_text_fundamentals$edit#Summary)

That's it for now! This article should have given you a good idea of how to start marking up text in HTML, and introduced you to some of the most important elements in this area. There are a lot more semantic elements to cover in this area, and we'll look at a lot more in our 'More Semantic Elements' article, later on in the course. In the next article, we'll be looking in detail at how to [create hyperlinks](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Creating_hyperlinks), possibly the most important element on the Web.

# HyperLinks

Hyperlinks are really important — they are what makes the Web a web. This article shows the syntax required to make a link, and discusses link best practices.

|  |  |
| --- | --- |
| **Prerequisites:** | Basic HTML familiarity, as covered in [Getting started with HTML](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Getting_started). HTML text formatting, as covered in [HTML text fundamentals](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/HTML_text_fundamentals). |
| **Objective:** | To learn how to implement a hyperlink effectively, and link multiple files together. |

## What is a hyperlink?[Edit](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Creating_hyperlinks$edit#What_is_a_hyperlink)

Hyperlinks are one of the most exciting innovations the Web has to offer. Well, they've been a feature of the Web since the very beginning, but they are what makes the Web a Web — they allow us to link our documents to any other document (or other resource) we want to, we can also link to specific parts of documents, and we can make apps available at a simple web address (contrast this to native apps, which have to be installed and all that business.) Just about any web content can be converted to a link, so that when clicked (or otherwise activated) it will make the web browser go to another web address ([URL](https://developer.mozilla.org/en-US/docs/Glossary/URL).)

## Anatomy of a link

A basic link is created by wrapping the text (or other content, see [Block level links](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Creating_hyperlinks#Block_level_links)) you want to turn into a link inside an [<a>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/a)element, and giving it an [href](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/a#attr-href) attribute (also known as a **Hypertext Reference** , or **target**) that will contain the web address you want the link to point to.

<p>I'm creating a link to

<a href="https://www.mozilla.org/en-US/">the Mozilla homepage</a>.

</p>

This gives us the following result:

I'm creating a link to [the Mozilla homepage](https://www.mozilla.org/en-US/).

### Adding supporting information with the title attribute

Another attribute you may want to add to your links is title; this is intended to contain supplementary useful information about the link, such as what kind of information the page contains, or things to be aware of. For example:

<p>I'm creating a link to

<a href="https://www.mozilla.org/en-US/"

title="The best place to find more information about Mozilla's

mission and how to contribute">the Mozilla homepage</a>.

</p>

This gives us the following result (the title will come up as a tooltip when the link is hovered over):

I'm creating a link to [the Mozilla homepage](https://www.mozilla.org/en-US/).

Distinguish title in link and title in header.

### Block level links

As mentioned before, you can turn just about any content into a link, even [block level elements](https://developer.mozilla.org/en-US/Learn/HTML/Introduction_to_HTML/Getting_started#Block_versus_inline_elements). If you had an image you wanted to turn into a link, you could just put the image between <a></a> tags.

<a href="https://www.mozilla.org/en-US/">

<img src="mozilla-image.png" alt="mozilla logo that links to the mozilla homepage">

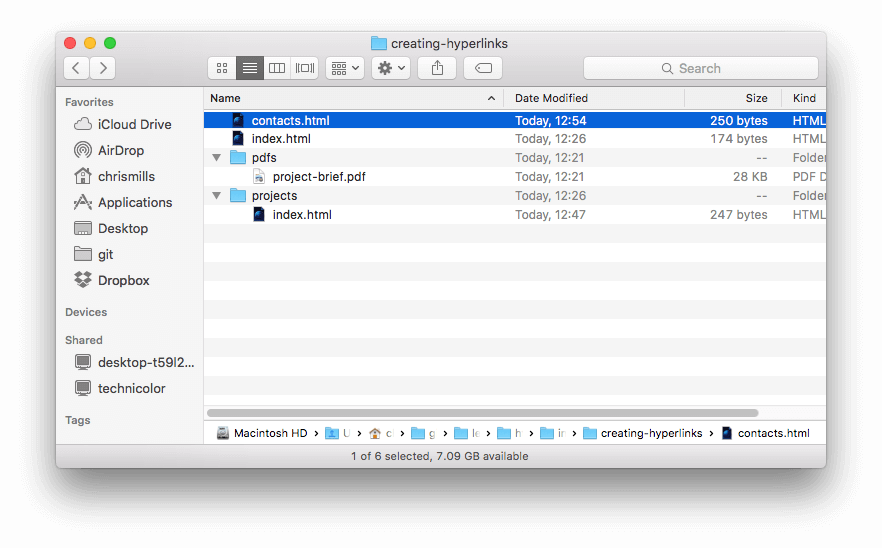
</a>

## A quick primer on URLs and paths

To fully understand link targets, you need to understand URLs and file paths. This section gives you the information you need to achieve this.

A URL, or Uniform Resource Locator is simply a string of text that defines where something is located on the Web. For example Mozilla's English homepage is located at https://www.mozilla.org/en-US/.

URLs use paths to find files. Paths specify where in the filesystem the file you are interested in is located. Let's look at a simple example of a directory structure (see the [creating-hyperlinks](https://github.com/mdn/learning-area/tree/master/html/introduction-to-html/creating-hyperlinks) directory.)



The **root** of this directory structure is called creating-hyperlinks. When working locally with a web site, you will have one directory that the whole site goes inside. Inside the root, we have an index.html file and a contacts.html. In a real website, index.html would be our home page or landing page (a web page that serves as the entry point for a website or a particular section of a website.).

There are also two directories inside our root — pdfs and projects. These each have a single file inside them — a PDF (project-brief.pdf) and an index.html file, respectively. Note how you can quite happily have two index.html files in one project as long as they are in different locations in the filesystem. Many web sites do. The second index.html would perhaps be the main landing page for project-related information.

* **Same directory**: If you wanted to include a hyperlink inside index.html (the top level index.html) pointing to contacts.html, you would just need to specify the filename of the file you want to link to, as it is in the same directory as the current file. So the URL you would use is contacts.html:
* <p>Want to contact a specific staff member?

Find details on our <a href="contacts.html">contacts page</a>.</p>

* **Moving down into subdirectories**: If you wanted to include a hyperlink inside index.html (the top level index.html) pointing to projects/index.html, you would need to go down into the projects directory before indicating the file you want to link to. This is done by specifying the directory's name, then a forward slash, then the name of the file. so the URL you would use is projects/index.html:

<p>Visit my <a href="projects/index.html">project homepage</a>.</p>

* **Moving back up into parent directories**: If you wanted to include a hyperlink inside projects/index.html pointing to pdfs/project-brief.pdf, you'd have to go up a directory level, then back down into the pdf directory. "Go up a directory" is indicated using two dots — .. — so the URL you would use is ../pdfs/project-brief.pdf:

<p>A link to my <a href="../pdfs/project-brief.pdf">project brief</a>.</p>

**Note**: You can combine multiple instances of these features into complex URLs, if needed, e.g. ../../../complex/path/to/my/file.html.

### Document fragments ---id attribute

It is possible to link to a specific part of an HTML document (known as a **document fragment**), rather than just to the top of the document. To do this you first have to assign an [id](https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes#attr-id) attribute to the element you want to link to. It normally makes sense to link to a specific heading, so this would look something like the following:

<h2 id="Mailing\_address">Mailing address</h2>

Then to link to that specific id, you'd include it at the end of the URL, preceded by a hash/pound symbol, for example:

<p>Want to write us a letter? Use our <a href="contacts.html#Mailing\_address">mailing address</a>.</p>

You can even use the document fragment reference on its own to link to another part of the same document:

<p>The <a href="#Mailing\_address">company mailing address</a> can be found at the bottom of this page.</p>

### Absolute versus relative URLs

Two terms you'll come across on the Web are **absolute URL** and **relative URL:**

**absolute URL**: Points to a location defined by its absolute location on the web, including [protocol](https://developer.mozilla.org/en-US/docs/Glossary/protocol) and [domain name](https://developer.mozilla.org/en-US/docs/Glossary/domain_name). So for example, if an index.html page is uploaded to a directory called projects that sits inside the root of a web server, and the web site's domain is http://www.example.com, the page would be available at http://www.example.com/projects/index.html (or even just http://www.example.com/projects/, as most web servers just look for a landing page such as index.html to load if it is not specified in the URL.)

An absolute URL will always point to the same location, no matter where it is used.

**relative URL**: Points to a location that is relative to the file you are linking from, more like what we looked at in the previous section. For example, if we wanted to link from our example file at http://www.example.com/projects/index.html to a PDF file in the same directory, the URL would just be the filename — e.g. project-brief.pdf — no extra information needed. If the PDF was available in a subdirectory inside projects called pdfs, the relative link would be pdfs/project-brief.pdf (the equivalent absolute URL would be http://www.example.com/projects/pdfs/project-brief.pdf.)

A relative URL will point to different places depending on where the file it is used inside is located — for example if we moved our index.html file out of the projects directory and into the root of the web site (the top level, not in any directories), the pdfs/project-brief.pdf relative URL would now point to http://www.example.com/pdfs/project-brief.pdf, not http://www.example.com/projects/pdfs/project-brief.pdf.

## Link best practices

There are some best practices to follow when writing links. Let's look at these now.

### Use clear link wording

It's easy to throw links up on your page. That's not enough. We need to make our links accessible to all readers, regardless of their current context and which tools they prefer. For example:

* Screenreader users like jumping around from link to link on the page, and reading links out of context.
* Search engines use link text to index target files, so it is a good idea to include keywords in your link text to effectively describe what is being linked to.
* Visual readers skim over the page rather than reading every word, and their eyes will be drawn to page features that stand out, like links. They will find descriptive link text useful.

Let's look at a specific example:

***Good*** link text: [Download Firefox](https://firefox.com/)

<p><a href="https://firefox.com/">

Download Firefox

</a></p>

***Bad*** link text: [Click here](https://firefox.com/) to download Firefox

<p><a href="https://firefox.com/">

Click here

</a>

to download Firefox</p>

Other tips:

* Don't repeat the URL as part of the link text — URLs look ugly, and sound even uglier when a screen reader reads them out letter by letter.
* Don't say "link" or "links to" in the link text — it's just noise. Screen readers tell people there's a link. Visual users will also know there's a link, because links are generally styled in a different colour and underlined (this convention generally shouldn't be broken, as users are so used to it.)
* Keep your link label as short as possible — long links especially annoy screen reader users, who have to hear the whole thing read out.
* Minimize instances where multiple copies of the same text are linked to different places. This can cause problems for screenreader users, who will often bring up a list of the links out of context — several links all labelled "click here", "click here", "click here" would be confusing.

### Use relative links wherever possible

From the description above, you might think that it is a good idea to just use absolute links all the time; after all, they don't break when a page is moved like relative links. However, you should use relative links wherever possible when linking to other locations within the same website (when linking to another website, you will need to use an absolute link):

* For a start, it is a lot easier to scan your code — relative URLs are generally a lot shorter than absolute URLs, which makes reading code much easier.
* Second, it is more efficient to use relative URLs wherever possible. When you use an absolute URL, the browser starts by looking up the real location of the server on the Domain Name Server ([DNS](https://developer.mozilla.org/en-US/docs/Glossary/DNS); see [How the web works](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/How_the_Web_works) for more information), then it goes to that server and finds the file that is being requested. With a relative URL on the other hand, the browser just looks up the file that is being requested, on the same server. So if you use absolute URLs where relative URLs would do, you are constantly making your browser do extra work, meaning that it will perform less efficiently.

### Linking to non-HTML resources — leave clear signposts

When linking to a resource that will be downloaded (like a PDF or Word document) or streamed (like video or audio) or has another potentially unexpected effect (opens a popup window, or loads a Flash movie), you should add clear wording to reduce any confusion. It can be quite annoying for example:

* If you are on a low bandwidth connection, click a link and then a multiple megabyte download starts unexpectedly.
* If you haven't got the Flash player installed, click a link and then suddenly get taken to a page that requires Flash.

Let's look at some examples, to see what kind of text can be used here:

<p><a href="http://www.example.com/large-report.pdf">

Download the sales report (PDF, 10MB)

</a></p>

<p><a href="http://www.example.com/video-stream/">

Watch the video (stream opens in separate tab, HD quality)

</a></p>

<p><a href="http://www.example.com/car-game">

Play the car game (requires Flash)

</a></p>

### Use the download attribute when linking to a download

When you are linking to a resource that is to be downloaded rather than opened in the browser, you can use the downloadattribute to provide a default save filename. Here's an example with a download link to the Windows version of Firefox 39:

<a href="https://download.mozilla.org/?product=firefox-39.0-SSL&os=win&lang=en-US"

download="firefox-39-installer.exe">

Download Firefox 39 for Windows

</a>

## Active learning: creating a navigation menu

For this exercise, we'd like you to link some pages together with a navigation menu to create a multi-page website. This is one common way in which a website is created — the same page structure is used on every page, including the same navigation menu, so when links are clicked it gives the impression that you are staying in the same place, and different content is being brought up.

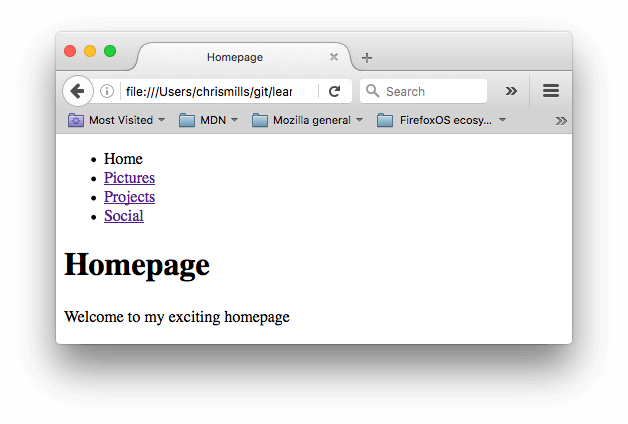
You'll need to make local copies of the following four pages, all in the same directory (see also the [navigation-menu-start](https://github.com/mdn/learning-area/tree/master/html/introduction-to-html/navigation-menu-start) directory for a full file listing):

* [index.html](https://github.com/mdn/learning-area/blob/master/html/introduction-to-html/navigation-menu-start/index.html)
* [projects.html](https://github.com/mdn/learning-area/blob/master/html/introduction-to-html/navigation-menu-start/projects.html)
* [pictures.html](https://github.com/mdn/learning-area/blob/master/html/introduction-to-html/navigation-menu-start/pictures.html)
* [social.html](https://github.com/mdn/learning-area/blob/master/html/introduction-to-html/navigation-menu-start/social.html)

You should:

1. Add an unordered list in the indicated place on one page, containing the names of the pages to link to. A navigation menu is usually just a list of links, so this is semantically ok.
2. Turn each page name into a link to that page.
3. Copy the navigation menu across to each page.
4. On each page, remove just the link to that same page — it is confusing and pointless for a page to include a link to itself, and the lack of a link acts a good visual reminder of what page you are currently on.

The finished example should end up looking something like this:



**Note**: If you get stuck, or are not sure if you have got it right, you can check the [navigation-menu-marked-up](https://github.com/mdn/learning-area/tree/master/html/introduction-to-html/navigation-menu-marked-up) directory to see the correct answer.

## E-mail links

It is possible to create links or buttons that, when clicked, open a new outgoing email message rather than linking to a resource or page. This is done using the [<a>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/a) element and the mailto: URL scheme.

In its most basic and commonly used form, a mailto: link simply indicates the email address of the intended recipient. For example:

<a href="mailto:nowhere@mozilla.org">Send email to nowhere</a>

This results in a link that looks like this: [Send email to nowhere](mailto:nowhere@mozilla.org).

In fact, the email address is even optional. If you leave it out (that is, your [href](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/a#attr-href) is simply "mailto:"), a new outgoing email window will be opened by the user's mail client that has no destination address specified yet. This is often useful as "Share" links that users can click to send an email to an address of their choosing.

### Specifying details

In addition to the email address, you can provide other information. In fact, any standard mail header fields can be added to the mailto URL you provide. The most commonly used of these are "subject", "cc", and "body" (which is not a true header field, but allows you to specify a short content message for the new email). Each field and its value is specified as a query term.

Here's an example that includes a cc, bcc, subject and body:

<a href="mailto:nowhere@mozilla.org?cc=name2@rapidtables.com&bcc=name3@rapidtables.com&amp;subject=The%20subject%20of%20the%20email &amp;body=The%20body%20of%20the%20email">

Send mail with cc, bcc, subject and body

</a>

**Note:** The values of each field must be URL-encoded, that is with non-printing characters (invisible characters like tabs, carriage returns, and page breaks) and spaces [percent-escaped](http://en.wikipedia.org/wiki/Percent-encoding). Also note the use of the question mark (?) to separate the main URL from the field values, and ampersands (&) to separate each field in the mailto: URL. This is standard URL query notation. Read [The GET method](https://developer.mozilla.org/en-US/docs/Learn/HTML/Forms/Sending_and_retrieving_form_data#The_GET_method) to understand what URL query notation is more comonly used for.

Here are a few other sample mailto URLs:

* <mailto:>
* <mailto:nowhere@mozilla.org>
* <mailto:nowhere@mozilla.org,nobody@mozilla.org>
* <mailto:nowhere@mozilla.org?cc=nobody@mozilla.org>
* <mailto:nowhere@mozilla.org?cc=nobody@mozilla.org&subject=This%20is%20the%20subject>

## Summary

That's it for links, for now anyway! You'll return to links later on in the course when you start to look at styling them. Next up for HTML, we'll return to text semantics and look at some more advanced/unusual features that you'll find useful — Advanced text formatting is your next stop.

# Advanced text format

There are many other elements in HTML for formatting text, which we didn't get to in the [HTML text fundamentals](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/HTML_text_fundamentals) article. The elements described in this article are less known, but still useful to know about (and this is still not a complete list by any means). In here you'll learn about marking up quotations, description lists, computer code and other related text, subscript and superscript, contact information, and more.

|  |  |
| --- | --- |
| **Prerequisites:** | Basic HTML familiarity, as covered in [Getting started with HTML](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Getting_started). HTML text formatting, as covered in [HTML text fundamentals](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/HTML_text_fundamentals). |
| **Objective:** | To learn how to use lesser-known HTML elements to mark up advanced semantic features. |

## Description lists[Edit](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Advanced_text_formatting$edit#Description_lists)

In HTML text fundamentals, we walked through how to [mark up basic lists](https://developer.mozilla.org/en-US/Learn/HTML/Introduction_to_HTML/HTML_text_fundamentals#Lists) in HTML, but we didn't mention the third type of list you'll occasionally come across — **description lists**. The purpose of these lists is to mark up a set of items and their associated descriptions, such as terms and definitions, or questions and answers. Let's look at an example of a set of terms and definitions:

Description lists use a different wrapper than the other list types — [<dl>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/dl); in addition each term is wrapped in a [<dt>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/dt) (description term) element, and each description is wrapped in a [<dd>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/dd) (description definition) element. Let's finish marking up our example:

<dl>

<dt>soliloquy</dt>

<dd>In drama, where a character speaks to themselves, representing their inner thoughts or feelings and in the process relaying them to the audience (but not to other characters.)</dd>

<dt>monologue</dt>

<dd>In drama, where a character speaks their thoughts out loud to share them with the audience and any other characters present.</dd>

<dt>aside</dt>

<dd>In drama, where a character shares a comment only with the audience for humorous or dramatic effect. This is usually a feeling, thought or piece of additional background information.</dd>

</dl>

The browser default styles will display description lists with the descriptions indented somewhat from the terms. MDN's styles follow this convention fairly closely, but also embolden the terms for extra definition.

## Quotations[Edit](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Advanced_text_formatting$edit#Quotations) distinguish with link

HTML also has features available for marking up quotations; which element you use depends on whether you are marking up a block or inline quotation.

### Blockquotes

If a section of block level content (be it a paragraph, multiple paragraphs, a list, etc.) is quoted from somewhere else, you should wrap it inside a [<blockquote>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/blockquote) element to signify this, and include a URL pointing to the source of the quote inside a [cite](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/blockquote#attr-cite)attribute. For example, the following markup is taken from the MDN <blockquote> element page:

<p>The <strong>HTML <code>&lt;blockquote&gt;</code> Element</strong> (or <em>HTML Block

Quotation Element</em>) indicates that the enclosed text is an extended quotation.</p>

To turn this into a block quote, we would just do this:

<blockquote cite="https://developer.mozilla.org/en-US/docs/Web/HTML/Element/blockquote">

<p>The <strong>HTML <code>&lt;blockquote&gt;</code> Element</strong> (or <em>HTML Block

Quotation Element</em>) indicates that the enclosed text is an extended quotation.</p>

</blockquote>

Browser default styling will render this as an indented paragraph, as an indicator that it is a quote; MDN does this, but also adds some extra styling:

The **HTML <blockquote> Element** (or HTML Block Quotation Element) indicates that the enclosed text is an extended quotation.

### Inline quotations

Inline quotations work in exactly the same way, except that they use the [<q>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/q) element. For example, the below bit of markup contains a quotation from the MDN <q> page:

<p>The quote element — <code>&lt;q&gt;</code> — is <q cite="https://developer.mozilla.org/en-US/docs/Web/HTML/Element/q">intended

for short quotations that don't require paragraph breaks.</q></p>

Browser default styling will render this as normal text put in quotes to indicate a quotation, like so:

The quote element — <q> — is intended for short quotations that don't require paragraph breaks.

### Citations

The content of the [cite](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/blockquote#attr-cite) attribute sounds useful, but unfortunately browsers, screenreaders, etc. don't really do much with it. There is no way to get the browser to display the contents of cite, without writing your own solution using JavaScript or CSS. If you want to make the source of the quotation available on the page, a better way to mark it up is put the [<cite>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/cite) element next to the quote element. This is really meant to contain the name of the quote source — i.e. the name of the book, or name of the person that said the quote — but there is no reason why you couldn't link the text inside <cite> to the quote source in some way:

<p>According to the <a href="https://developer.mozilla.org/en-US/docs/Web/HTML/Element/blockquote">

<cite>MDN blockquote page</cite></a>:

</p>

<blockquote cite="https://developer.mozilla.org/en-US/docs/Web/HTML/Element/blockquote">

<p>The <strong>HTML <code>&lt;blockquote&gt;</code> Element</strong> (or <em>HTML Block

Quotation Element</em>) indicates that the enclosed text is an extended quotation.</p>

</blockquote>

<p>The quote element — <code>&lt;q&gt;</code> — is <q cite="https://developer.mozilla.org/en-US/docs/Web/HTML/Element/q">intended

for short quotations that don't require paragraph breaks.</q> -- <a href="https://developer.mozilla.org/en-US/docs/Web/HTML/Element/q">

<cite>MDN q page</cite></a>.</p>

Citations are styled in italic font by default. You can see this code at work in our [quotations.html](https://github.com/mdn/learning-area/blob/master/html/introduction-to-html/advanced-text-formatting/quotations.html) example.

### Active learning: Who said that?

Time for another active learning example! In this example we'd like you to:

1. Turn the middle paragraph into a blockquote, which includes a cite attribute.
2. Turn part of the third paragraph into an inline quote, which includes a cite attribute.
3. Include a <cite> element for each quote

Search online to find appropriate quote sources.

If you make a mistake, you can always reset it using the Reset button. If you get really stuck, press the Show solution button to see the answer.

Open in CodePenOpen in JSFiddle

## Abbreviations[Edit](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Advanced_text_formatting$edit#Abbreviations)

Another fairly common element you'll meet when looking around the Web is [<abbr>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/abbr) — this is used to wrap around an abbreviation or acronym, and provide a full expansion of the term (included inside a [title](https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes#attr-title) attribute.) Let's look at a couple of examples:

<p>We use <abbr title="Hypertext Markup Language">HTML</abbr> to structure our web documents.</p>

<p>I think <abbr title="Reverend">Rev.</abbr> Green did it in the kitchen with the chainsaw.</p>

These will come out looking something like this (the expansion will appear in a tooltip when the term is hovered over):

We use HTML to structure our web documents.

I think Rev. Green did it in the kitchen with the chainsaw.

**Note**: There is another element, [<acronym>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/acronym), which basically does the same thing as <abbr>, and was intended specifically for acronyms rather than abbreviations. This however has fallen into disuse — it wasn't supported as well in browsers as well as <abbr>, and has such as similar function that it was felt pointless to have both. Just use <abbr> instead.

## Marking up contact details[Edit](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Advanced_text_formatting$edit#Marking_up_contact_details)

HTML has an element for marking up contact details — [<address>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/address). This simply wraps around your contact details, for example:

<address>

<p>Chris Mills, Manchester, The Grim North, UK</p>

</address>

One thing to remember however is that the <address> element is meant for marking up the contact details of the person who wrote the HTML document, not any address. So the above would only be ok if Chris had written the document the markup appears on. Note that something like this would also be ok:

<address>

<p>Page written by <a href="../authors/chris-mills/">Chris Mills</a>.</p>

</address>

## Superscript and subscript[Edit](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Advanced_text_formatting$edit#Superscript_and_subscript)

You will occasionally need to use superscript and subscript when marking up items like dates, chemical formulae, and mathematical equations so they have the correct meaning. The [<sup>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/sup) and [<sub>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/sub) elements handle this job. For example:

<p>My birthday is on the 25<sup>th</sup> of May 2001.</p>

<p>Caffeine's chemical formula is C<sub>8</sub>H<sub>10</sub>N<sub>4</sub>O<sub>2</sub>.</p>

<p>If x<sup>2</sup> is 9, x must equal 3 or -3.</p>

The output of this code looks like so:

My birthday is on the 25th of May 2001.

Caffeine's chemical formula is C8H10N4O2.

If x2 is 9, x must equal 3 or -3.

## Representing computer code[Edit](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Advanced_text_formatting$edit#Representing_computer_code)

There are a number of elements available for marking up computer code using HTML:

* [<code>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/code): For marking up generic pieces of computer code.
* [<pre>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/pre): For retaining whitespace (generally code blocks) — if you use indentation or excess whitespace inside your text, browsers will ignore it and you will not see it on your rendered page. If you wrap the text in <pre></pre> tags however, your whitespace will be rendered identically to how you see it in your text editor.
* [<var>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/var): For specifically marking up variable names.
* [<kbd>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/kbd): For marking up keyboard (and other types of) input entered into the computer.
* [<samp>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/samp): For marking up the output of a computer program.

Let's look at a few examples. You should try having a play with these (try grabbing a copy of our [other-semantics.html](https://github.com/mdn/learning-area/blob/master/html/introduction-to-html/advanced-text-formatting/other-semantics.html) sample file):

<pre><code>var para = document.querySelector('p');

para.onclick = function() {

alert('Owww, stop poking me!');

}</code></pre>

<p>You shouldn't use presentational elements like <code>&lt;font&gt;</code> and <code>&lt;center&gt;</code>.</p>

<p>In the above JavaScript example, <var>para</var> represents a paragraph element.</p>

<p>Select all the text with <kbd>Ctrl</kbd>/<kbd>Cmd</kbd> + <kbd>A</kbd>.</p>

<pre>$ <kbd>ping mozilla.org</kbd>

<samp>PING mozilla.org (63.245.215.20): 56 data bytes

64 bytes from 63.245.215.20: icmp\_seq=0 ttl=40 time=158.233 ms</samp></pre>

The above code will look like so:

Open in CodePenOpen in JSFiddle

## Marking up times and dates[Edit](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Advanced_text_formatting$edit#Marking_up_times_and_dates)

HTML also provides the [<time>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/time) element for marking up times and dates in a machine-readable format. For example:

<time datetime="2016-01-20">20 January 2016</time>

Why is this useful? Well, there are many different ways that humans write down dates. The above date could be written as:

* 20 January 2016
* 20th January 2016
* Jan 20 2016
* 20/06/16
* 06/20/16
* The 20th of next month
* 20e Janvier 2016
* 2016年1月20日
* And so on

But these different forms cannot be easily recognised by the computers — what if you wanted to automatically grab the dates of all events in a page and insert them into a calendar? The [<time>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/time) element allows you to attach an unambiguous, machine-readable time/date for this purpose.

The basic example above just provides a simple machine readable date, but there are many other options that are possible, for example:

<!-- Standard simple date -->

<time datetime="2016-01-20">20 January 2016</time>

<!-- Just year and month -->

<time datetime="2016-01">January 2016</time>

<!-- Just month and day -->

<time datetime="01-20">20 January</time>

<!-- Just time, hours and minutes -->

<time datetime="19:30">19:30</time>

<!-- You can do seconds and milliseconds too! -->

<time datetime="19:30:01.856">19:30:01.856</time>

<!-- Date and time -->

<time datetime="2016-01-20T19:30">7.30pm, 20 January 2016</time>

<!-- Date and time with timezone offset-->

<time datetime="2016-01-20T19:30+01:00">7.30pm, 20 January 2016 is 8.30pm in France</time>

<!-- Calling out a specific week number-->

<time datetime="2016-W04">The fourth week of 2016</time>

## Summary[Edit](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Advanced_text_formatting$edit#Summary)

That marks the end of our study of HTML text semantics. Bear in mind that what you have seen during this course is not an exhaustive list of HTML text elements — we wanted to try to cover the essentials, and some of the more common ones you will see in the wild, or at least might find interesting. To find way more HTML elements, you can take a look at our [HTML element reference](https://developer.mozilla.org/en-US/docs/Web/HTML/Element)(the [Inline text semantics](https://developer.mozilla.org/en-US/docs/Web/HTML/Element#Inline_text_semantics) section would be a great place to start.) In the next article we will look at the HTML elements you'd use to structure the different parts of an HTML document.

< b > < i > 是视觉要素（presentationl elements），分别表示无意义的加粗，无意义的斜体，表现样式为 { font-weight: bolder }，仅仅表示「这里应该用粗体显示」或者「这里应该用斜体显示」，此两个标签在HTML4.01中并不被推荐使用；

< em > 和 < strong > 是表达要素(phrase elements)。 < em > （emphasized text）表示一般的强调文本，而 < strong > （strong emphasized text）表示比 < em > 语义更强的的强调文本。

而在新的 HTML5 工作草案 中：

< em > 和 < strong > 仍旧是表达要素(phrase elements)。但这时的 < strong > 表示html页面上的强调（emphasized text）， < em > 表示句子中的强调（即强调语义）