Creating 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.

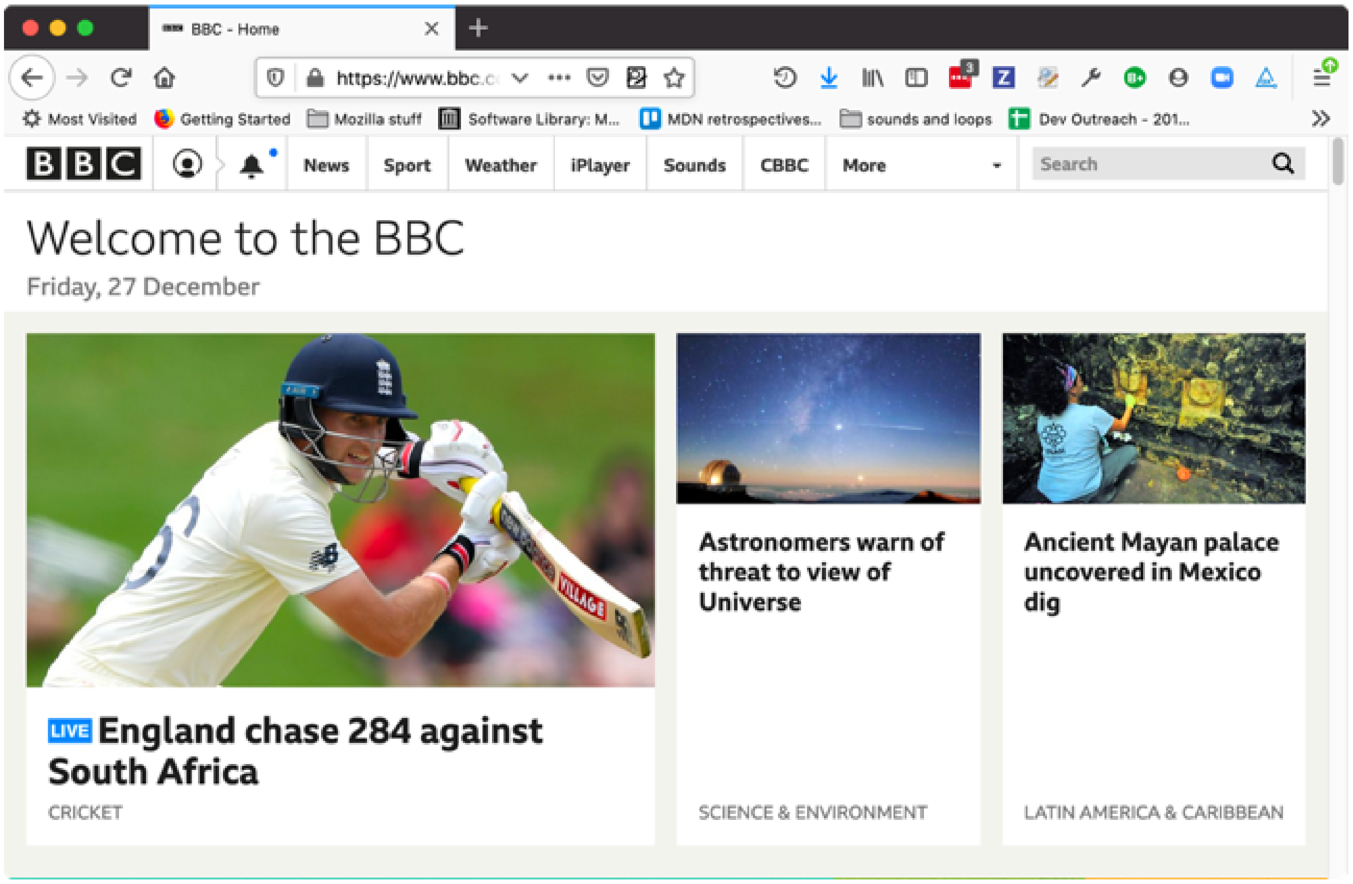
Prerequisites:	Basic HTML familiarity, as covered in Getting started with HTML. HTML text formatting, as covered in HTML text fundamentals.
Objective:	To learn how to implement a hyperlink effectively, and link multiple files together.

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).

Note: A URL can point to HTML files, text files, images, text documents, video and audio files, and anything else that can live on the Web. If the web browser doesn't know how to display or handle the file, it will ask you if you want to open the file (in which case the duty of opening or handling the file is passed to a suitable native app on the device) or download the file (in which case you can try to deal with it later on.)

The BBC homepage, for example, contains a large number of links that point not only to multiple news stories, but also different areas of the site (navigation functionality), login/registration pages (user tools) and more.



Welcome to the BBC

Friday, 27 December

LIVE England chase 284 against South Africa

CRICKET

Astronomers warn of threat to view of Universe

SCIENCE & ENVIRONMENT

Ancient Mayan palace uncovered in Mexico dig

LATIN AMERICA & CARIBBEAN

Anatomy of a link

A basic link is created by wrapping the text (or other content, see Block level links) you want to turn into a link inside an `<a>` element, and giving it an `href` attribute (also known as a **Hypertext Reference** , or **target**) that will contain the web address you want the link to point to.

```html
<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.

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:

```html
<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.

Note: A link title is only revealed on mouse hover, which means that people relying on keyboard controls to navigate web pages will have difficulty accessing title information. If a title's information is truly important to the usability of page, then you should present it in a manner that will be accessible to all users, for example by putting it in the regular text.

Active learning: creating your own example link

Active learning time: we'd like you to create an HTML document using your local code editor (our getting started template would do just fine.)

- Inside the HTML body, try adding one or more paragraphs or other types of content you already know about.
- Turn some of the content into links.
- Include title attributes.

Block level links

As mentioned before, you can turn just about any content into a link, even block level 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
</a>
```

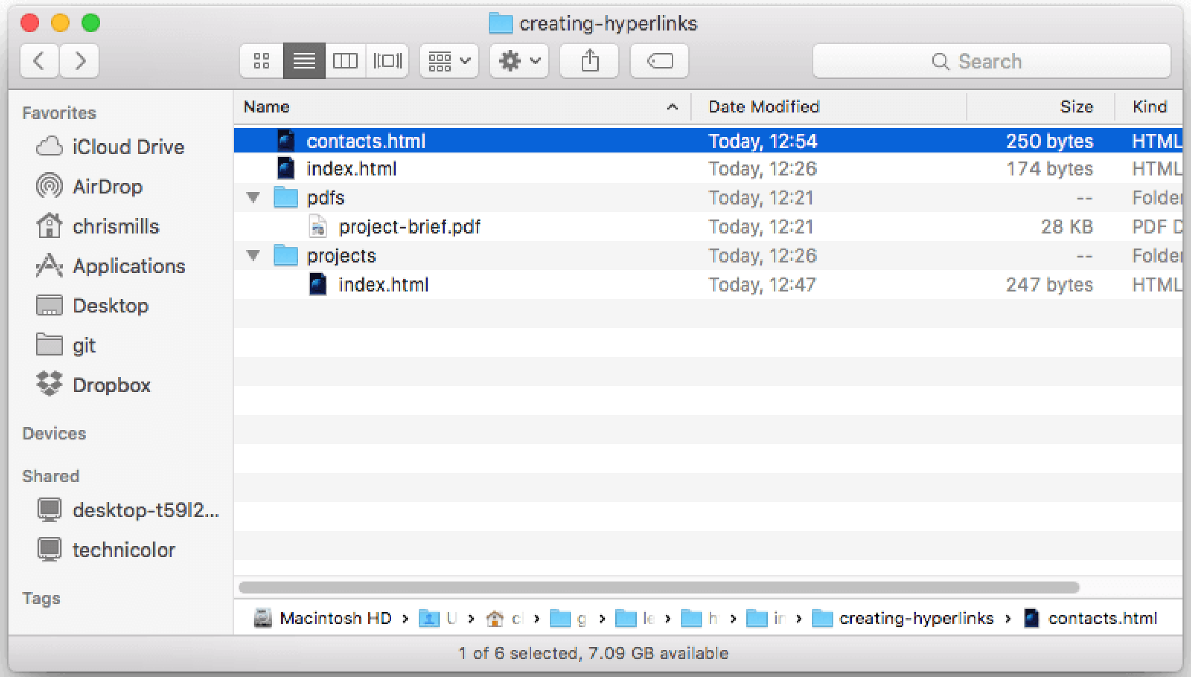
Note: You'll find out a lot more about using images on the Web in a future article.

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 directory.)



Name	Date Modified	Size	Kind
contacts.html	Today, 12:54	250 bytes	HTML
index.html	Today, 12:26	174 bytes	HTML
▼ pdfs	Today, 12:21	--	Folde
project-brief.pdf	Today, 12:21	28 KB	PDF D
▼ projects	Today, 12:26	--	Folde
index.html	Today, 12:47	247 bytes	HTML

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`:

```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>.</
```

- **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
```

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

Document fragments

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` 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_addres
```

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
```

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 and 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 the actual location of the file you refer from — 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 link inside it would now point to a file located at `http://www.example.com/pdfs/project-brief.pdf`, not a file located at `http://www.example.com/projects/pdfs/project-brief.pdf`.

Of course, the location of the `project-brief.pdf` file and `pdfs` folder won't suddenly change because you moved the `index.html` file — this would make your link point to the wrong place, so it wouldn't work if clicked on. You need to be careful!

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

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

Bad *link text:* Click here 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 System (DNS; see 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:

```html
<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/" target="_blank">
  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 `download` attribute to provide a default save filename. Here's an example with a download link to the latest Windows version of Firefox:

```html
<a href="https://download.mozilla.org/?product=firefox-latest-ssl&os=win64
   download="firefox-latest-64bit-installer.exe">
  Download Latest Firefox for Windows (64-bit) (English, US)
</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.

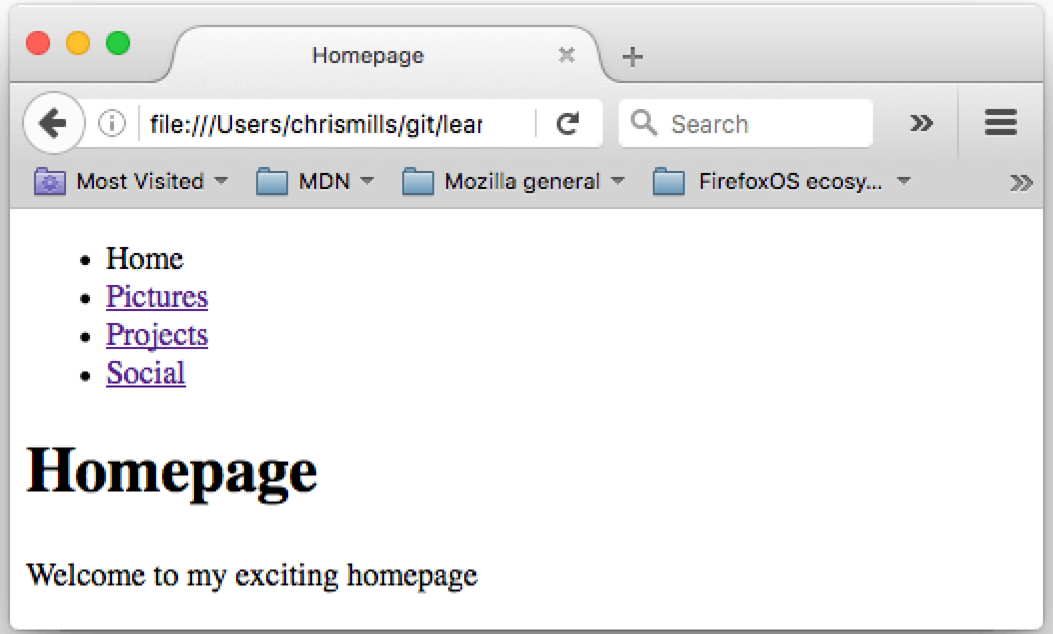
You'll need to make local copies of the following four pages, all in the same directory (see also the navigation-menu-start directory for a full file listing):

- index.html
- projects.html
- pictures.html
- 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:



- Home
- Pictures
- Projects
- Social

Homepage

Welcome to my exciting homepage

Note: If you get stuck, or are not sure if you have got it right, you can check the 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>` 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.

In fact, the email address is even optional. If you leave it out (that is, your `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@rap
  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. 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 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

Test your skills!

You've reached the end of this article, but can you remember the most important information? You can find some further tests to verify that you've retained this information before you move on — see Test your skills: Links.

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.

Previous · Overview: Introduction to HTML · Next

In this module

Last modified: Feb 6, 2020, by MDN contributors

Related Topics

Further resources

▸ Common questions

How to contribute