



Absolute vs. Relative Paths

By: [Bryan Ashcraft](#)

 Page 1 of 1

[Set for printing](#)

When creating hyperlinks in a web document, you have two options. You can define an absolute or relative path to the target object. While there is only one way to make an absolute link path, there are two options when using relative paths. First, and most often used, is a document relative path, and second is a root relative path. In this article we will look at each option and discuss when each type is an appropriate choice.

Absolute Paths

An absolute path is created whenever your link uses the full URL of an object or page. For instance, <http://www.communitymx.com> is an absolute path to a specific web site. This method is the best choice whenever you need to send a visitor to another site or need to get content from another site. While you can use it within your own site, there is rarely ever a need to do so.

Relative Paths

As you can probably guess, relative paths are used much more frequently than absolute paths. Any time you need to send a visitor to another page within your site or include an object from your site (like an image) on one of your pages a relative link will work just fine. Which form of relative link you should use will depend on how the site is constructed. Document and root relative paths each have a place in the building of a web site. Read on for a break down of each type.

Document Relative

Document relative paths are the most widely used type of relative linking, and for good reason. They can easily move up and down your site's hierarchy, in and out of directories (folders) without a hitch. So what does a document relative path look like? Let's say you have a site structure like Figure 1.

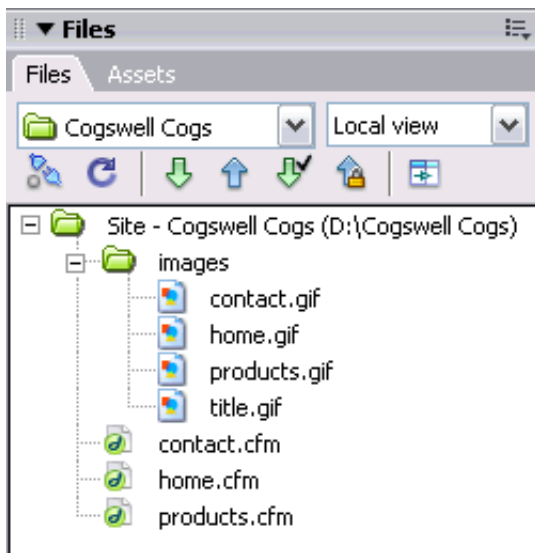


Figure 1 *Site Hierarchy*

Now you need to include the title gif on every page, and a header gif (home, products, contact) on its respective page. Using a document relative path the code for the home page would look something this:

```
  

```

Granted this isn't an accurate representation as you would have a width and height defined for the images and they would be in some type of formal layout. But you should have an idea of what a document relative path looks like.

Now, let's say, instead of the products page being in the same directory as the home page it is actually located in a subdirectory. You still need to include the images in the page but the page is in a directory (folder) deeper than the root level of your site (see Figure 2.)

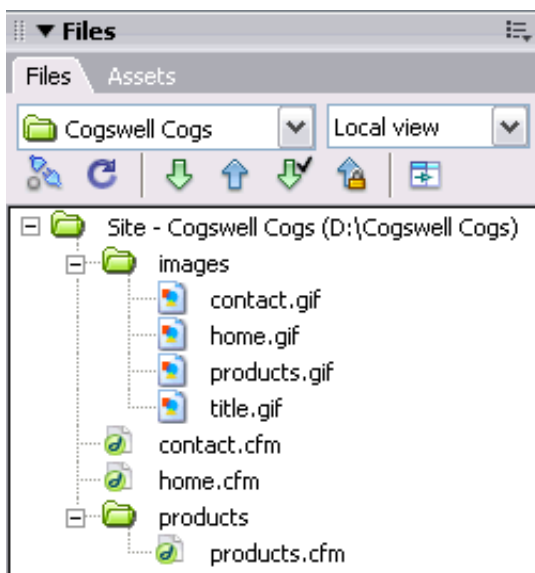


Figure 2 *Site with a Subdirectory*

As you can see you need to move up and out of the products directory in order to get to the images folder in order to access the needed images. The code for this would like what you see below.

```
  

```

Notice that the path now has "../../" at the beginning. This basically says go up one directory. So the full

instructions would be "Go up one directory then into the images folder and grab the requested image file". This will work no matter how many subdirectories down your page is in the site. If you need to move up three directories and into the images folder the path would be something like the following.

```
  

```

As you can see document relative paths are very flexible and very useful for almost any need. So why would you ever need to use a root relative path? Read the next section to find out.

Root Relative

Well, after seeing how useful document relative paths are, you might be wondering why you would ever need to use root relative paths at all. Trust me, root relative paths can play a very important role, under certain circumstances, in ensuring a site looks and functions properly.

One instance in which a root relative path is not only appropriate, but actually required, is when you use "included" files in a site that has multiple directories. Included files actually are files that are "included", or pulled into, other pages. Using "included" files can greatly decrease the development time of a site. For example, let's take the site we have been using during this article. Now let's say our page header is the same on every page. Instead having to keep recreating it on each page we can make it its own file and include it, or pull it, into every page on the site. So let's say we have the following site hierarchy (figure 3).

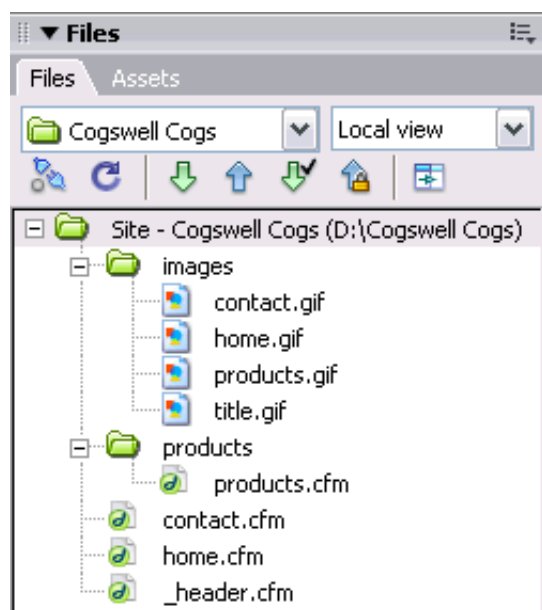


Figure 3 Site that uses an "included" header file

We set up our title image in the header file and saved the file as "_header.cfm":

```

```

NOTE: The code necessary to include one file in another is beyond the scope of this article, since it can vary based on the server language used.

When a visitor comes to the site, the home page pulls in the code from the header file which says "Go into the images folder and get the title.gif image" and displays the image on the page. "This works great!", we're thinking, so we add the code to pull in the header file on every page. Now a visitor comes to our site, sees the home page in all its glory and then decides to take a look at the products we have to offer. Uh-oh! Our

products page pulls in the header file and reads the code that says "Go into the images folder and get the title.gif image." This doesn't work since there is no images folder in our products folder. So, they get the dreaded red x. This is not good. The products page needs instructions that say, "Go up one directory, into the images folder, and grab the title.gif image."

You might be thinking "Well so much for this code reuse and speedier development." Not so fast. This is where root relative paths are invaluable. A root relative path always starts at the root of the site and works down. So no matter where the page is within the site's hierarchy, if you use a root relative path, that link will always start at the root of the site in order to get where it is going. So, if we use a root relative path instead of a document relative one in our header file, that file will always find the right target for our link. Regardless of where the code is placed in our site.

Now, after all the rambling, what does a root relative path look like? Surprisingly similar to a document relative path. To create a root relative path all we have to do is add a forward slash (/) to the beginning of our path.

```

```

Amazing how important the one character can be isn't it? But by adding that one character our "included" header file will work in every page of our site. Thus enabling code reuse and speedier site development.

Conclusion

Hopefully this article has helped you understand the linking options available in HTML, and the appropriate time to apply them. From now on there should never be a time when you have to ask "How do I link this image or that file?"

Approximate download size: 126k

Page 1 of 1

1



[Download Support Files](#)

Keywords

HTML, linking, page linking, object linking, absolute paths, relative paths, root relative, document relative