File Structure

```
my_website
    about
        index.html
    css
    images
    index.html
    fonts
    projects
        project-1
            project-1.html
        project-2
            project-2.html
```

Anchors

Links are created with the <a> element, which stands for “anchor”.

It works just like all the elements in the previous chapter: when you wrap some text in <a> tags, it alters the meaning of that content.

index.html

```
<p>This example is about links and <a>images</a>.</p>
```

Links

HTML **attributes** add meaning to the element it’s attached to.

Different elements take different attributes. **href** is an attribute that’s used for links.

index.html

```
<p>This example is about links and
<a href=’images.html’>images</a>.</p>
```

Absolute, Relative, and Root-Relative Links

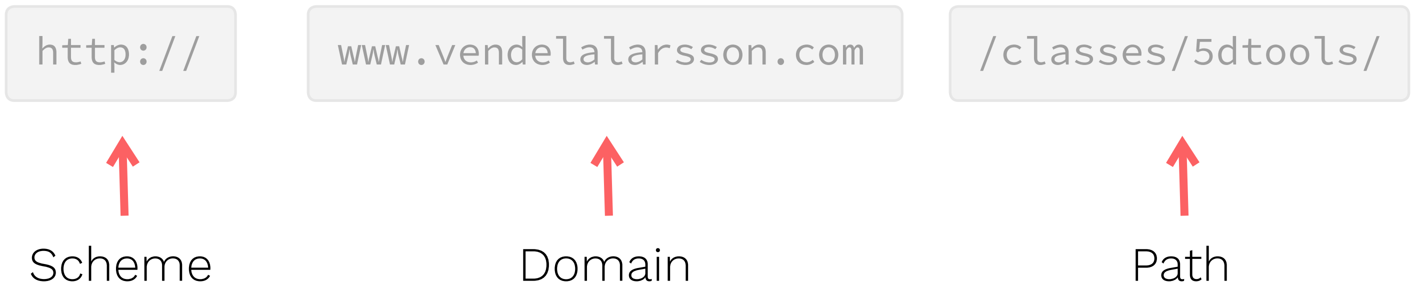
A website is just a collection of HTML files organized into folders.

To refer to those files from inside another file, the Internet uses “uniform resource locators” (URLs).

URLs can take different forms depending on where they’re leading to.

Absolute Links

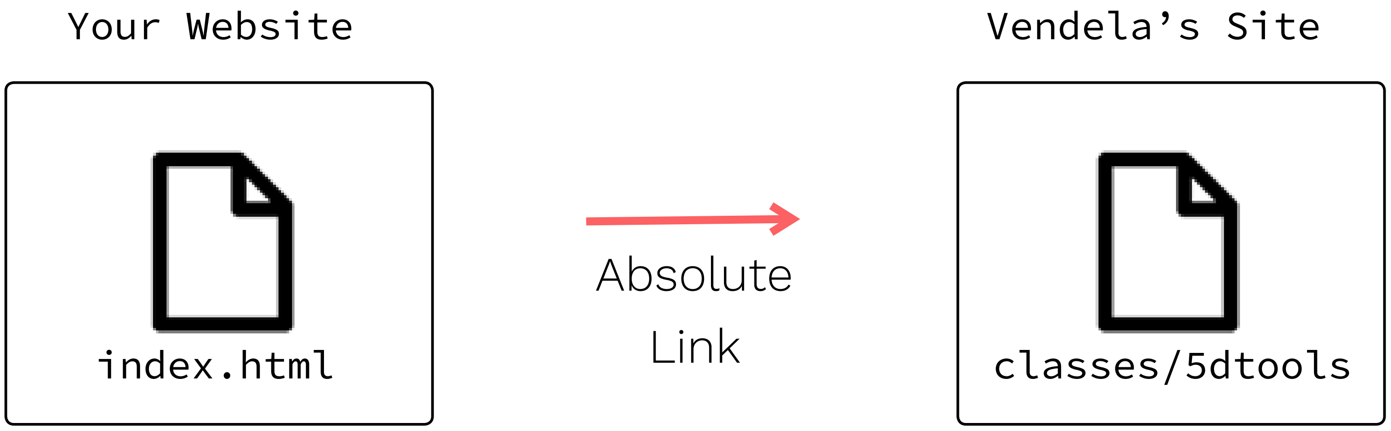
Most detailed URLs



```
http://   www.vendelalarsson.com   /classes/5dtools/
```

index.html

```
<p>Here is <a href='http://www.vendelalarsson.com/
classes/5dtools/'>my website</a>.</p>
```



Relative Links

Point to another file in your website from the vantage point of the file you're editing. It's implied that the scheme and domain name are the same as the current page, so the only thing you need to supply is the path.

index.html

```
<p>Link to the <a href='about/index.html'>about page</a>.</p>
```

In this case, the **href** attribute represents the file path to index.html (within the about folder) from the index.html file in the root. Since index.html isn't in the same folder as index.html, we need to include the about folder in the URL

Each folder and file in a path is separated by a forward slash (/). So, if we were trying to get to a file that was two folders deep, we'd need a URL like this:

```
projects/project-1/project-1.html
```

```
my_website
    about
        index.html
    css
    images
    index.html
    fonts
    projects
        project-1
            project-1.html
        project-2
            project-2.html
```

That works for referring to files that are in the same folder or a deeper folder. What about linking to pages that are in a directory above the current file?

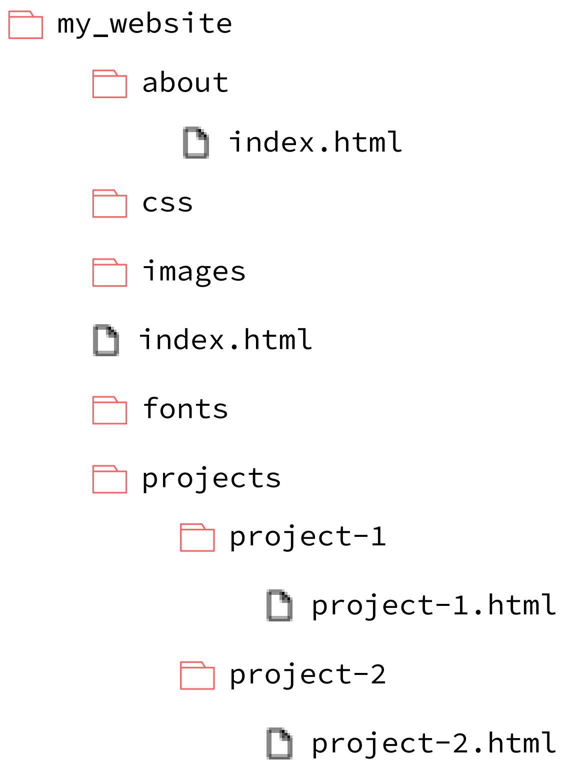
```
<p>This is the about page, but
what if I want to go link to <a
href=”project-1.html”>project-1</
a>?</p>
```

```
my_website
    about
        index.html
    css
    images
    index.html
    fonts
    projects
        project-1
            project-1.html
        project-2
            project-2.html
```

If you do this, it will complain that the file doesn’t exist.

It’s looking in the wrong directory. It expects there to be a project-1.html file in the about directory. To fix this and get to the right place we need the `..` syntax.

```
<p>This is the about page, but what
if I want to go link to <a href=”../
projects/project-1/project-1.htm-
l”>project-1</a>?</p>
```



Now the browser is looking at the parent directory of about, and then going into the projects directory to look for the project-1 directory where the project-1.html file is located.

To go up more than one directory level, you can use `../..`

```
<p>If I’m currently in the proj-
ect-1.html file but want to link to
the <a href=”../../about/”>about
page</a> I can do this.</p>
```

```
my_website
    about
        index.html
    css
    images
    index.html
    fonts
    projects
        project-1
            project-1.html
        project-2
            project-2.html
```

Hold on… why are there so many folders?

Go to this link, and check out the URLs.

When an HTML file is not within a folder, the URL includes the .html portion which looks more sloppy. Keeping your files within their own directories keeps your URLs cleaner.

Ok continue… Root-Relative Links

Similar to the previous section, but instead of being relative to the current page, they’re relative to the “root” of the entire website.

This entire class has used local HTML files instead of a website hosted on a web server.

The only difference between a root-relative link and a relative one is that the former starts with a forward slash. That initial forward slash represents the root of your site.

You can add more folders and files to the path after that initial slash, just like relative links. The following path will work correctly no matter where the current page is located

```
/about/index.html
```

(and since I'm using folders to contain these files, I can even just write this:)

```
/about/
```

```
my_website
    about
        index.html
    css
    images
    index.html
    fonts
    projects
        project-1
            project-1.html
        project-2
            project-2.html
```

Link Targets

By default, most browsers replace the current page with the new one. We can use the target attribute to ask the browser to open a link in a new window/tab.

index.html

```
<p>Open a new tab with <a href="/about/" target="_
blank">this link</a> please!</p>
```

Naming Conventions

Keep it lowercase (for consistency). Don’t use spaces. Example:

```
root
    spaces are bad.html
```

in the browser

```
spaces%20are%20bad.html
```

Keep in mind that what you name your files and folders is public. Everyone can see your URLs once your website is live. This applies to all the files on your site, not just pages.

Better:

```
lowercase-no-spaces
    index.html
```

```
in the browser
```

```
/lowercase-no-spaces/
```

Images

We can use the same linking conventions we just learned to link images to our document! The syntax looks like this:

index.html

```
<img src=”my-image.jpg” />
```

Absolute

🗋 index.html

```
<img src=”https://vendelalarsson.com/my-image.jpg” />
```

Relative

🗋 index.html

```
<img src=”../images/my-image.jpg” />
```

Root-relative

🗋 index.html

```
<img src=”/images/my-image.jpg” />
```