

Refining Markup Details

In this lesson, we will refine some markup details and learn to work with URLs in HTML.

WE'LL COVER THE FOLLOWING



- Working with URLs
- Listing 5-6: Exercise-05-06/index.html
- Listing 5-7: Exercise-05-06/Region1Page1.html
- Listing 5-8: Exercise-05-06/PageWithFragment.html
- Listing 5-9: Exercise-05-07/index.html

In the chapters you have already gone through, you have seen a number of HTML elements, and you have utilized them in exercises and samples. Most exercises had a “*How It Works*” section that explained important details, however, there were important subtleties not treated there.

In this section you will learn the most important nitty-gritty markup details that are crucial to know before diving into deeper subjects.

Working with URLs

When creating an HTML page, there are many markup elements that require you to specify URLs where URL stands for *Uniform Resource Locator*. The first that probably jumps into your mind is the `<a>` element's `href` attribute, which contains a link to a target page. There are other elements where URLs play an important role, and we can divide them into groups just to see that there are different situations where you can meet URLs:

You can use them to link to other documents and resources. The previously mentioned `<a>` tag is one example, but you can also use the `<link>` tag in this context; for example, when you use it to point to the next document, provided the current document is a part of a series.

You often link external style sheets and scripts with the `<link>` and `<script>` tags, and here you refer those external files with URLs. Your web page may cite an external reference, such as the `<q>`, `<blockquote>`, `<ins>`, and `` elements, and they use URLs.

To provide rich user experience, you often include images, audio, and video resources in your pages and all related markup elements specify the source of these media through URLs.

Forms are submitted to web servers, and their target is defined with URLs, too.

It is time to examine how you can access resources through URLs with the markup. The resources can be within your website, can point to external websites somewhere in the intranet of your company, or even can address other locations in the internet.


To reach all kind of resources, you can use absolute and relative URLs that might be extended with an optional fragment identifier.

An absolute URL is independent from any relationship, when you use it, you point directly to a resource. For example, the following URL is an absolute one: <http://othercompany.com/pages/catalog.html>.


It does not matter in which page you use it, this resource identifier undoubtedly names the resource it wants to use.

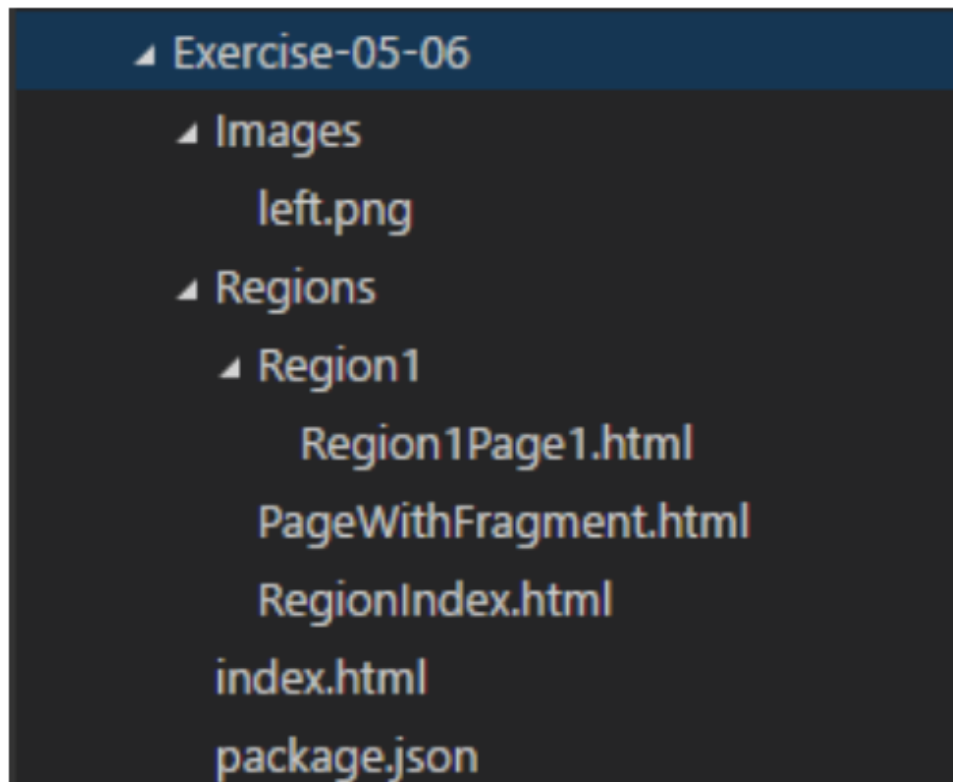
In contrast, relative URLs always specify resources in relation to the current location of the web page that refers to the resource.

For example, let's assume, that a page is in the <http://mycompany.com/Admin> folder (the name of the page is irrelevant). If this page uses the `/Images/logo.png` resource identifier, it is a URL relative to the page location, so it will be reached at the <http://mycompany.com/Images/logo.png> address. Should this page use the `HowTo/Login.mp4` resource in a `<video>` tag, this resource would be loaded from the <http://mycompany.com/Admin/HowTo/Login.mp4> URL.

 **NOTE:** Folders mirrored by a URL do not need to specify physical folders, they can be virtual ones as well. For example, the <http://mycompany.com/Admin> address does not mean that there must be an Admin folder physically on the webserver serving mycompany.com, it just means that the web server understands that there are resources in the server that can be addressed with the `/Admin` route.

The concept of absolute and relative URLs is pretty easy to understand, especially if you try using them. In the source code download of this chapter, you find a sample in the Exercise-05-06 folder. When you load it into the code editor (Visual Studio Code), it displays the folder structure, as shown in the image below:

 Please note that the images have been kept on the back end to allow optimization of editor on our platform.



The structure of the sample project

The `index.html` page is directly within the root of this website, its markup is

shown in Listing 5-6.

Listing 5-6: Exercise-05-06/index.html

```
ï»¿<!DOCTYPE html>
<html>
<head>
  <title>Region1/Page1</title>
</head>
<body>
  <h1>This is Page #1 of Region #1.</h1>
  <a href="/index.html">
    
    Back to the index page
  </a>
</body>
</html>
```

The first three links in the page specify relative URLs, while the last two define absolute ones.

The first relative resource identifier starts with a slash (/), and it says that this resource should be accessed from the root, under the **Regions** folder, within the **Region1** subfolder, and that the resource is named **Region1Page1.html**.

The second URL starts with the Regions folder name, and because the current page (**index.html**) is in the root, it will be resolved to the RegionIndex.html file in the Regions folder. Because index.html is in the webroot, you could have started this resource with a slash too, and it would still have resolved to the very same page.

The third URL is a simple page name, so it will be served from the same folder as **index.html**, which happens to be the root folder.

Remember that the slash character at the beginning of the first relative URL is very important. It provides that moving the **index.html** to another folder will still find the **Region1Page1.html** resource because it is specified relatively from the root of the site.

The second relative link would not survive the movement of the page, since it accesses the RegionIndex.html file relatively from the current page. So, moving index.html and leaving RegionIndex.html in its place would definitely break the link.

The **Region1Page1.html** file contains a link pointing back to the index page, as shown in Listing 5-7. It displays the left.png image that can be found in the Images folder.

Listing 5-7: Exercise-05-06/Region1Page1.html

#

```
<!DOCTYPE html>
<html>
<head>
  <title>Region1/Page1</title>
</head>
<body>
  <h1>This is Page #1 of Region #1.</h1>
  <a href="/index.html">
    
    Back to the index page
  </a>
</body>
</html>
```

As the markup shows, the `"../../Images/left.png"` relative URL is specified to reach the image file. The `"../"` means a step up in the folder hierarchy, so the `"../../"` path means to step up to the root of the site. This image can be accessed with the `"/Images/left.png"` relative URL, too.

When you refer to a page, the browser loads it and displays the page from its top. You can use fragment identifiers to instruct the browser to scroll down to a specified fragment of the page when displaying it. Listing 5-8 shows you the **PageWithFragment.html** that demonstrates this feature.

Listing 5-8: Exercise-05-06/PageWithFragment.html

```
i»<!DOCTYPE html>
<html>
<head>
  <title>Region1/Page1</title>
</head>
<body>
  <h1>This is Page #1 of Region #1.</h1>
  <a href="/index.html">
    
    Back to the index page
  </a>
```

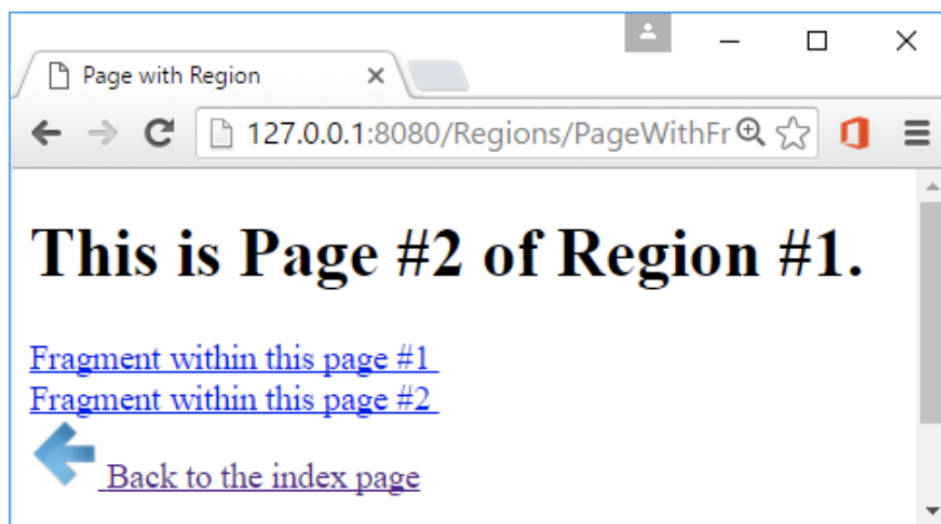
```
</a>  
</body>  
</html>
```

There's a `<p>` tag with `"date"` identifier, and the first two links point to this tag with the `#date` fragment ID.

The very first link uses a relative URL that addresses the page itself, and the `#date` fragment within.

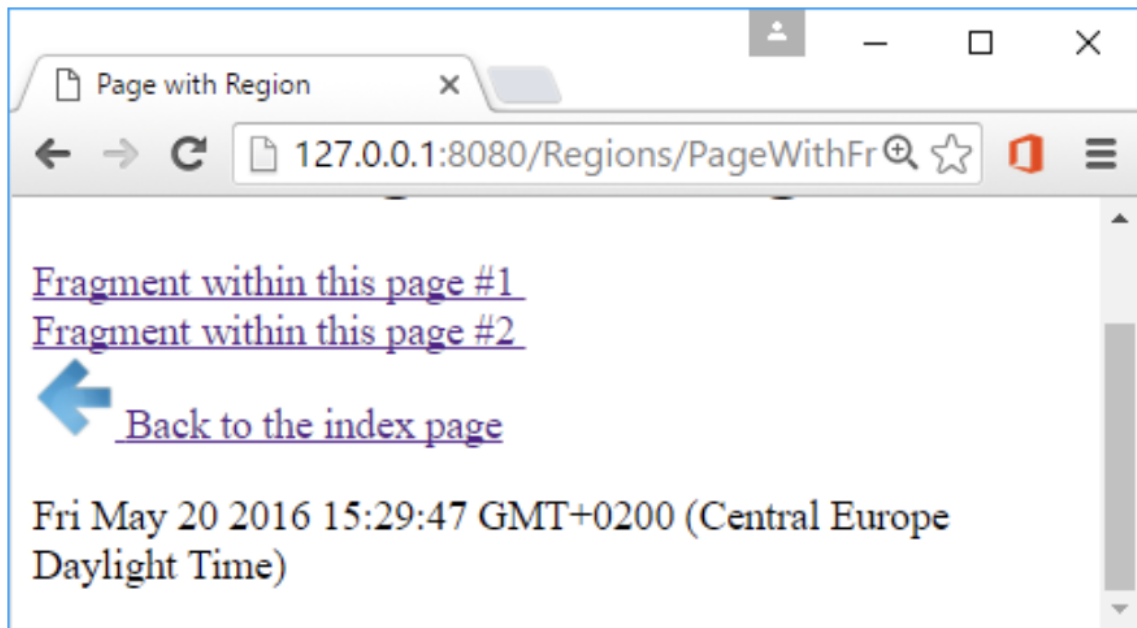
The second link uses only the `"#date"` URL, and that points to the very same fragment. The page contains a script that writes the current date into the `<p>` tag while the page is being loaded.

The image below shows the page displayed in a short browser window. The date information cannot be seen because that overflows the bottom of the window.



The date information cannot be seen

When you click any of the first two links, the browser navigates to the date information. The image below shows this situation.



The browser displays the date information

This sample also demonstrates that the browser does not load the page from the web server again when you click any of the links pointing to the date fragment. The date information displayed remains exactly the same after clicking the links as at the initial load.

However, refreshing the page with **F5** changes the date information, and indicates that the page has been reloaded from the server.

There is an HTML element, **<base>**, which specifies the default URL and target for all relative URLs in a page. Listing 5-9 shows an example of using this tag.

Listing 5-9: Exercise-05-07/index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Using &lt;base> sample</title>
  <base href="http://www.w3.org/" target="_blank" />
</head>
<body>
  <a href="standards/webdesign/htmlcss">
    Click to access the HTML & CSS Standard pages
  </a>
  <br />
  <a href="html/wg">
    Click to access the HTML Working Group page
  </a>
```

```
</body>  
</html>
```

The `<base>` tag must be in the `<head>` section and may occur up to once. Its `href` attribute defines the default URL, `target` specifies the default target for all hyperlinks and forms in the page. When you run Listing 5-9, and click the links, the corresponding pages of the www.w3.org web site appear in a new browser tab.

In the *next lesson*, we will deal with inline frames.

See you there!