

IS 360 Lab 1

Introduction to HTML 5

Ekedahl

Contents

Abstract.....	1
Creating a Web Site and HTML 5 Document in Visual Studio:.....	2
Running the Web Page.....	8
Headings and Content.....	9
Basic Formatting	11
Debugging the Web Page.....	11
The W3C Validation Service	13
Hyperlinks	15
Absolute Links	16
Relative Links	16
Bookmarks	17
Named and Numbered Entities	18
Structural Semantic Tags	19
URLs and their Relationship to Web Site Pages.....	21
Posting the Site	21
Configuring FileZilla.....	22

Abstract

In this lab, you will apply the following skills:

- Create an HTML 5 document in Visual Studio
- Create various tags, attributes, and content in that document

- Create links between documents and to the Web
- View Web pages in a Web browser
- Use the W3C Validation Service to verify the correctness of the HTML
- Upload a simple site to the College of Business Student Web server and test it

You might not complete this lab in the class time allotted. You can certainly finish the lab outside of our class time.

Use the following table as a simple tag reference:

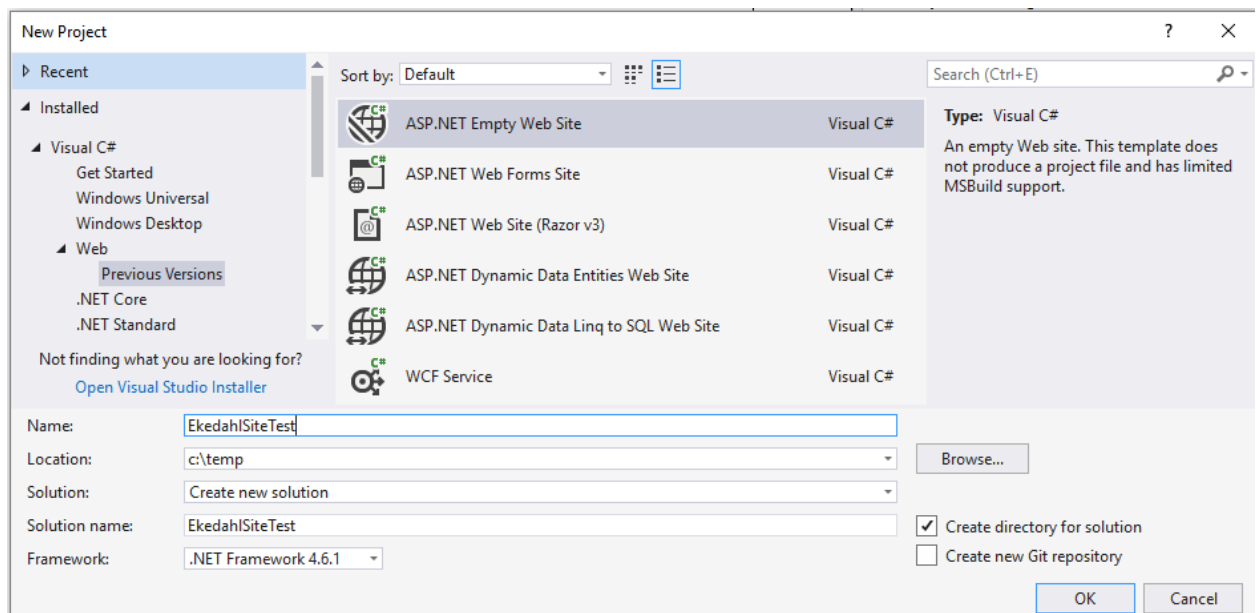
Tag name	Description
<title>	The document title that appears in the browser's title bar.
<a>	Anchor (link) <code>href</code> contains the internal / external link or bookmark
	Bold (older non-semantic tag)

	Line break
<article>	Semantic article tag
<section>	Semantic section tag
	Emphasis (semantic)
<code>	Use a monospace courier (code) font.
<h1>, <h2>, ...	Document headings
<hr />	Horizontal rule (line)
<i>	Italics
	Image <code>src</code> contains the internal / external link to the image <code>alt</code> contains the alternate text that will appear if the image cannot be displayed (also used to improve accessibility).
<pre>	Preformatted text
	Semantic emphasis tag

Creating a Web Site and HTML 5 Document in Visual Studio:

Hands-on Steps: Create a local Web site and add a Web page to it

1. Start Visual Studio. I'm using Visual Studio 2017. *I strongly encourage you to use Visual Studio 2017 as you complete these labs. Visual Studio 2019 is the current version released through the Imagine license. That version will also work fine. If you choose to use the Community Edition, some of the steps will operate differently.*
2. Create a new Visual Studio Web site by clicking **File, New, Project**. The New Project dialog box appears. As you saw in IS 350, the tab on left shows the languages and project types, and the region on the left shows the detailed project kinds. *Again, depending on the .NET Edition and version you are running the installed templates and project types will differ.*



- Expand the folders as shown in the above figure (**Visual C# / Web / Previous Versions**). Select an **ASP.NET Empty Web Site**. Give the project a name of your choosing. **Do not use spaces in folder or file names**. Make Sure **Create new solution** is selected. Make sure the check box titled **Create directory for solution** is selected.

I strongly suggest that you create the project on a USB drive or in your StudentHome directory (U: drive). To do so, click the **Browse** button in the **Location** drop-down, and select the folder where you want the Web site to be created.

Make sure that you create an empty Web site. If you create the other Web site types, a vast amount of code will be added to the site that is beyond the scope of the class.

Do not put spaces in the folder name. It will cause problems later! Do not put spaces in file names either. They make for very ugly URLs.

It does not matter whether you specify a Visual C# or Visual Basic template as we will not be creating any server code. Click **OK** to create the project.

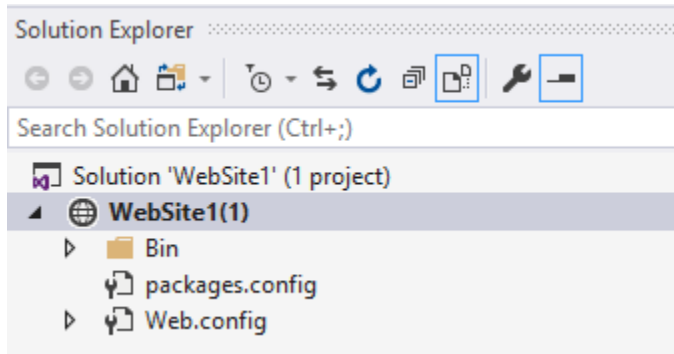
- The Solution Explorer will appear and displays the files that make up the solution. At this point, there are two files in the folder.

The file `web.config` is the application configuration file.

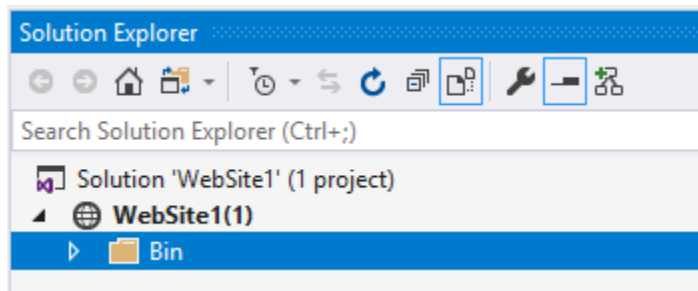
The `packages.config` file tells .NET, which version of the compiler to use and the target version of the .NET Framework. These files are used by ASP.NET and are needed by simple HTML

5 pages. For now, you can ignore them.

When you upload your Web site to the Web server, DO NOT upload the Web.config or the packages.config files. If you do, the Web server will consider this to be an ASP.NET application. However, the Web server is not configured to run your ASP.NET application. The page will not load and your application will not run. Instead, you will receive a server error.

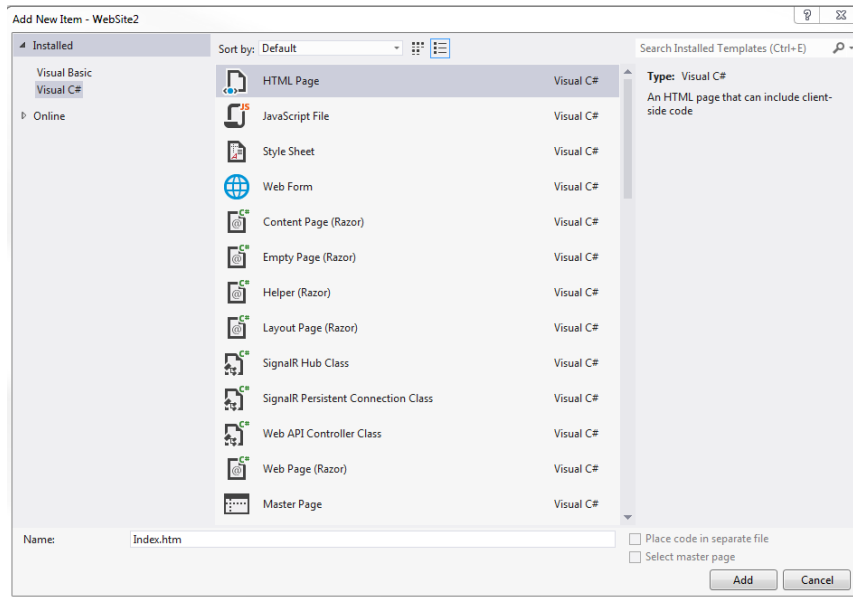


5. In the Solution Explorer, right-click on the file names and delete them. Again, these files are used by Microsoft ASP. We will not be creating an ASP application for some time. When complete, the Solution Explorer should be empty, with the exception of the Bin folder:



You could just as well get rid of the Bin folder.

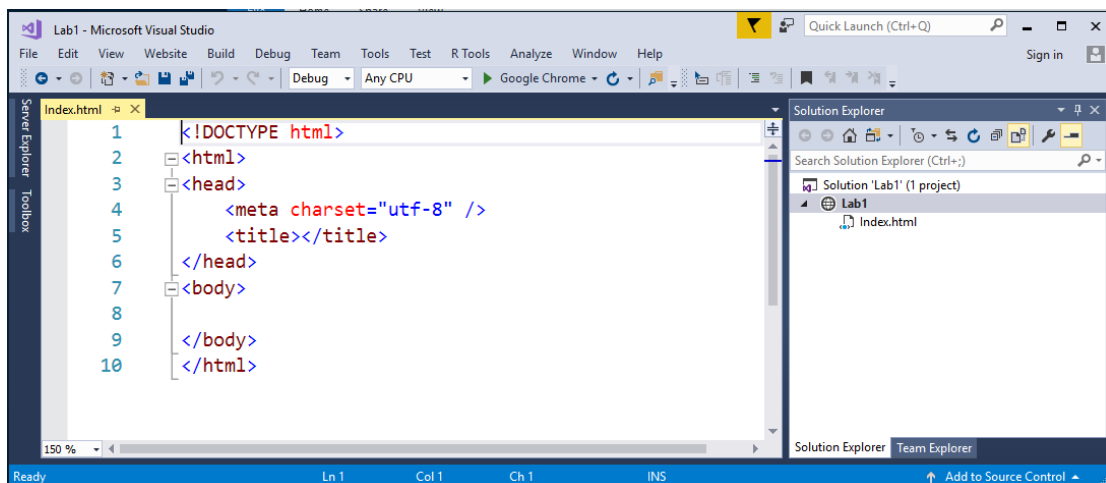
6. Next, add an HTML 5 file (page) to the project by clicking **Website, Add New Item**. As you can see, there are several file types. The following dialog box will appear:



7. Select **HTML Page**. Name the page **index.htm** or **index.html**.

*Web pages (HTML documents) typically have a file extension of .htm or .html. Furthermore, index.htm or index.html typically designates a page as a default page for Web servers. **Again, don't create Web pages having spaces in their file names.***

8. In the dialog box, click **Add** to create the page. It appears in the Visual Studio designer window as shown in the following figure:



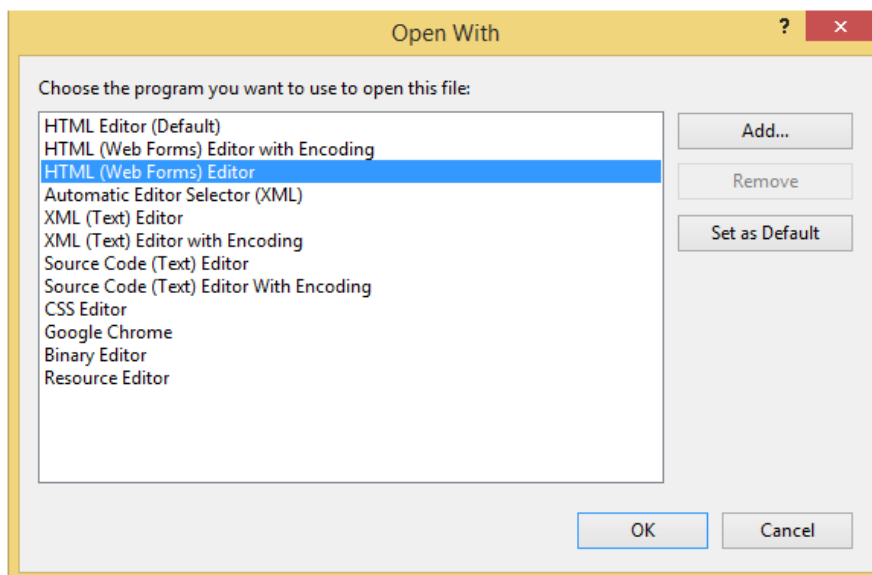
Remember that the page named index.html is magical. When a file named Index.htm or Index.html appears in a folder, the Web server will automatically render a file of this name when no file name is specified in the URL. On Windows servers, the case of the file name is not significant.

The HTML 5 file is created and is made the active file in the Code Editor. The file also appears in the Solution Explorer. You can see that the file named index.htm(l) is created. Visual Studio, using a template, copied some default text into the document for you.

Visual Studio 2013, introduced a new default HTML Editor, which replaced the ASP.NET Web Forms editor. However, the new editor does not support a previewer. In the following steps, you will associate Visual Studio with the older Web Forms Editor. You may or may not have to perform this step in the College of Business labs.

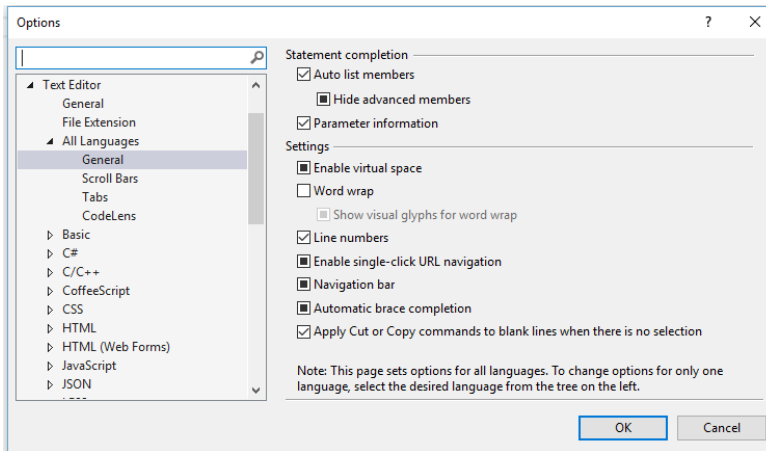
You only need to perform the following steps if you want to see the output preview.

1. In the Solution Explorer, right-click the index.htm file that you just created, and click **Open With** in the context menu. In the dialog that follows, click **HTML (Web Forms) Editor**, and then click **OK**. *If the document is open, a warning message will appear asking you to close it.*

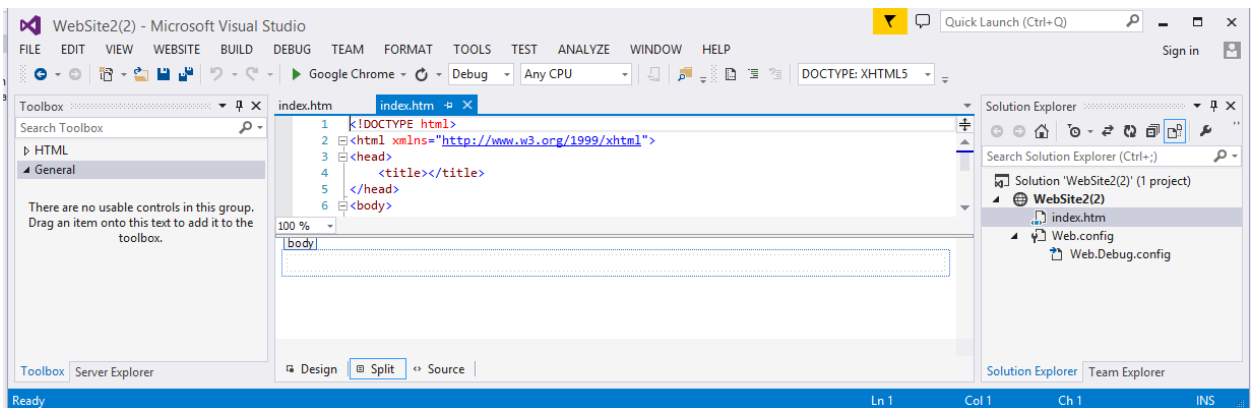


Note the tabs along the bottom of the screen. The Design tab provides a WYSIWYG (What you see is what you get) interface to develop the page. The Split tab splits the window so that you can see both the HTML 5 code and the WYSIWYG representation. The final tab, Source, shows only the source (HTML 5) code. The following figure shows the window Code Editor in Split mode.

Depending on the configuration, line numbers may or may not appear. If you want line numbers, click **Tools, Options**. Expand the **Text Editor** folder. Expand the **All Languages** folder. Select the **General** node, and then check the **Line Numbers** box.



- The following figure shows roughly what the development environment should look like.



Visual Studio used a template to create this file. The DTD reference was automatically inserted to the HTML 5 DTD. Default `<head>`, `<title>`, and `<body>` tags were also added. Visual Studio uses templates for many different types of files.

- Click the **Split** tab so that you can see both the HTML 5 and rendered markup. The Design tab will be blank because you have not yet created any content.
- Change the document title to **My First Web Page**. This title will appear in the browser's title bar and selected tab. That is, edit the text for the `<title>` tag.

```
<head>
  <meta charset="utf-8" />
  <title>My First Web Page</title>
</head>
```

Note that if you edit the content in the Source view, you will need to save the file for the changes

to be synchronized in the Design view. In addition, note that the Visual Studio previewer is not perfect. It will not correctly render complex documents.

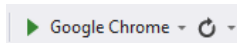
Running the Web Page

Visual Studio allows you to test your Web pages from inside of the development environment. When a page is selected and run, it is rendered in the default browser that you have selected. Visual Studio will allow you to select from any browsers installed on your system.

For reasons that will become clear when you begin to develop JavaScript code, I **strongly suggest** that you use Chrome as the default browser. It has the best debugging tools and supports great CSS development and debugging tools.

There are a couple of ways to run the program, thereby displaying the Web page in a browser. Select the page that you want to view in the Solution Explorer, and then click the **Run** button. You can also press **F5**.

Make sure to select the page to run in the Solution Explorer.NET does not inherently know which page to run. Also, the following toolbar item allows you to select the default browser:

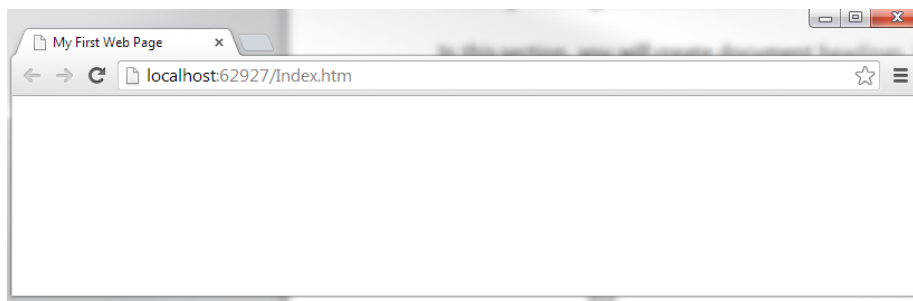


Hands-on Steps: Running a Web page

1. In the Solution Explorer, select the file named **Index.htm**. Press **F5** or click the **Run** button. The page appears in the selected Web browser (in this case Chrome).

You can also right-click on the page. From the context menu select **View in Browser**.


Most browsers cache pages. That is, if you load the page over and over again, the page might not be downloaded from the Web server. If you don't see changes you have made to a page, right-click on the page in the browser and select **Reload**.



If you look at the above URL, localhost indicates that the page is being run from local file. The

value 62927 is the temporary port used to communicate with the local Web server. Your port number will likely differ.

Visual Studio has a small built-in Web server which it uses to send your page(s) to a browser.

2. There is no content in the page because you have not created any content yet. Close the Browser.
3. Press the **End**  button to stop running the program. Visual Studio returns to design mode.
4. Close the browser tab.

Headings and Content

In this section, you will create some textual elements.

- Document headings: Document headings are marked with the `<h1>`, `<h2>`, `<h3>` ... tags.
- Paragraphs: Paragraphs are marked with the `<p>` tags.
- Horizontal rule: A horizontal rule is a line drawn across the page. The tag contains no content so it's written with the shorthand notation `
`.

Hands-on Steps: Creating Headings and Basic Content:

1. In the `<body>` section of the Web page that you just created, create an `<h1>` tag so that the text **My First HTML5 Document** appears as the heading text. You can create the HTML in two ways.
 - **Using Source View:** as you create this first tag, pay attention to the Intellisense technology. The ending tag is created automatically. Possible tags appear in the popup list.
 - **Using Design View:** the Formatting ToolBar will create headings and other formatting elements. To use the formatting toolbar, select the text to be formatted and click the desired formatting selections.



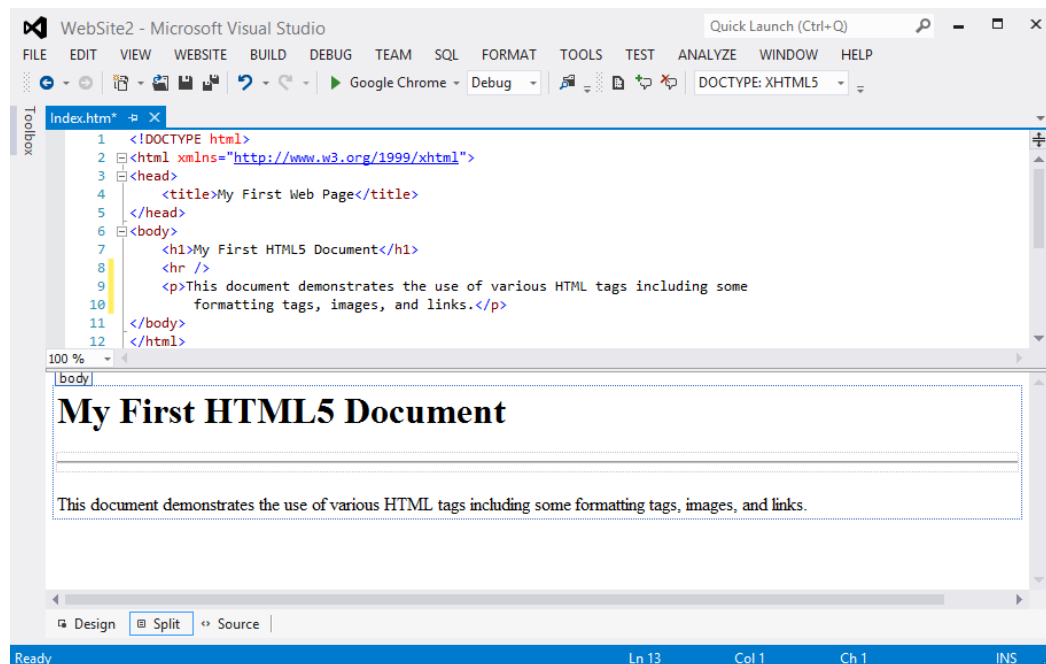
2. Create a horizontal rule (line) just below the `<h1>` tag that you just created. Use the `<hr />` tag. Remember that this is the shorthand notation to combine both a starting and ending tag.

3. Insert a paragraph `<p>` tag after the `<h1>` tag and `<hr />` tags. Remember that a paragraph tag requires both a starting and ending tag. Then enter the following content:

This document demonstrates the use of various HTML tags including some formatting tags, images, and links.

Remember from the lecture that HTML applies whitespace normalization. Thus, you can enter the above paragraph on one line or several lines. The same output will be rendered.

At this point, your document should resemble the following figure. Remember that it does not matter where you put the carriage returns as HTML performs whitespace normalization (removes extra tabs, spaces, carriage returns, etc...). *It does not matter that you're your tags exactly match mine. Feel free to be creative and explore. Just make sure that your HTML is syntactically correct!*



4. Right-click somewhere in the region of the HTML 5 document and click **View in Browser** or press the **Run** button. The document should appear in the default browser (typically IE or Chrome in the College of Business labs).

Basic Formatting

You will learn much more about formatting and elements over the next couple of weeks. Here, you should focus on the syntax of elements and attributes.

The formatting being applied to the following tags happens because browsers apply a default Cascading Style Sheet. When you learn about CSS in the next section of the course, you will see how to format these tags any way that you want.

- The `` tag makes text appear bold.
 - The `` tag makes text appear in italics.
 - The `<code>` tag make text appear in a mono-space courier font.
1. Create a level 1 heading `<h1>` at the end of the document. The text for the heading should be **Formatting**.
 2. Next, create four paragraphs so that they appear formatted as follows: If you cannot remember the tag names, then refer to the table at the beginning of this document. You might also want to look at <http://www.w3schools.com/html/default.asp>

This text is **bold**.

This text is *italics*.

This text appears in a **monospace courier** font.

This text is both ***bold and italic***.

Note that the second to the last paragraph uses a monospace courier font appearing in a bold typeface. Thus, you need to nest the `` and `<code>` tags. *When you look at the text in the browser, the monospace courier text may not appear bolded. It probably is bold. However, against the default font, it does not appear that way.*

3. Now try to create some preformatted text. At the end of the document, create a `<pre>` tag and some content between it. The content should appear on multiple lines with several embedded spaces. When you view the content in the browser, the whitespace is not normalized.

Debugging the Web Page

As you program, you will likely make mistakes. The Visual Studio Code Editor validates the code you write, be it C#, JavaScript or HTML. When errors are discovered, they are listed in another window called the Error List window. Visual Studio also highlights syntax errors in the Code Editor as you make them.

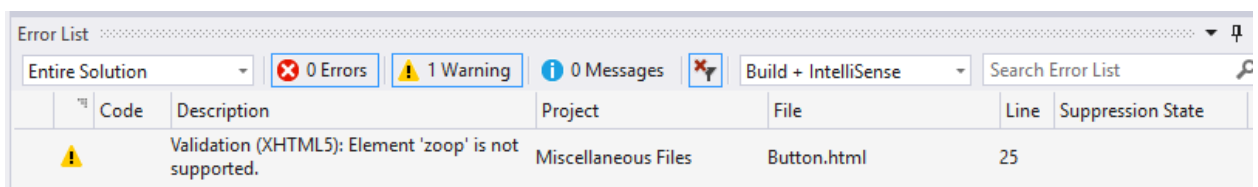
Hands-on Steps: To create and locate syntax errors:

1. Using the same HTML file that you have been using, enter an invalid tag within the `<body>` element. I used the following incorrect tag named zoop `<zoop></zoop>`. There is no element named `<zoop>`.

`<zoop></zoop>`

You should see a jagged green line indicating a warning error. Red lines indicate more catastrophic errors.

2. Open the Error List window, if necessary, by clicking **View, Error List**. The following figure shows the Error List window.

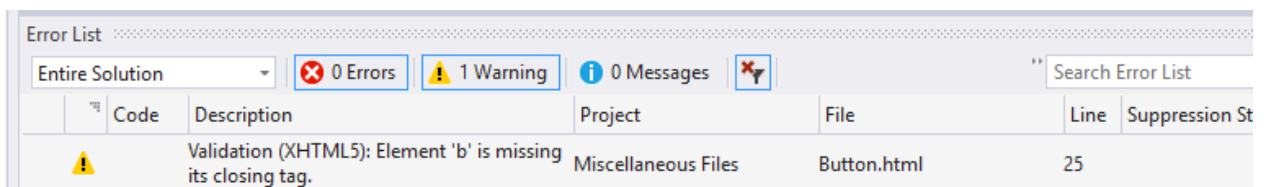


One error is displayed per line. Each line contains the error description and the line where the error was found.

3. Now, remove the “offending” text, and you should see the error removed from the Error List window.

Intellisense is constantly scanning the code that you write. And errors might not be detected immediately depending on how fast you are typing. Try making another mistake and correct it. Try putting a paragraph `<p>` tag in the header. You will get an error because `<p>` tags must appear in the document body.

4. Create an opening tag without a closing tag. You should see the following message in the Error List window. I used a tag named `` without a closing tag.



The W3C Validation Service


The World-Wide-Web Consortium (W3C) provides a validation service that will check your HTML for errors. It basically does the same thing as the Intellisense that you just used.

While you will not use them in this lab, the W3C provides other validators. There are validators to check CSS, validate that hyperlinks work, and mobile accessibility.

Hands-on Step: In the following exercise, you will use the W3C Validation Service

1. Make sure that you have saved the Web page that you created in the previous exercise.
2. In the browser of your choosing, visit the W3C Markup Validation Service at <http://validator.w3.org>.
3. Click the **Validate by File Upload** tab so that you can upload the Web page.
4. Click the **Choose File** button to select the file to upload. In the explorer dialog that appears, select the HTML file to validate.
5. Click the **More Options** button to select the DTD. The selections you make should be similar to those in the following figure:

By default, the markup service will determine the character encoding and document type from the information in the document itself. Therefore, you generally do not need to set these values.



Markup Validation Service

Check the markup (HTML, XHTML, ...) of Web documents

Validate by URI
Validate by File Upload
Validate by Direct Input

Validate by File Upload

Upload a document for validation:

File: IS360FirstPage.htm

▼ More Options

Character Encoding
utf-8 (Unicode, worldwide)
☐ Only if missing

Document Type
HTML5 (experimental)
☐ Only if missing

☒ List Messages Sequentially
☐ Group Error Messages by Type

☐ Show Source
☐ Clean up Markup with HTML-Tidy

☐ Show Outline
☐ Validate error pages
☐ Verbose Output

Check

Note: file upload may not work with Internet Explorer on some versions of Windows XP Service Pack 2, see our [information page](#) on the W3C QA Website.

- Click **Check** to run the validator. The errors and warnings appear at the bottom of the form. The following two errors are acceptable although you should not see them.

Here is a second screen shot with different errors from the validator.

Nu Html Checker

This tool is an ongoing experiment in better HTML checking, and its behavior remains subject to change

Showing results for uploaded file Button.html

Checker Input

Show
☐ source
☐ outline
☐ image report

Check by
file upload
No file chosen

Uploaded files with .xhtml or .xht extensions are parsed using the XML parser.

Use the Message Filtering button below to hide/show particular messages, and to see total counts of errors and warnings.

Message Filtering

1. **Error** Element `title` must not be empty.
From line 4, column 12; to line 4, column 19

```
<title></title>
```

2. **Error** Element `z` not allowed as child of element `body` in this context. (Suppressing further errors from this subtree.)
From line 25, column 5; to line 25, column 7

```
span><z></z>as
```

Content model for element `body`:
[Flow content](#)

The HTML specification requires that the `<title>` tag have content.

The second error is one that I created. I put in a `<z>` tag which is not a validate tag.

7. Review the errors.

Here are a few general debugging tips:

- Debug as you develop. When Intellisense flags an error, read the message and correct it immediately. Syntax errors “cascade”. That is, one error leads to another, which leads to another, and so on.
- When you see multiple syntax errors, try to correct the first error first. As syntax errors cascade, correcting the first error will often solve other ones.
- The Error List window and the W3 Validator both supply error descriptions and line / column numbers. Pay attention to this information. It often tells you what you are looking for and where to find it.

Hyperlinks

Hyperlinks are used to navigate to different pages or to navigate to specific location on the same page. Hyperlinks can be categorized as follows:

- **Absolute links** contain complete URL references to sites on the Web.
 - **Relative links** contains a relative to a URL within your Web site.
 - **Bookmarks** are links to content on the same page as the hyperlink.
- The following tag shows a first hyperlink:

Here is a first hyperlink

```
<a href="http://www.unr.edu">University of Nevada, Reno</a>
```

- The `<a>` (anchor) tag creates a hyperlink.
- The `href` attribute contains the **link target**. The above link contains an absolute reference to <http://www.unr.edu>.
- The element content (University of Nevada, Reno) contains the text seen by the user. When the link is clicked, the browser visits the link contained in the `href` attribute. As always, the

element content appears between the opening and closing tags.

Absolute Links

Absolute links are usually links to another Web site. When creating absolute links, make sure to include the protocol part of the URL (`http://`) or the browser will likely not resolve the link and a 404 (not found) error will occur.

Hands-on Step: Creating absolute links

1. At the bottom of the page, create another `<h1>` tag titled **Links**.
2. After that tag, create the above link to unr.edu.
3. Create a second absolute link of your choosing.
4. Put a `
` tag between the links so that each appears on its own line.
5. Save and run the page to test that the links are working correctly.

Relative Links

Relative links are used to link to pages within your Web site. Absolute links can always be used in lieu of relative links but relative links tend to be shorter. There are two “special” character patterns used with relative links.

- A dot (.) means the current directory (folder).
- Two dots (..) means the parent directory (folder).
- The following `<a>` tag says to visit the page named **Page2.html** in the same directory as the current page.

```
<a href="./Page2.html">Page 2</a>
```

- The following `<a>` tag says to visit the page named **PageUp.html** in the parent directory.

```
<a href="../PageUp.html">Page Up</a>
```


- The following `<a>` tag says to visit the page named `Other.html`. The `..` says to move up to the parent directory and the `/Misc` says to move down to the **Misc** directory.

```
<a href="../../Misc/Other.html">Other Page</a>
```

In this lab, all of the files for this simple Web site appear in the same directory. However, most Web sites have a complex directory structure. Pages are grouped together in different directories based on their purpose.

Hands-on Steps: Creating multiple pages and links between them

1. Add a second Web page to your project. To do this, click **File, New, File** on the menu. The Add New Item dialog will appear as before. Select an **HTML page**, and name the page **CopyrightPage.html**.
2. On the **Copyright.html** page, create a relative link back to the home page. Make sure to create this tag in the `<body>` section.

```
<a href="../Index.html">Home</a>
```

3. On the main page, create a relative link to the `Copyright.html` page. Create the link with the links you already created.
4. Again, test the pages in the browser to make sure that the links work.

Bookmarks

A bookmark is designed to navigate a particular place in a document. There are two steps to creating a bookmark:

- First, you use the `id` attribute and a unique name to create the bookmark.
- Then you use the `<a>` tag to create a link to that bookmark. The value of the `href` attribute contains the value of the bookmark's `id` attribute preceded by a pound sign.

```
<h1 id="MyBookmark">
<a href="#MyBookmark">Jump to MyBookmark</a>
```

In the above, the `id` attribute is set to **MyBookmark**. The `href` attribute is set to **#MyBookmark** (the name of the bookmark preceded by a pound sign).

If the bookmark name and value of the `id` attributes don't match, the bookmark will not work. Also, the error will fail quietly. You will not see an error message. The browser will just not do anything. Furthermore, `id`'s must be unique on a page. Just like variables in other programming languages, they cannot have duplicate names. Finally, `id` values ARE CASE SENSITIVE.

Hands-on Steps: Creating bookmarks

1. In the index page, create a unique `id` value for each of the `<h1>` tags.
2. Now at the top of the page, create anchor tag that will jump to each of the bookmarks that you just created.
3. Again, test your changes in the browser to make sure that they work.

Named and Numbered Entities

Because HTML interprets characters like the less-than and greater-than signs as tag markers, we need a way display those characters as “literal characters”. We have the same problem with other special characters like quotation marks and spaces. Recall the HTML is whitespace normalized – multiple spaces and tabs are converted to a single space. So how do you preserve multiple spaces. The answer is named and numbered entities.

Named entities begin with an ampersand (&), followed by the entity name, followed by a semi-colon (;). As in `<` or `>`;

Numbered entities work similarly. Except instead of a name, a hexadecimal or decimal value appears as in

`¼` will produce the fraction $\frac{1}{4}$.

If the value is to be interpreted as hexadecimal, then the hash tag (#) precedes the numeric value.

Here is a list of often used named entities;

- `>` - greater than
- `<` - less than
- ` ` – non-breaking space
- `©` – copyright sign
- `"` - double quotation mark
- `'` - apostrophe
- `&` - ampersand

Here is a canonical list: <https://dev.w3.org/html5/html-author/charref>

Hands-on Steps: In the following exercise, you will work named entities and text

1. Create a heading at the end of the document. The content for the heading should be **Named and Numbered Entities**.
2. Insert a copyright symbol in another paragraph using the following named entity: (`©`) so that it appears as:

© Michael V. Ekedahl

3. Create the following paragraph after the one just you created so as to demonstrate the use of named entities:

```
<p>The <code><code>&lt;br /&gt;</code> tag is used to insert a horizontal rule</p>
```

4. View the page in the browser. You should see that the paragraph rendered as

The `
` tag is used to insert a horizontal rule.

5. Now, try it on your own. Write the HTML 5 to render the following text. Preserve the extra spaces.

The ampersand (&) character precedes the less-than (<) sign when referencing a **named entity**.

6. The following link lists all of named and numbered entities in HTML5. As you can see, there are a lot of them. <http://www.freeformatter.com/html-entities.html#iso88591-symbols>
7. Try and use the numbered entities to produce the Greek symbols Sigma and Omega.
8. Just for fun, here is a complete list of emojis. <https://unicode.org/emoji/charts/full-emoji-list.html>. Try and display a few of them in the browser.

Structural Semantic Tags

Many new semantic tags were introduced with HTML 5. These semantic enhancements do not generally

change the formatting of a document. Rather, they are designed to add context that can be recognized by search engines and other tools.

- An `<article>` is designed to depict self-contained content. An article should make sense on its own. Articles include such content as a news story, or a blog or discussion forum post.
- A `<section>` defines sections in a document or article.
- An `<address>` depicts contact information for the content creator or owner. If it appears inside an `<article>` then the contact information should apply to the article. If it appears inside the `<body>`, it defines the contact information for the entire document.
- The `<header>` tag designates a header for a page or section. A document can contain several `<header>` tags. A `<header>` tag cannot appear within a `<footer>`. Don't confuse the semantic `<header>` tag with the `<head>` tag.
- The `<footer>` tag designates a footer for a page or section. The same rules apply to the `<header>` and `<footer>` tags.
- The `<nav>` tag marks a block of "major" navigation links (menu).
- The `<aside>` tag is designed to specify tangential content related to an article or section.
- The `<address>` tag marks contact information about the author of an article or page.
- The `<time>` tag is designed to create a timestamp indicating when a page, article, section, or whatever was created or revised. The `<time>` tag has an attribute named `datetime` that contains the timestamp. The timestamp has a required format of YYYY-MM-DD HH:MM:SS. The element content is rendered to the user.

Exactly when and where to use these semantic tags is subject to some interpretation. Some sites don't use them at all. Search engines are using them more and more to improve search results because of the additional contextual information.

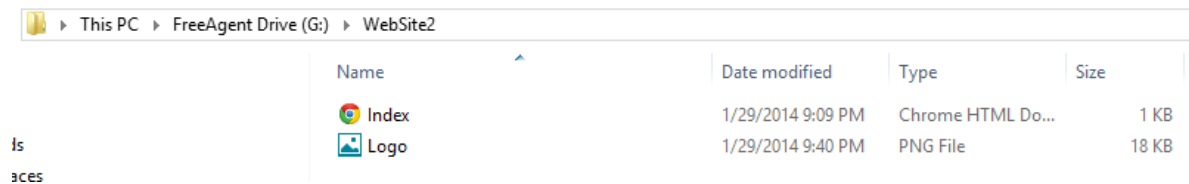
Hands-on Steps: Building semantic tags



1. On the index page, wrap the page body in an article.
2. Wrap each heading `<h1>` tag in a section.

3. On the copyright page, add an address tag, sample contact information and a time stamp.

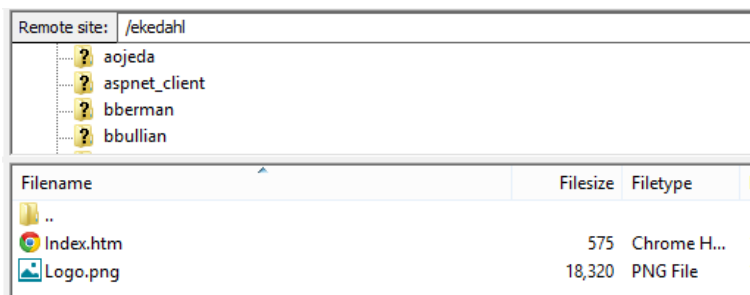
URLs and their Relationship to Web Site Pages








A Web site can be as simple as a single HTML 5 file that resides in a folder on a Web server. The folder and file name correspond to the URL. As you develop a Web site, the folder structure that you implement on the local computer mirrors the structure of the Web site as it will appear on the Web server. For example, suppose that I have a Web site with two files named **Index.html** and **Logo.png** as follows:



This PC > FreeAgent Drive (G:) > WebSite2				
	Name	Date modified	Type	Size
Is aces	 Index	1/29/2014 9:09 PM	Chrome HTML Do...	1 KB
	 Logo	1/29/2014 9:40 PM	PNG File	18 KB

I upload these files to the remote site using FileZilla as follows. In other words, the two files appear in the folder Ekedahl.



Remote site: /ekedahl			
 aojeda  aspnet_client  bberman  bbullian			
Filename	Filesize	Filetype	L
 ..			
 Index.htm	575	Chrome H...	1
 Logo.png	18,320	PNG File	1

You reference a Web page with a URL made up of a domain, folder, and file as discussed in class.

Since the host is **swww.coba.unr.edu**, and the folder is **ekedahl**, the URL would be <http://swww.coba.unr.edu/ekedahl>

The page named index.htm is the default Web page, so it need not be specified explicitly in the URL. However, you could fully qualify the name as in:

<http://swww.coba.unr.edu/ekedahl/index.htm>

Posting the Site

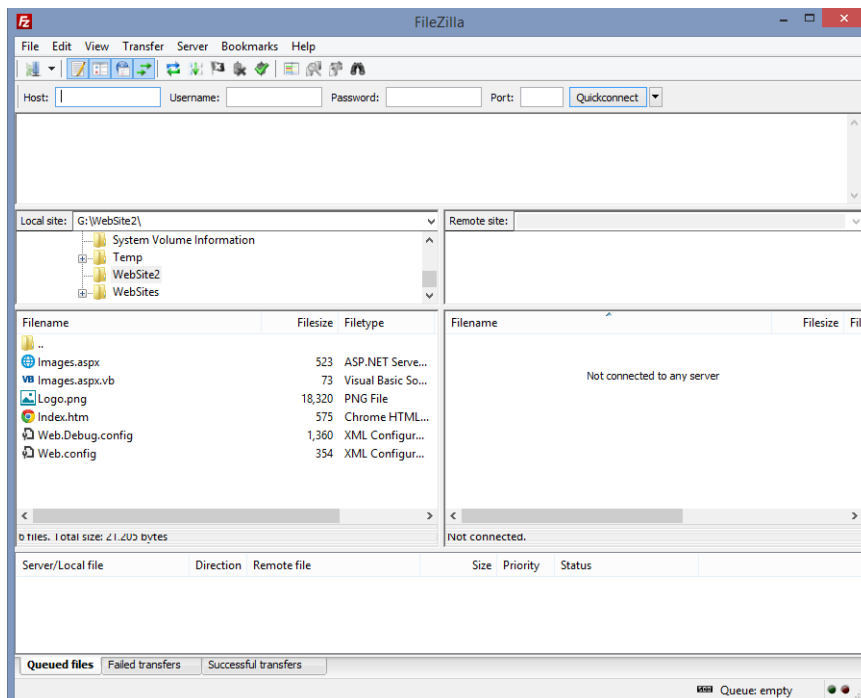
In this section, you will post the Web site that you have created to the Web. In this class, we will use FileZilla. It's free and widely available. You can download at

<http://sourceforge.net>.

<https://sourceforge.net/directory/os:windows/?q=filezilla>

Note that FileZilla is already installed in all of the COBA labs.

1. **Start FileZilla.** Your screen should appear similar to the following one.



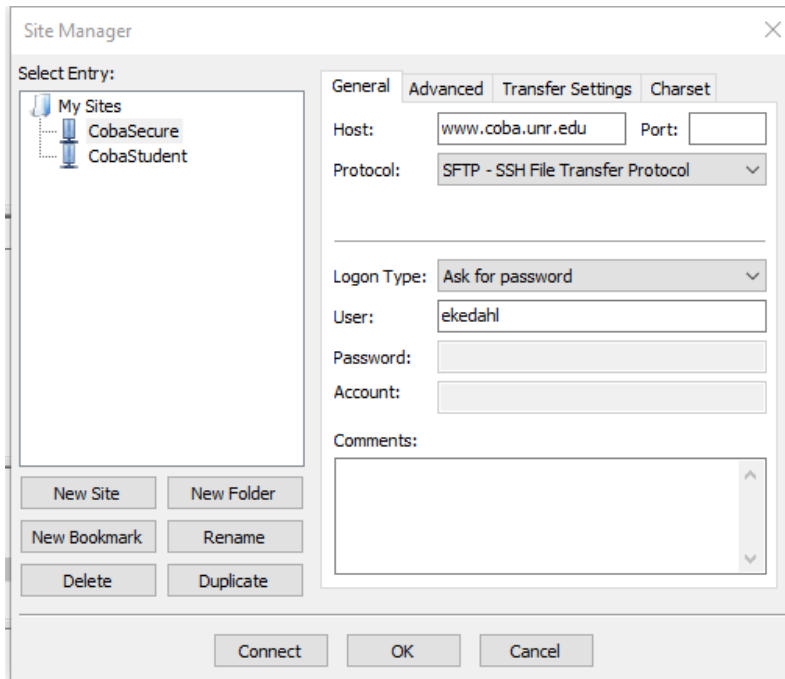
- The FTP requests and responses appear at the top of the screen. There have been no requests or responses so the region is blank.
- On the left side of the screen is the local file system (your computer)
- On the right side, is the remote file system. Nothing appears in the above figure because we are not connected to the remote server.
- Status and log information appears in the lower window.

Configuring FileZilla

Next, you configure the host to which you will upload your web site. The host is named **www.coba.unr.edu**. **students.coba.unr.edu** will also work.

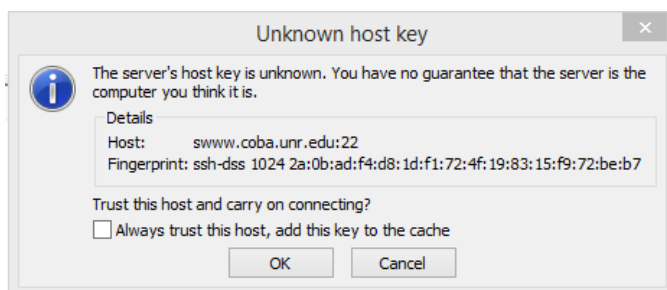
I suggest that you follow my steps to create a new site. Don't just enter the host and credentials into the QuickConnect dialog. We use a secure FTP server and it's a different port number and protocol.

1. Click **File, Site Manager**. The following dialog box appears.



2. Enter the host named **swwww.coba.unr.edu**. Make sure to select the **SFTP – SSH File Transfer Protocol**. We run secure FTP. For the user, use your NetId. Our student Web server authenticates using your NetId. I suggest prompting for passwords! That way, your password will not be stored on a public computer.
3. Click **Connect**. You will be prompted for a password if you did not save one.

Because we don't have a current certificate, the following error message appears.

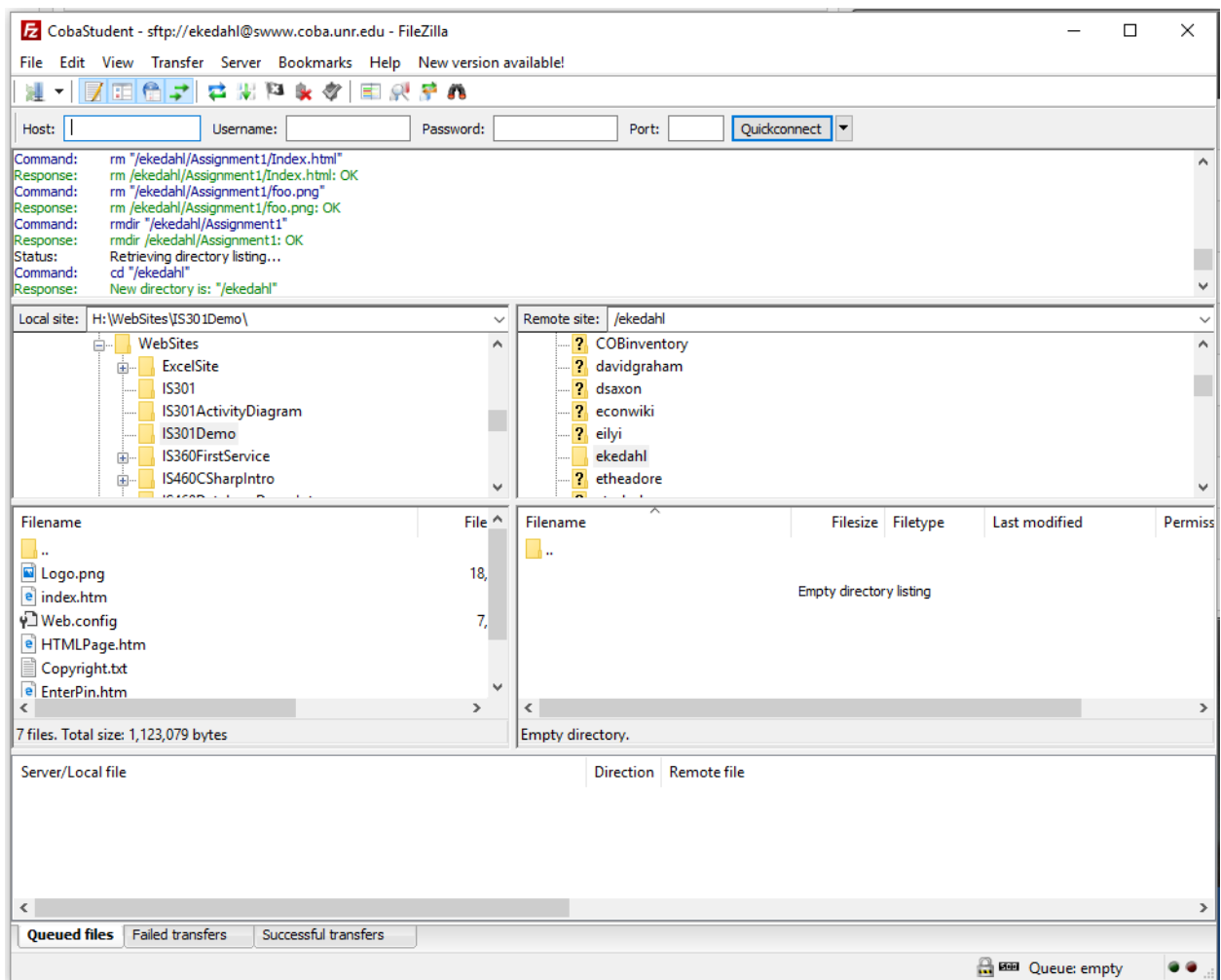


4. Click **OK** to continue. The top screen panel should display the successful connection as follows. In addition, you should see an empty home directory in the right panel (server).

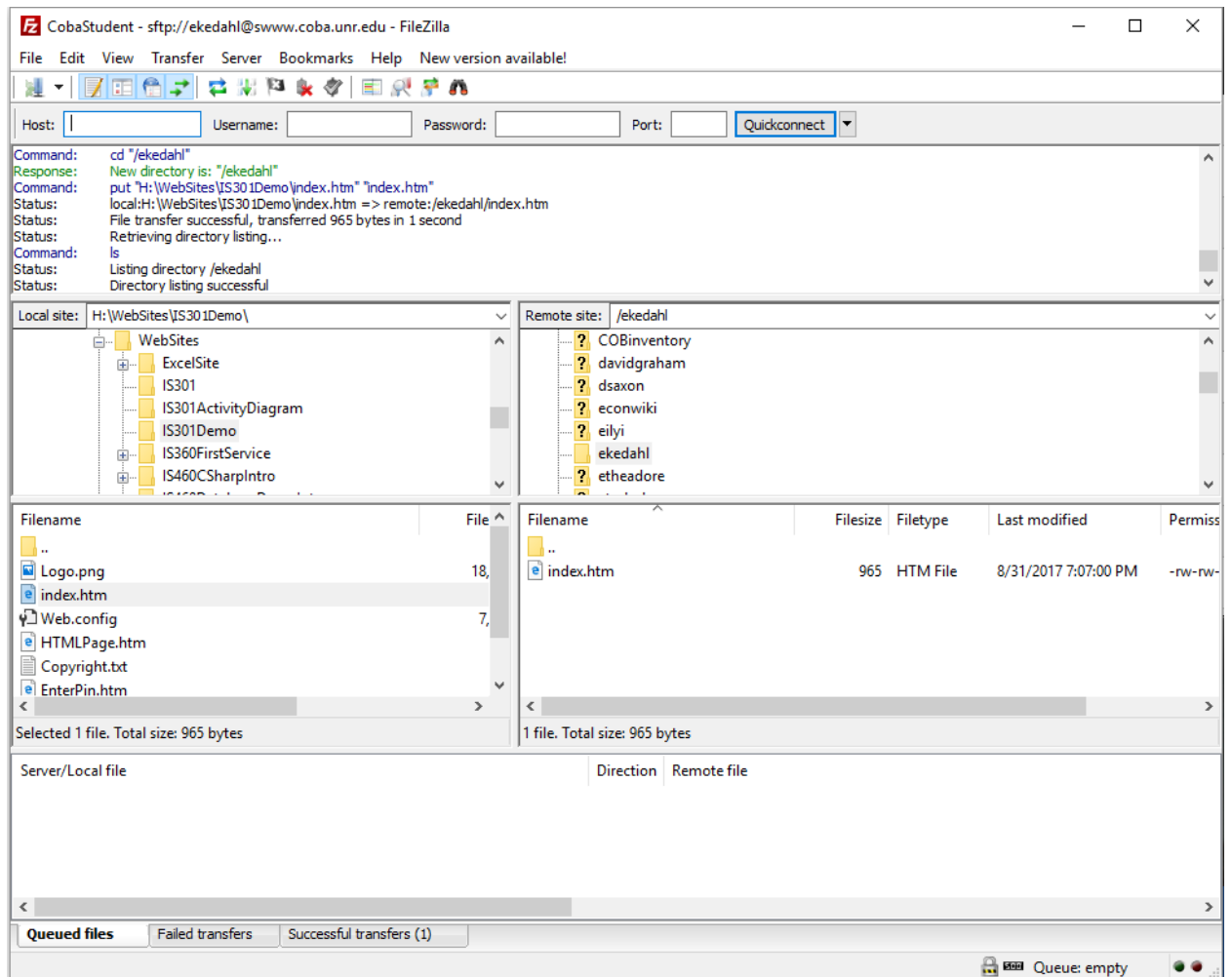
```
Status: Connecting to swww.coba.unr.edu...
Response: fzSftp started
Command: open "ekedahl@swww.coba.unr.edu" 22
```

If you see authentication error messages in red, you did not successfully log in. Try again.

- Next you will upload the Web site. In the rightmost window, select the drive and folder that contains your local Web site.



- Drag each individual file from the left window to the right window. As you do, the command and status windows are updated. In the following figure, you see the file index.htm on the server.



- Finally, try to see if the Web site works. In the browser of your choosing, enter the following URL <http://swww.coba.unr.edu/netid> where *netid* is your actual NetId. If you did everything correctly, your Web page should appear.