Web Module 3:

HTML, Part II

Introduction

In this module, the second of a two-module sequence on HTML (Part I and Part II), we'll advance your HTML coding career by presenting some more HTML concepts and techniques.

When you're done working through this module, you'll know:

- What HTML standards are, and why you should care about them
- How to get your web pages on the Web
- How to create beautiful code
- How to create a multi-page website
- How to validate a web page and an entire website

Caveat Lector

These eight Web modules build on each other. For example, in Module 4 it's assumed you understand the concepts and techniques from Modules 1-3. So, if you run into anything you're unfamiliar with in this module, browse through the earlier modules or turn to Google for help.

HTML Standards

A standard is a specification that is accepted and put into practice by its target audience. An HTML standard is an accepted set of features and rules of the HTML language.

The latest HTML standard is HTML5. That's the standard we're using for these Web modules.

The previous standards were: HTML 4.01 (1999), HTML 3.2 (1997), HTML 2.0 (1995), and HTML 1.1 (1992).

Who decides what features get added to the HTML language and supported by browsers for a new standard:

Browser vendors like Mozilla, webKit, and Chromium?

Or standards bodies like the W3C (World Wide Web Consortium)?

They both do! It's a delicate dance ...

Browser vendors don't want to implement browser features from a new specification before they're sure that this specification will be accepted as a standard.

Standards bodies don't want to finish a specification before browser users have had ample time to try it out, find bugs, register complaints, etc.

Both HTML and browser features tend to evolve rapidly. You can use these sites to keep an eye on what new features are coming down the pike:

Web Standards Bodies

The W3C – w3.org

The Web Hypertext Application Technology Working Group – whatwg.org

Browser Vendors:

Mozilla (Firefox) – mozilla.org/en-US

webKit (Safari) – webkit.org

Chromium (Chrome) – chromium.org

Getting your Pages on the Web

To get your HTML pages on the Web so that they can be viewed by anyone with an Internet connection and a browser:

Upload all of your page files – .html, .jpg, .gif, .mp3, etc. – to a web server.

A web server is a server connected to the Internet and the World Wide Web.

To do this, you need to have an account on the web server.

All RIT students have an account on the RIT web server named Gibson.

The URL of the Gibson server is http://people.rit.edu

To upload files to your web account, you must use an FTP client.

There are many FTP clients out there, both free and commercial. Here are some of the most popular:

Windows – Smart FTP, WS_FTP, Filezilla

Mac – Fetch, PureFTP, Filezilla

Tip: If you code web pages by hand, rather than in an HTML authoring tool with built-in FTP like Dreamweaver, you'll be using your FTP client frequently. So do the research, compare clients, and make an informed choice.

The Client-Server Relationship

What does a web server (like Gibson) do?

It sits around waiting for a request for a web page (HTML files) and its associated media items (.jpg, .gif, .mp3 files).

When the server receives such a request, it sends (serves) the requested items back to the requester – generally a user sitting at a web browser.

What does a web client (browser) do?

It requests web pages and associated media items from web servers.

The user issues this request by "going to a web page."

It receives web pages and associated media items from web servers and displays them – or in the case of music, movie, and animation items, it plays them.

FTP Commands

FTP clients use these three commands to perform their file-transfer tasks:

PUT − copy (upload) files from your local computer to the server

GET − copy (download) files from the server to your local computer (hard disk, flash drive)

MKDIR − create a directory (folder) on the server

But you usually don't have to bother with these commands, because the FTP client issues them behind the scenes.

Instead, what you'll typically do to transfer files is:

Select the files in your local computer and drag them to the server.

Or drag files from the server to your local computer.

To create a folder on the server you'll typically:

Choose "Make new folder" (or equivalent) in the client.

Required Information for Using an FTP Client

The domain name of the web server

For example: gibson.rit.edu

Your RIT user name

For example: rpsvks

Your password

For example: ********** (Hey, that's classified!)

The FTP protocol type

FTP – Port 21 or SFTP (Secure FTP) – Port 22

You need to use SFTP (Port 22) to connect to the Gibson server.

Once connected, you need to select a folder to upload files to.

On Gibson, this has to be your www folder or a subfolder within www.

Any files in your Gibson account that are not located in the www folder or a subfolder of www will not be accessible by a web browser.

File and Folder Permissions

Once you've uploaded files to your web server – HTML docs, images, mp3s, etc. – there's one crucial step left: You need to set the correct permissions for both the uploaded files and the folders they reside in.

If the permissions are incorrectly set, no one will be able to access the files: view web pages and images in a browser, play mp3s, etc.

In some cases, the server takes care of this for you, behind the scenes. You simply upload your files and their permissions are set so that the world can view the files.

Alas, this is not the case with the RIT Gibson server. After uploading web files to Gibson, you need to manually set their permissions and the permissions of the folders they reside in.

To do this, use your FTP client to set the following permissions value for each web file you upload and each folder in the path leading to that file:

755

If your FTP client doesn't allow you to specify a numeric permissions value, set the permissions as follows:

Owner permissions: Read Write Execute (rwx)

Group permissions: Read Execute (r-x)

Public permissions: Read Execute (r-x)

Don't forget, when you upload web files to Gibson: 755 TO MAKE 'EM LIVE!

Including an index.html File in your Web Folders

One more tidbit before the Practice task for this section.

If you don't specify a file name in a URL, just a folder name, for example:

http://rachmiel.org/music

the web server (rachmiel.org) will search through the specified folder (/music) for a file whose name begins with index: index.html, index.htm, index.asp, etc.

If the server finds an index.* file, the browser will open it, even though the file's name was not included in the URL.

If the server doesn't find an index.* file, the browser will display a listing of all the files in that folder. (Uh-oh!)

We recommend that you include an index.html file in all of your web folders that contain files.

It makes typing URLs easier for users.

Instead of having to type rachmiel.org/music/index.html users can simply type rachmiel.org/music

It prevents the user from being able to violate your privacy and see a listing of all the files in your folders.

Practice: Creating an HTML5 Web Page and Making it Accessible to the World

In this Practice task, you'll create an HTML5 web page, upload it to your Gibson server account, and set file/folder permissions to make the page accessible to anyone with an Internet connection and a browser.

First you'll create the HTML web page.

1. In a text editor, open the HTML5 template you created in the HTML, Part I module.

   If you didn't create an HTML5 template, please do so now:

   a. In your editor, create a new text file.

   b. Copy/paste this HTML5 code to the file:

```
<!doctype html>

<html lang="en">

<head>
   <meta charset="utf-8">
   <title>pagetitle</title>
</head>

<body>
   pagecontent
</body>

</html>
```

   c. Save the file as html5_template.html

2. Replace *pagetitle* with an appropriate page title and save the file as index.html
3. Use your awesome HTML5 coding skills to add a handful of elements to the page: a <p> or two, an <a> anchor (link), an <ol> or <ul> list, an <hr> line, a nice juicy <img> (image), etc.
4. Verify that your page looks/works like you want it to by viewing it in a browser or two.

Next you'll upload your web page to your Gibson server account.

5. Launch your FTP client and use this information to connect to the Gibson server:

   Host: gibson.rit.edu

   Username: your RIT account username

   Password: your RIT account password

   Protocol: SFTP (Port 22)

6. Once you're connected to Gibson, navigate to your www folder (on Gibson) and create a new folder in www named test.
7. Upload your index.html file and any media files associated with it (images, audio, etc.) from

your local computer to your remote (Gibson) test folder.

And now you'll set file/folder permissions to make your web page accessible to the world.

8. Give both folders, www and test, the file permissions value 755. (Owner RWE, Group RE, Public RE.) Do the same for *all* the files you uploaded to test: index.html, images, etc.

   Remember: You need to give ALL your files and the folders they reside in the 755 file permissions value to make them accessible to web browsers. 755 TO MAKE 'EM LIVE!

9. Verify that all is well by using a browser to view your page with this URL:

   people.rit.edu/*username*/test/index.html

   Replace *username* with your RIT user name.

   Alternately, you could use this URL to view your web page, because as explained above, when a URL points to a folder, servers automatically open an index.* file in this folder:

   people.rit.edu/*username*/test

   Due to how Gibson is configured, you cannot include the www folder in your URL. These URLs won't work (try 'em and see):

   people.rit.edu/username/www/test/index.html

   people.rit.edu/username/www/test

Beautiful Coding

Just like anything else, code can be done sloppily ... or it can be a thing of beauty. We exhort you take to Door #2!

Beautiful code is much more clear and understandable than sloppy code. It looks great. And you'll be proud of it. What's not to love?

The Four Commandments of Beautiful Coding

Thou shalt indent scrupulously.

Thou shalt use sufficient white space

Thou shalt comment appropriately.

Thou shalt be consistent to the last drop.

Thou shalt indent scrupulously.

Choose a value for your indents: one tab stop, 2-3 spaces, etc. Then stick with it!

NO

```
<html lang="en">
<head>
<title>title</title>
</head>
<body>
<p>I'm a paragraph.<p>
<ul>
<li>List item 1</li>
</ul>
</body>
</html>
```

Yes!

```
<html lang="en">
<head>
  <title>title</title>
</head>
<body>
  <p>I'm a paragraph.<p>
  <ul>
    <li>List item 1</li>
  </ul>
</body>
</html>
```

Thou shalt use sufficient white space.

Use white (blank) line space to separate logical sections of your code.

NO

```
<html lang="en">
<head>
  <title>title</title>
</head>
<body>
  <p>I'm a paragraph.<p>
  <ul>
    <li>List item 1</li>
  </ul>
</body>
</html>
```

Yes!

```
<html lang="en">

<head>
  <title>title</title>
</head>

<body>
  <p>I'm a paragraph.<p>
  <ul>
    <li>List item 1</li>
  </ul>
</body>

</html>
```

Thou shalt comment appropriately.

Use comments to clarify your code; don't write a novel!

NO

```
<html lang="en">

<head>
  <title>title</title>
</head>

<body>
  <p>I'm a paragraph.<p>
  <ul>
    <li>List item 1</li>
  </ul>
</body>

</html>
```

Yes!

```
<html lang="en">

<head>
  <!-- The page title is displayed
       in the browser tab. -->
  <title>title</title>
</head>

<body>
  <p>I'm a paragraph.<p>
  <ul> <!-- unordered list -->
    <li>List item 1</li>
  </ul>
</body>

</html>
```

Here's the syntax of an HTML comment:

```
<!-- HTML comment -->
```

Note that CSS comments have a different syntax:

```
/* CSS comment */
```

Both HTML and CSS comments can span multiple lines.

```
<!-- HTML comment line 1
     ... ... .... line 2 -->

/* CSS comment line 1
   ... ... ... line 2 */
```

Thou shalt be consistent to the last drop.

Once you decide on a coding style, stay with it throughout all your code.

NO	Yes!

`<style>`
`  h1 {color:red;}`
`h2 { color: blue;`
`  }`
`</style>`
	`<style>`
	`  h1 { color: red; }`
	`  h2 { color: blue; }`
	`</style>`

Practice: Beautify Some Uggily Code
1. Use the View, Source command of your browser to find a web page whose source code egregiously violates the Four Commandments of Beautiful Coding: is poorly laid out, inconsistent, over- or under-commented, etc. 2. Copy/paste the code to a blank text file, and follow the Commandments to turn it into a thing of beauty and grace.

HTML Validation Basics

What is valid HTML code?

Code that adheres strictly to a specified HTML standard. The latest standard is HTML5.

Why should you bother writing valid HTML5 code?

- Because it helps you debug your web pages: find/fix HTML coding errors.
- Because it provides consistency across different browsers/platforms.
- Because it provides forward compatibility with future browsing devices.
- Because it plays well with CSS.

Will invalid HTML5 code do just as good a job?

NO!

Or, more accurately, perhaps 75% of the time it will, but the other 25% might be catastrophic in terms of page display.

It's for this 25% that you do the extra work to write valid code.

It is the Way of the Lonely Web Designer.

How to Write Valid HTML5 Code

Begin all your web pages with a valid HTML5 template.

We showed you how to create an HTML5 template earlier in this module.

Adhere strictly to HTML5 syntax.

For details, turn to W3Schools (w3schools.com/tags).

Include all attributes required by HTML5. Don't use any attributes not supported by HTML5.

To learn the required/unsupported attributes for a tag, see the tag's entry at W3Schools.

How to Validate an HTML5 Web Page

To validate an HTML5 web page, you run it through a reputable HTML validator until it validates without any errors.

The W3C Validator (validator.w3.org) is the gold standard.

You can send the W3C Validator an HTML file that is:

On your local computer – by using the Validate by File Upload tab

On the Internet – by using the Validate by URI tab

You can also test HTML5 code (snippets) directly by using the Validate by Direct Input tab.

The validation process is iterative:

1. Run the HTML page through the W3C Validator.
2. Fix all the errors the validator reports.
3. Loop 1-2 until the Validator says these magic words:

   This document was successfully checked as HTML5!

Practice: Create a Multi-Page Site that Validates as HTML5

1. Using a text editor and your HTML5 template, create a set of 3-5 web pages that are all linked together via a navigation bar that:

   Is formatted as an unordered list (<ul>).

   Has one link (<a>) for each page, including the current page.

   Appears in the same exact position on each page.

   > Name your home page index.html and give the other pages descriptive names: news.html, info.html, contact.html, etc.

2. Use the iterative process to develop your site: Code, test, re-code, re-test, etc. Make sure that every navbar link on every page does its job correctly.

3. When your site is working 100%, use the W3C Validator (validator.w3.org) to validate each of its pages. Don't rest until you see this lovely line:

   This document was successfully checked as HTML5!

   > Tip: The W3C Validator is quite good (if a bit verbose) at describing validation errors and telling you where they occur in the HTML document. Take advantage of this!

4. When every page of your site validates, upload the whole site – including any media files (images, sounds) you added to individual pages – to a new folder on your Gibson account. For instructions, see the first Practice task of this module.

   Verify that all is well by viewing every page of the uploaded site in a browser