

Webucator Courseware Authoring

(WCW101 version 1.2.0)

Copyright Information

© Copyright 2009 Webucator. All rights reserved.

The Author

Nat Dunn

Nat Dunn founded Webucator in 2003 to combine his passion for Web development with his business expertise and to help companies benefit from both. Nat began programming games in Basic on a TRS-80 at age 14. He has been doing Web development and Web development training since 1998.

More information on Nat can be found at <http://www.webucator.com/NatDunn.cfm>.

Table of Contents

1. The Process.....	1
Constituents.....	1
Versioning System.....	1
The Process.....	1
Deliverables.....	1
2. Creating Lesson Files.....	3
Structuring Lesson Files.....	3
Naming Lesson Files.....	3
The <head /> Tag.....	3
The <body /> Tag.....	4
Block Elements.....	6
Inline Elements.....	9
Creating Cross References.....	13
Validating the Lesson.....	13
3. Creating External Files.....	15
The WCWC Directory Structure.....	15
Creating External Files.....	15
Emphazing Content.....	17
Omitting Content.....	17
Files that Don't Get Validated.....	18
4. The Setup Directory.....	21
Setup Files.....	21
Course Descriptor.....	21
Read Me.....	23
5. Creating Slides.....	25
Section Summaries.....	25
Special Cases.....	25
6. Style Guide.....	27
General Rules.....	27
Specific Rules.....	27
Demos vs. Code Blocks.....	28
Syntax and Code Blocks.....	28
Really Specific.....	28

1. The Process

In this lesson, you will learn...

1. To understand how the Webucator authoring process is managed.

1.1 Constituents

There are usually three people involved in the authoring process:

1. Author
2. Editor
3. Project Manager (PM)

In most cases, the author or the editor will be the first person to deliver the course.

1.2 Versioning System

All course files are stored in a subversion repository at <http://svn.webucator.com/svn/Category>.

1.3 The Process

Each constituent is responsible for specific deliverables, which are laid out in a spreadsheet containing project specific milestones and timeline.

The process is most easily understood by laying out the deliverables for a five-day course:

Deliverables

Deliverable	Description	Constituent
Complete outline	Text format. Must include: <ul style="list-style-type: none">• Overview• Prereqs• Goals• Outline• Tech Requirements	Author
Review Outline	Must agree that complete outline is appropriate.	Editor
Sign off	PM signs off on deliverable.	PM
Course Skeleton	All shell lesson files created with titles, h1s, h2s, and placeholders for demos and exercises. (Examples of Demo and Exercise placeholders are shown after this table.) The ClassFiles structure must also be built out.	Author
Build File	Build file created and first build should be run.	Author
Review of Build	Build reviewed and compared to text outline. Editor agrees that Skeleton build matches Complete outline and that the demos and exercises are appropriate.	Editor
Sign off	PM signs off on deliverable.	PM

Sample Demo Placeholder

```
<p>A demo showing how to write a for loop</p>
```

Sample Exercise Placeholder

```
<div class="Exercise">  
  <h1>Looping</h1>  
  <p>An exercise to practice creating a for loop.</p>  
</div>
```

1.4 Conclusion

In this lesson, you have learned how a courseware project is managed and what your role is in the process.

2. Creating Lesson Files

In this lesson, you will learn...

1. To use the Webucator CourseWare Creation (WCWC) system
2. How Webucator manuals are created.
3. To properly structure lesson files.
4. To add headings, paragraphs, lists, and tables.
5. To add inline markup.
6. To insert images.
7. To create cross references and external links.
8. To create footnotes.
9. To validate lesson files.

Webucator manuals are broken into lessons. Each lesson is made up of one XHTML file with 0 or more references to external files (See page 15) which are maintained in a simple XML format.

This document, which itself was created using this system, provides instructions for creating lesson files.

2.1 Structuring Lesson Files

Each lesson file is a valid XHTML Strict file, but only a subset of HTML tags is used and the structure is very specific.

Naming Lesson Files

Lesson files should be named so that the file name relates to the lesson content. Words in the file name should be separated with dashes.

The `<head />` Tag

The `<head />` tag contains several tags that should not be changed. Tags that you should modify are listed below.

- The `<title />` tag should contain the lesson title, which should not be repeated in the body as a heading.
- The Author `<meta />` tag should be changed to reflect the actual author of the lesson.
- The Category `<meta />` tag should be changed to reflect the actual category of the lesson. There can only be one category. If you are unsure of the value

Creating Lesson Files

for this, please email Nat at courseware@webucator.com. In most cases, you will use the same category for every lesson in a particular manual.

Code Sample

LessonFiles/Demos/Template-Head.html

```
<!--
This is the template for writing lessons used in Webucator manuals.  A
manual will be made up of multiple lesson files.
It uses a subset of XHTML Strict 1.0.

Required resources
- lesson.css: for styling the authoring environment.  No new styles should
be defined.
- lesson.js: for validating the structure.

Document History
- 2007-Nov-09 - first published by Nat Dunn
-->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Lesson Title</title><!--Insert the title of the lesson in title
case (e.g, Conditionals and Loops)-->
  <meta name="Author" content="Nat Dunn" /><!--Insert your name as the
value of the content attribute-->
  <meta name="Category" content="WCWC" /><!--Insert the lesson category
(e.g, CSS, JavaScript, Silverlight, XML, etc.) as the value of the content
attribute.  If you don't know what category to insert, email Nat at
ndunn@webucator.com.-->
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <link rel="stylesheet" type="text/css" href="http://www.webucator
manuals.com/WCWC/Authoring/lesson.css"/>
  <script src="http://www.webucatormanuals.com/WCWC/Authoring/les
son.js" type="text/javascript"></script>
</head>

-----CODE OMITTED-----
```

Code Explanation

The section which you need to modify is highlighted.

The `<body />` Tag

id Attribute

The `<body />` tag should have an id attribute that uniquely identifies the lesson. This will often be the file name, but be aware of file names that aren't unique. For example, an HTML course might be combined with a CSS course in a single manual. If both courses have a lesson with the filename Basics.html there could be a conflict. A solution is to name the files HTML-Basics.html and CSS-Basics.html with `<body />` id attributes set to "HTML-Basics" and "CSS-Basics", respectively.

Subsections

The body is divided into a `<div />` and an optional `` as shown below:

1. `<div id="Content" />` - contains the lesson content.
2. `<ol id="Footnotes" />` - footnotes.

`<div id="Content" />`

The Content section is structured as follows:

1. `<ol id="LearnItems" />` (required)
2. `<div id="Introduction" />` (optional)
3. Block Elements: Headings, Paragraphs, Lists, Tables, Code Blocks, Syntax Blocks, Demos, and Exercises
4. `<div id="Conclusion" />`

Each section is described in detail below.

`<ol id="LearnItems" />`

The LearnItems section is a ordered list showing what is to be covered in the lesson. In the manual, this section will be prefixed with "In this lesson you will learn...", so each list item should begin with a word that flow from that (e.g, "to, how, about, etc.").

Note that each learn item should end with a period.

Creating Lesson Files

Code Sample

LessonFiles/Demos/Template-LearnItems.html

```
<!--
This is the template for writing lessons used in Webucator manuals.  A
manual will be made up of multiple lesson files.
It uses a subset of XHTML Strict 1.0.

Required resources
- lesson.css: for styling the authoring environment.  No new styles should
be defined.
- lesson.js: for validating the structure.

Document History
- 2007-Nov-09 - first published by Nat Dunn
-->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Lesson Title</title><!--Insert the title of the lesson in title
case (e.g, Conditionals and Loops)-->
    <meta name="Author" content="Nat Dunn" /><!--Insert your name as the
value of the content attribute-->
    <meta name="Category" content="WCWC" /><!--Insert the lesson category
(e.g, CSS, JavaScript, Silverlight, XML, etc.) as the value of the content
attribute.  If you don't know what category to insert, email Nat at
ndunn@webucator.com.-->
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <link rel="stylesheet" type="text/css" href="http://www.webucator
manuals.com/WCWC/Authoring/lesson.css"/>
    <script src="http://www.webucatormanuals.com/WCWC/Authoring/les
son.js" type="text/javascript"></script>
  </head>
  <body id="Lesson-Title">
    <div id="Content">
      <ol id="LearnItems">
        <li>To use...</li>
        <li>To create...</li>
        <li>About...</li>
      </ol>

      -----CODE OMITTED-----

```

```
<div id="Introduction" />
```

The optional Introduction section can contain paragraphs, lists and tables.

Block Elements

This section describes the block-level elements allowed in the body. The elements can appear in any order and can all be repeated. Except where indicated, you should assume that you can nest these elements according to the rules of XHTML.

Headings

You can use `<h1 />`, `<h2 />`, `<h3 />`, and `<h4 />` headers only. They should be structured in a logical order, so that `<h4>` elements are always preceded by `<h3>` elements, `<h3>` elements are always preceded by `<h2>` elements, and `<h2>` elements are always preceded by `<h1>` elements. Think of legal documents.

`<h1 />` and `<h2 />` elements appear in the table of contents.

Paragraphs

`<p />` tags denote paragraphs.

Blockquotes

`<blockquote />` tags denote blocks of quoted text. Do not use them for formatting purposes alone.

Lists

- `` - for ordered lists - it's okay to nest lists. Sublists have no class.
- `` - for unordered lists - it's okay to nest lists. Sublists have no class.

Tables

Tables are denoted with `<table />` and are structured as follows:

- `<caption />`
- `<thead />`
 - `<tr />`
 - `<th />` - colspan and rowspan attributes are allowed.
- `<tbody />`
 - `<tr />`
 - `<td />` - colspan and rowspan attributes are allowed.

The `<table />` tag takes an optional "Wide" class which indicates that the table can stretch into the margin in the printed manual.

Code Blocks

Code Blocks are denoted with `<div class="CodeBlock" />`:

- used for short code blocks. Longer code samples should be stored in external files and referenced as Demos.
- All whitespace is relevant. The start of the code should be flush up against the open `<div class="CodeBlock" >` tag and the end of the code should be flush up against the corresponding close `</div>` tag. Tabs should be used for indenting.
- output will be preformatted.
- only subtag allowed is ``
- optional title attribute is used to provide a caption in the output
- we recommend that you write these in code view so that you can control spacing and linefeeds.
- Note that when you preview them in the browser, they will *not* appear preformatted.

```
var foo = "Hello World"; //set variable
alert(foo); //alert message
```

Syntax Blocks

Syntax Blocks are denoted with `<div class="SyntaxBlock" />` and are structured as follows:

- Just like CodeBlocks except that they are prefixed with "Syntax: " and the optional title in the output.

Special Inline Tags

Syntax Blocks can have special inline `` tags for optional code and substitution code (placeholder text that the developer would substitute, like a class name).

Here is an example:

Syntax

```
<div class="SyntaxBlock" title="Java Compile Command">javac <span
class="Optional">[-classpath <span class="Subst">classpathEle
ments</span>]</span> <span class="Optional">[-d outputDirecto
ry]</span> <span class="Subst">className</span>.java</div>
```

And here is the approximate output:

Syntax**Java Compile Command**

```
javac [-classpath classpathElements] [-d outputDirectory]
      className.java
```

Demos

Demos are denoted with `<div class="Demo" />` and are structured as follows:

- `Sample`
- `<div class="CodeExplanation" />`
 - Paragraphs and Lists

Exercises

Exercises are denoted with `<div class="Exercise" />` and are structured as follows:

- `<h1 />` - Exercise title
- `<div class="Duration" />`
- `Num Minutes`
- `Num Minutes`
- `<ol class="Instructions" />` - it's okay to nest lists
- `<div class="CodeSample" />` - zero or more, structured like Demo blocks (See page 9)
- `<div class="Challenge" />` - zero or more
- `<ol class="Instructions" />` - it's okay to nest lists. Sublists have no class.
- `<div class="Solution" />` - zero or more, structured like `<div class="Demo"/>`
- `<div class="SolutionToChallenge" />` - one for each challenge

Inline Elements**Emphasis**

Emphasized text is marked up with `` tags.

Links

Links to web resources and email addresses should use standard HTML syntax¹: `Link text`. In the online manual, this will show up as a normal link. In the print manual, the link text will be footnoted and the link will show up in the footnote.

File Paths

File paths should be marked up as `File Name` if you want them clickable and `File Path` if you don't want them clickable. In the print manual, the two will appear exactly the same. In the online version, links (but not spans) will be clickable and bring up the code sample.

Code Snippets

Code snippets are marked up with `<code />` tags.

Images

Images are marked up as `` tags.

The `` tag takes an optional "Wide" class which indicates that the table can stretch into the margin in the printed manual.

By default images will appear as blocks. To have them appear inline, assign the `` tag the "Inline" class.

Book Titles

Book titles should be marked up as follows: ``. The id is required so that it can be footnoted.

Footnotes

Footnotes are stored in an ordered list at the end of the lesson: `<ol id="Footnotes" />`

1. Standard link syntax for a tags is explained in Webucator's online [HTML tutorial](#).

Each footnote is a list item. It should begin with an empty `` tag and followed by a div tag containing any block-level elements as shown below:

Code Sample

LessonFiles/Demos/Template-Footnotes.html

```
<!--
This is the template for writing lessons used in Webucator manuals.  A
manual will be made up of multiple lesson files.
It uses a subset of XHTML Strict 1.0.

Required resources
- lesson.css: for styling the authoring environment.  No new styles should
be defined.
- lesson.js: for validating the structure.

Document History
- 2007-Nov-09 - first published by Nat Dunn
-->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Lesson Title</title><!--Insert the title of the lesson in title
case (e.g, Conditionals and Loops)-->
    <meta name="Author" content="Nat Dunn" /><!--Insert your name as the
value of the content attribute-->
    <meta name="Category" content="WCWC" /><!--Insert the lesson category
(e.g, CSS, JavaScript, Silverlight, XML, etc.) as the value of the content
attribute.  If you don't know what category to insert, email Nat at
ndunn@webucator.com.-->
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <link rel="stylesheet" type="text/css" href="http://www.webucator
manuals.com/WCWC/Authoring/lesson.css"/>
    <script src="http://www.webucatormanuals.com/WCWC/Authoring/les
son.js" type="text/javascript"></script>
  </head>
  <body id="Lesson-Title">
    <div id="Content">
      <ol id="LearnItems">
        <li>To use...</li>
        <li>To create...</li>
        <li>About...</li>
      </ol>
      <div id="Introduction">
        <p>You might start your lesson with an introduction or you might have
the <code>LearnItems</code> stand alone as the introduction.</p>
      </div>
      <h1 id="TagIdentifier">First Heading 1</h1>
      <p>Paragraph content</p>
      <div id="Conclusion">
        <p>In this lesson, you have learned...</p>
      </div>
    </div>
  </body>
</html>
```

```
</div>
<ol id="Footnotes">
  <li>
    <a href="#TagIdentifier"></a><!--This <a /> tag should be empty, but
don't use the XML shortcut close syntax.-->
    <div>
      <p>This is where the footnote content goes.</p>
    </div>
  </li>
</ol>
</body>
</html>
```

Important: Elements that contain block-level elements cannot be footnoted. For example, a `<table />` cannot be footnoted as it's unclear where the footnote marker should show up. You could footnote the `<caption />` or the first `<th />` instead.

2.2 Creating Cross References

You can id elements that do not have reserved ids. Each id should be unique throughout the document.

- You can reference any element with an id like this: `text`.
- If you are referencing the beginning of the file, you just need the file name: `text`.
- If you are referencing a tag in the same file, you just need the anchor name: `text`.

2.3 Validating the Lesson

There are two steps to validating the file:

1. Open the file in Firefox to have it validated with JavaScript.
2. Upload the file to the [W3C Markup Validation Service](#). It should be valid valid XHTML 1.0 Strict.

2.4 Conclusion

In this lesson, you have learned how to create lesson files.

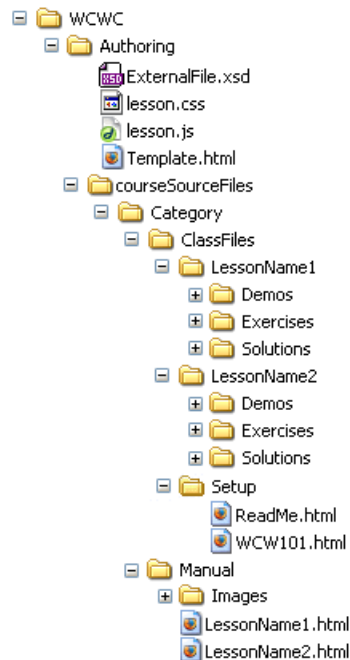
3. Creating External Files

In this lesson, you will learn...

1. About the directory structure.
2. To create external files.
3. To mark up sections to be emphasized in the text.
4. To mark up sections to be omitted in the text.
5. To create the Setup directory.

3.1 The WCWC² Directory Structure

As an author of Webucator courses, you should start with a directory structure that looks like this:



3.2 Creating External Files

Any text files can be references from a lesson file. They just need to be marked up with the very simple Webucator Courseware XML language.

You should start by writing the Demo, Exercise, and Solution files in whatever tool and format you would normally use. After you have tested them and feel comfortable

Creating External Files

that all are working as they should, you should open them in an XML editor and mark them up so that they are valid according to the XML Schema shown below:

Code Sample

ExternalFiles/Demos/ExternalFile.xsd

```
<xs:schema xmlns="http://www.webucator.com/Schemas/Courseware"
xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.webucator.com/Schemas/Courseware" elementFormDefault="qualified">
  <xs:element name="File">
    <xs:complexType mixed="true">
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="Omit" type="xs:string"/>
        <xs:element name="Em" type="xs:string"/>
      </xs:choice>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

The first thing to do is to wrap the whole file in `<cw:File>` tags as shown below:

Syntax

Step 1 of making external files compatible with WCCS

```
<cw:File xmlns:cw="http://www.webucator.com/Schemas/Courseware"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.webucator.com/Schemas/Courseware
../../../../../../../../Authoring/ExternalFile.xsd"><![CDATA
TA[Your code starts here. Notice how it bumps right against the
beginning of the CDATA section

Here's some more code.

This code is what you want to emphasize in the manual.

This is a lot of extra code that will remain in the
file but should be omitted from the manual.

This is the end of the file. Notice how it bumps right up against
the close of the CDATA section.]]>
</cw:File>
```

Code Explanation

Things to notice:

1. Namespace³: `http://www.webucator.com/Schemas/Courseware` - qualifier is "cw".

3. Namespaces are covered in Webucator's online [XML Schema tutorial](#).

2. The file must be well-formed XML. Because many code files include characters that will make the file poorly formed, you should always wrap all text in CDATA⁴ sections.
3. `../../../../Authoring/ExternalFile.xsd` is a relative path to the schema, so that the file can easily be validated by any XML authoring tool.

3.3 Emphazing Content

To draw attention to content in the manual, wrap it in `<cw:Em>` tags as shown below:

Syntax

Step 2 of making external files compatible with WCCS

```
<cw:File xmlns:cw="http://www.webucator.com/Schemas/Courseware"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLo
cation="http://www.webucator.com/Schemas/Courseware
../../../../Authoring/ExternalFile.xsd"><![CDATA[Your code starts
here. Notice how it bumps right against the beginning of the CDATA
section
```

Here's some more code.

```
]]><cw:Em><![CDATA[This code is what you want to emphasize in the
manual.]]></cw:Em><![CDATA[
```

This is a lot of extra code that will remain in the file but should be omitted from the manual.

```
This is the end of the file. Notice how it bumps right up against
the close of the CDATA section.]]>
</cw:File>
```

3.4 Omitting Content

To omit content in the manual, wrap it in `<cw:Omit>` tags as shown below:

4. CDATA is explained in Webucator's online [XML tutorial](#).

Syntax

Step 3 of making external files compatible with WCCS

```
<cw:File xmlns:cw="http://www.webucator.com/Schemas/Courseware"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLo
cation="http://www.webucator.com/Schemas/Courseware
../../../../../../../../Authoring/ExternalFile.xsd"><![CDATA[Your code starts
here. Notice how it bumps right against the beginning of the CDATA
section
```

Here's some more code.

```
]]><cw:Em><![CDATA[This code is what you want to emphasize in the
manual.]]></cw:Em><![CDATA[ ]]><cw:Omit><![CDATA[
```

This is a lot of extra code that will remain in the file but should be omitted from the manual.

```
]]></cw:Omit><![CDATA[This is the end of the file. Notice how it
bumps right up against the close of the CDATA section.]]
</cw:File>
```

3.5 Files that Don't Get Validated

Files with the following extensions are copied as is to the output folder:

- gif
- jpg
- png
- mdb
- zip
- pdf
- txt
- class
- dll
- pdb
- cache
- sln
- suo

All other files in the ClassFiles directory will be transformed, so they must be marked up as valid WCCW files.

3.6 Conclusion

In this lesson, you have learned to create and mark up external files. That's all there is to it.

4. The Setup Directory

In this lesson, you will learn...

1. To create the SetupDirectory directory.

4.1 Setup Files

The SetupDirectory directory should contains the following files:

1. ReadMe.html
2. Any supplemental setup files (e.g, a database script or a build file)

Course Descriptor

There should be one course descriptor for each class that can be created from these class files. For example, if there could be an introductory class, intermediate class, and advanced class, then there should be three course descriptors. A course descriptor is an HTML file that follows a very specific structure. The demo below shows how it is formatted.

Code Sample

TheSetupDirectory/Demos/ReadMe.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>Read Me</title>
  </head>
  <body>
    <h1>HTML Setup Instructions</h1>
    <p>This document contains setup instructions for Webucator's HTML courses.</p>
    <section id="allCourses">
      <ol>
        <li>If you haven't already, download the class files listed at the bottom of this page.</li>
        <li>Create a folder in the root of your C drive called "Webucator" and extract the class files zip file to that folder.</li>
        <li>Make sure that you have installed the editor you plan to use in class.</li>
        <li>See the course outline for your class to determine which editors you can use.</li>
        <li>If this is for a private onsite class make sure that each computer has the editor installed.</li>
        <li>If you are using a trial version of the editor, be aware that some products require a unique trial key on each computer.</li>
      </ol>
    </section>
    <section id="HTML01">
      <p class="ClassOutline">Outline available at <a href="http://www.webucator.com/WebDesign/HTML01.cfm">http://www.webucator.com/WebDesign/HTML01.cfm</a></p>
      <p class="ClassFiles">Class files available at <a href="http://www.webucator.com/ClassFiles/HTML01.zip">http://www.webucator.com/ClassFiles/HTML01.zip</a></p>
      <p class="TechReqs">Technical requirements listed at <a href="http://www.webucator.com/WebDesign/HTML01.cfm#TechReqs">http://www.webucator.com/WebDesign/HTML01.cfm#TechReqs</a>.</p>
    </section>
    <section id="HTML201">
      <p class="ClassOutline">Outline available at <a href="http://www.webucator.com/WebDesign/HTML201.cfm">http://www.webucator.com/WebDesign/HTML201.cfm</a></p>
      <p class="ClassFiles">Class files available at <a href="http://www.webucator.com/ClassFiles/HTML201.zip">http://www.webucator.com/ClassFiles/HTML201.zip</a></p>
    </section>
  </body>
</html>
```

```
cator.com/ClassFiles/HTM201.zip">http://www.webucator.com/Class
Files/HTM101.zip</a>
</p>
<p class="TechReqs">Technical requirements listed at <a
href="http://www.webucator.com/WebDesign/HTM201.cfm#TechReqs">http://www.we
bucator.com/WebDesign/HTM101.cfm#TechReqs</a>.</p>
</section>
</body>
</html>
```

Read Me

The [ReadMe.html](#) file has instructions for how to get started with setup. This is where people setting up for class will go *after* they have properly installed all the software needed for class. It should tell people where to put the class files and how to set them up and how to run tests to make sure setup has been done correctly.

If the class can be broken up into different classes, the [ReadMe.html](#) file should explain what needs to get done for each class.

This file should be valid XHTML Strict. It must be broken into HTML5 `<section>` tags as follows:

1. One `<section>` block that applies to all courses (`<section id="allCourses">`)
2. One `<section>` block for each course (e.g, intro, intermediate, advanced)

4.2 Conclusion

In this lesson, you have learned to create the SetupDirectory directory.

5. Creating Slides

In this lesson, you will learn...

1. To create corresponding slides.

Some courseware is just PowerPoint slides with no supporting content. The instructor is left to fill in all the blanks.

Some courseware is just a manual written in prose and the instructor has nothing good to present.⁵

Some companies do provide both, but they have a hard time keeping the slides in sync with the manual. A large part of the problem is that the manual and the slides are written with separate tools and sometimes by separate people. If a section is removed or added to the manual, the authors must remember to make the same change in the slides.

Webucator courseware attempts to deliver the best of both worlds: great manuals written in prose with slides that directly correspond to the manual. The slides even give page numbers to the corresponding content. And the instructor has a PDF version of the full manual with the slides in the same document. The slides and the manual sections are cross-linked to make it easy to go back and forth.

The upkeep is a lot easier because the slides are written in HTML inside of the lesson itself.

5.1 Section Summaries

Each `<h1/>`, `<h2/>`, and `<h3/>` can/should be followed by a `<div class="SectionSummary" />` tag that is divided into `` and `` tags. Each list will become a slide that will link back to the corresponding heading. The `title` attribute of the list will be the title of the slide and the list items will be the bullets. It is okay to nest lists. Nested lists should not have `title` attributes.

Special Cases

Cross References

You can create a cross reference to the lesson content so that you do not have to retype the same code. This is most useful for code and syntax blocks and demos. To create a cross reference, put an `<a />` tag in the list item with the class of

5. Webucator's courseware used to fit this model.

"CrossReference" and the href value of "#refID", where refID is the id of the element being referenced.

Cross Reference

```
<li><a class="CrossReference" href="#demoFoo">CR</a></li>
```

Images

Images can be referenced in list items with a standard image tag:

Image in Slide

```
<li></li>
```

Note that if the image is already in the manual, you can reference the image by cross reference.

Exact Copies

If you want a full slide to be an exact copy of an existing element in the lesson, you can use the following tag instead of an `ol` or `ul`:

Image in Slide

```
<a href="#id" class="ExactCopy">Copy</a>
```

This is most useful for existing lists in the manual, but can be used for other content as well. Note that the element being referenced should have a `title` attribute, which will be used as the title of the slide.

5.2 Conclusion

In this lesson, you have learned to use `SectionSummary` divs to create slides that directly correspond to lesson content.

6. Style Guide

In this lesson, you will learn...

1. About basic style rules for Webucator manuals.

6.1 General Rules

1. The ideal flow should be:
 1. Explain.
 2. Demo(s) with explanations.
 3. Exercise.
 4. Solution.
2. Keep in mind that the PDF of the courseware might be used for presentation.
 1. Use HTML lists (ul,ol) instead of comma-delimited lists.
 2. When you have multiple related headings followed by short descriptions consider using tables instead.
 3. Keep your descriptions as short as possible and use footnotes to make caveats or to point people to further reading on subjects not necessary for completing the labs. The trainer can fill in as appropriate.
3. Keep in mind that students will NOT be reading the manual cover to cover and will not be reading every paragraph during class.
 1. If you can make it shorter, do it.
 2. If it's not necessary, leave it out or put it in a footnote or in a reference table.
4. Ideally, no code will be shown that is beyond the scope of the class. If demo code does contain stuff that is beyond the scope of the clas:
 1. Be careful to point out that it is beyond the scope of the class.
 2. Ideally, provide a brief summary of what the code does (perhaps in a comment).

6.2 Specific Rules

Headings

1. Use appropriate decreasing levels of headings. For example, h1 subheadings should be h2s, h2 subheadings should be h3s and so on.

Lists

1. Use periods at the end of list items in lists in which one or more of the list items contains a phrase. When in doubt, use periods.
2. When leading into a list, end the previous sentence with a colon. For example, "The following browsers support CSS pseudo-classes:"

Demos vs. Code Blocks

1. In most cases, Demos are preferred to Code Blocks. Only use Code Blocks for:
 1. incomplete code.
 2. to draw further attention to something contained within a demo when explaining the demo.

<code>

1. Use <code> tag when mentioning syntax words, library classes/functions, etc.

```
<span class="FilePath">
```

1. Use for any file or directory names that come from the Demos/Exercises/Solutions.

Syntax and Code Blocks

1. For syntax blocks, always use the title attribute. For example:

```
Syntax  
SyntaxBlock Syntax  
<div class="SyntaxBlock" title="Basic SELECT statement">...
```

2. For code blocks, the title attribute is optional.

6.3 Really Specific

1. Avoid the word "utilize". Use "use" instead. See <http://ezinearticles.com/?Writing---How-To-Use-Use-Versus-Utilize-Correctly?&id=479574> for an explanation on this.

6.4 Conclusion

In this lesson, you have learned some basic style rules for Webucator manuals.