

CIS 4004: Web Based Information Technology Spring 2011

XHTML – Part 1

Instructor : Dr. Mark Llewellyn
markl@cs.ucf.edu
HEC 236, 407-823-2790
<http://www.cs.ucf.edu/courses/cis4004/spr2011>

School of Electrical Engineering and Computer Science
University of Central Florida



Overview of Markup Languages

- A **markup language** is simply a set of rules that defines the layout, format, or structure of text within a document.
- After markup instructions are added to a document, the document must be read, or processed, by a program that knows how to interpret the markup elements.
- Markup languages existed long before the World Wide Web. They were used primarily in the publishing industry prior to their adaptation to computer programming.
- Work began in the 1960s to develop a standardized document markup language that would be platform independent.



Overview of Markup Languages

- The **Standard Generalized Mark Language (SGML)** was the result of this initiative and was the first standardized markup language to gain acceptance.
- It wasn't until the Web exploded in popularity in the mid-1990s that the benefits of an open standard for markup languages became overwhelmingly apparent.



Overview of Markup Languages

- SGML is the ancestor of, and provides the framework for, current Web markup languages, including XHTML, XML, and HTML.
- SGML was developed as a markup language for large documents, such as technical documentation.
- It was adopted as an international standard by the [International Organization for Standards \(ISO\)](#) in 1986, and has been widely used by many industries, including the automotive industry, the health care industry, the IRS, and the US Department of Defense, for large scale documentation projects.
- SGML is extremely complex, and thus very expensive. SGML has proved useful mainly to organizations that have the expertise and budget to implement the expansive SGML specification.



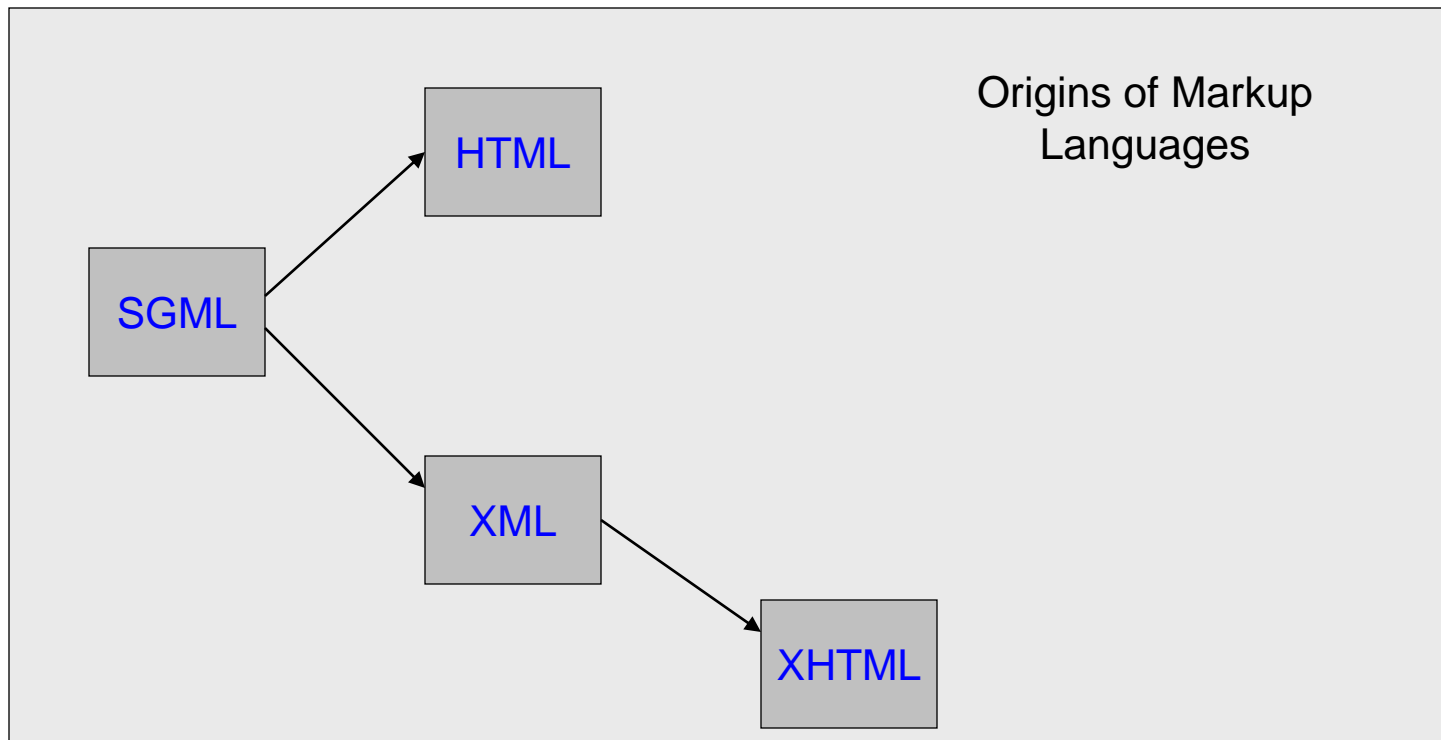
Overview of Markup Languages

- When the World Wide Web was in its infancy in the late 1980s and early 1990s, SGML was the perfect tool for building the markup language that would be used to create documents for this new medium.
- **Hypertext Markup Language (HTML)** was developed as a lightweight SGML by researchers at CERN (European Organization for Nuclear Research) in the early 1990s.
- CERN had been involved with working on the SGML specification for many years.
- HTML was much smaller than SGML and gained widespread acceptance very quickly. It provided content developers with a portable document format that was not tied to any particular program or platform, and being an open standard, it was completely free to use.



Overview of Markup Languages

- HTML was adopted shortly after its development by the W3C (www.w3c.org), which continues to maintain the HTML specification, currently at version 4.01 (although this will also be the last version of HTML).



Overview of Markup Languages

- Because HTML documents are simple text documents embedded with markup elements, they are completely portable across platforms and programs.
- HTML documents can be displayed using any program running on any operating system that knows how to interpret the HTML language.
- This gives developers an incredible amount of flexibility and allows them to move files freely among platforms and programs.
 - For example, an HTML file created with a Mac text editor would look the same when open in a Windows text editor, and would be displayed the same when viewed using Netscape Navigator or Internet Explorer, either on a Mac or a PC.



Limitations of HTML

- As web technologies continue to advance at a very rapid pace, HTML has been pushed to its limits by developers and vendors.
- Somewhat ironically, the traits of HTML that helped build its popularity – its small size, limited number of elements, and ease of use – have become its downfall.
- HTML is a fixed specification with a finite set of elements. It is not extendable, and as a result of this limitation, Web developers and software vendors have stretched the usefulness of HTML almost to a breaking point.
- Browser vendors such as Microsoft and Netscape, have added proprietary features and additional HTML elements to their browsers based on demands for more functionality, but in so doing, they have compromised one of the most important benefits that HTML has to offer – portability. Given these proprietary additions to particular browsers, HTML pages developed for use in one Web browser may not display the same way when displayed in another browser or on another platform.



Limitations of HTML

- Even though HTML has syntax rules, Web browsers have always been fairly forgiving in that they allow poorly written HTML code to be displayed.
- If a browser encounters code that is written incorrectly, it can usually compensate and will display the correct output.
- This is unfortunate in that it has allowed millions of poorly written pages to be published on the Web. It is also unfortunate because the newest breed of Web clients, including cell phones, PDAs, and other devices are not as forgiving as Web browsers and will either display the pages incorrectly or not at all.



Limitations of HTML

- Despite all of its benefits and the revolution it helped to spark, HTML has some serious limitations that inhibit its future usefulness as Internet technologies continue to advance. These include:
 - HTML elements are primarily used for defining presentation and formatting styles, but they do not provide any information about the data itself (data without context).
 - HTML has a finite number of elements, which cannot be extended or customized.
 - HTML does not force documents to adhere to strict syntax rules, making it difficult for parsers to interpret poorly written code.
 - Until recently, Web browsers were the primary client program used to view content on the Web. However, HTML's limitations are further being felt with the introduction of new technologies, such as wireless devices like cell phones, voice, and speech programs, and PDAs.
 - Even though HTML has syntax rules, Web browsers have always been fairly forgiving in that they allow poorly written HTML code to be displayed.



Limitations of HTML

- As a result of these limitations, the most recent version of HTML, HTML 4.01, will be the last.
- The first version of XHTML, 1.0, was released by the W3C in 2000 as the successor to HTML. This is the reason that the terms HTML and XHTML are often used interchangeably.
- **Extensible Hypertext Markup Language (XHTML)** is the modern “version” of HTML.
- XHTML 1.0 is the current standard adopted by W3C in 2002.



XML – The Future of Web Markup Languages

- The limitations of HTML made it very clear that a new and better language was necessary for formulating Web documents.
- In addition, many companies were adding transactional functionality to their Web sites, such as allowing visitors to purchase items and services online.
- This marked a radical departure from the first generation of websites, which mainly provided static information that was easily stored as text. These new websites relied heavily on data gathered from different sources, such as databases, news feeds, and other Web sites.
- The resulting language was the **Extensible Markup Language (XML)**. XML is an extensible language because, unlike HTML, it allows users to define their own tags.



XML – The Future of Web Markup Languages

- The first recommendation, XML 1.0, was released in 1998.
- The XML family of technologies was developed to separate document data from presentation and to give developers the ability to extend the element sets of XML languages as needed.
- XML itself is **not** a language – it is a **meta language**. A meta language is a set of rules used for building markup languages.
- Structured languages can be developed that describe certain types of data rather than just the presentation of the data. Such structured languages include elements that describe documents containing information about an account, an item, a service, or a transaction.
- XHTML is an application of XML that is used for formatting Web documents. There are many other XML languages, some still under development, such as RSS (Really Simple Syndication), MathML, GraphML, Scalable Vector Graphics, and MusicXML.



Some Of The Benefits Of XML

- It allows data to be self-describing, as opposed to being limited by a pre-defined set of elements.
- You can provide rules for XML elements that limit the type of data an element can contain, such as only letters, only numbers, or a certain number of characters.
- Lets you create custom data structures for industry-specific or company-specific needs.
- Because XML describes data, you can present the data any number of ways by applying different presentation styles.
- It provides a rich set of tools for linking.
- You can use it to interchange data between proprietary formats and between databases or data structures.
- You can use the tools to defines a standard syntax for many markup languages.
- It has much more robust and reliable data searching capabilities that HTML.



Some Of The Benefits Of XML

- As its name implies, XHTML is extensible, meaning that its element set is not finite like the element set of HTML.
- Like any XML language, XHTML can be extended to add elements if needed or it can incorporate elements from various XML languages into its element set.
- Keep in mind, however, that if your goal is to keep XHTML compatible with current browsers (certainly one of our goals in this class), you may not be able to use some of its more advanced features. But if your documents are written in XHTML now, you will be able to integrate these features with ease as soon as they become mainstream.



XHTML Document Building Blocks

- Like any language, XHTML has a number of building blocks that are used to create complete documents.
 - The English language, for example, is made up of nouns, verbs, adjectives, adverbs, prepositions, and so on, which are used in conjunction with each other to form sentences and paragraphs.
- Both HTML and XHTML provide language building blocks that can be added to any text document. Web browsers know how to interpret these elements in order to present the document based on formatting rules.

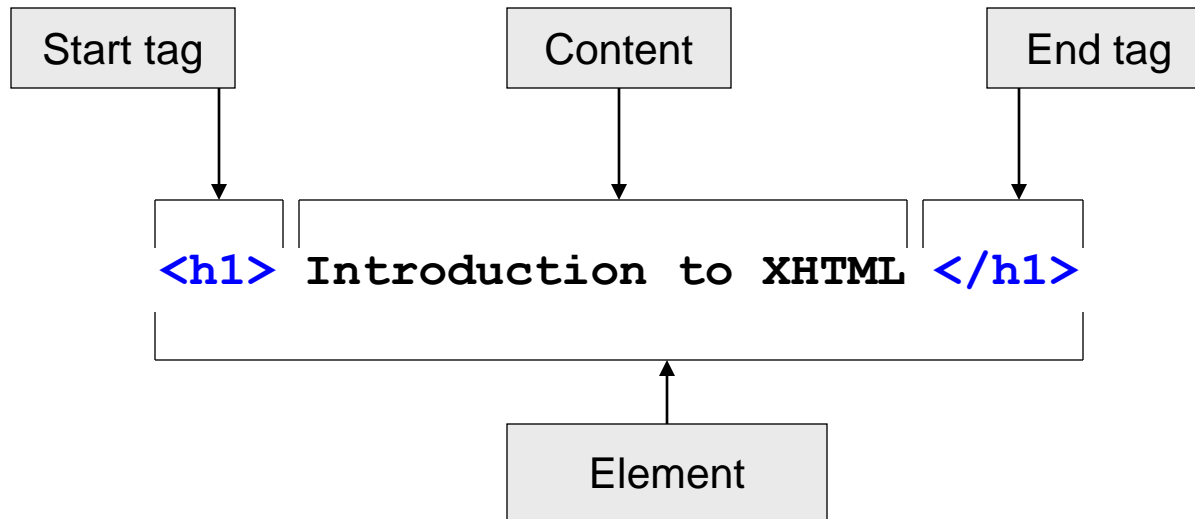


XHTML Elements

- XHTML **elements** are the core components of XHTML documents and are used to describe the data in a document.
 - Elements are like nouns in the English language.
- **Elements** are the **markup**, or formatting instructions, of the XHTML document.
- Elements define the text styles, formatting links, and other pieces of the document.
- Elements and **tags** are sometimes used interchangeably, but strictly speaking, a tag is a piece of an element.



XHTML Elements



- All elements, except for empty elements, consist of three pieces: a [start tag](#), [content](#), and an [end tag](#).
- XHTML element names must be written in lowercase letters.



Empty Elements

- Empty elements are used primarily to describe pieces of data that don't contain any content.
- For example, some common empty elements in HTML are:
 - `
` for line break
 - `` for image
 - `<p>` for paragraph
- In XML and XHTML, all elements must have a start tag and end tag.
- The XML specification provides a shortcut for writing an empty element using a single tag. This is shown below:

`
` for line break

`` for image

`<p />` for paragraph

NOTE: The practice of adding the extra space is **not** part of the syntax of either XML or XHTML. The reason for the space is to make this code compatible with older versions of Web browsers. Because HTML does not require end tags for these elements, most older browsers do not know how to handle the new empty element syntax. The extra space allows these browsers to interpret the syntax correctly.



Attributes

- XHTML **attributes** are pieces of information that help to describe elements. Some elements have required attributes, others are optional and depend on the content that is being marked up.
- Attributes are referred to as **name-value pairs** and have the following syntax: The name of the attribute is on the left, followed by an equal sign, then the value.

`name = "value"`

- Here are a few examples:

```
<a href="http://mark.com">Click here</a>
```

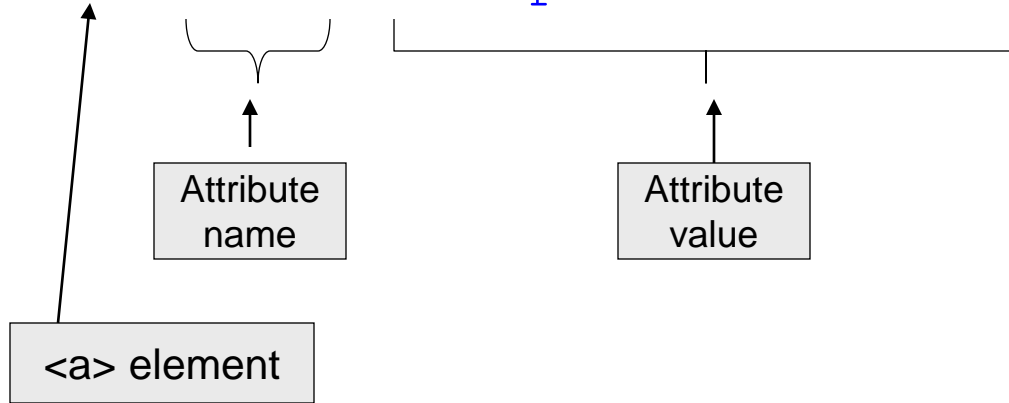
```

```



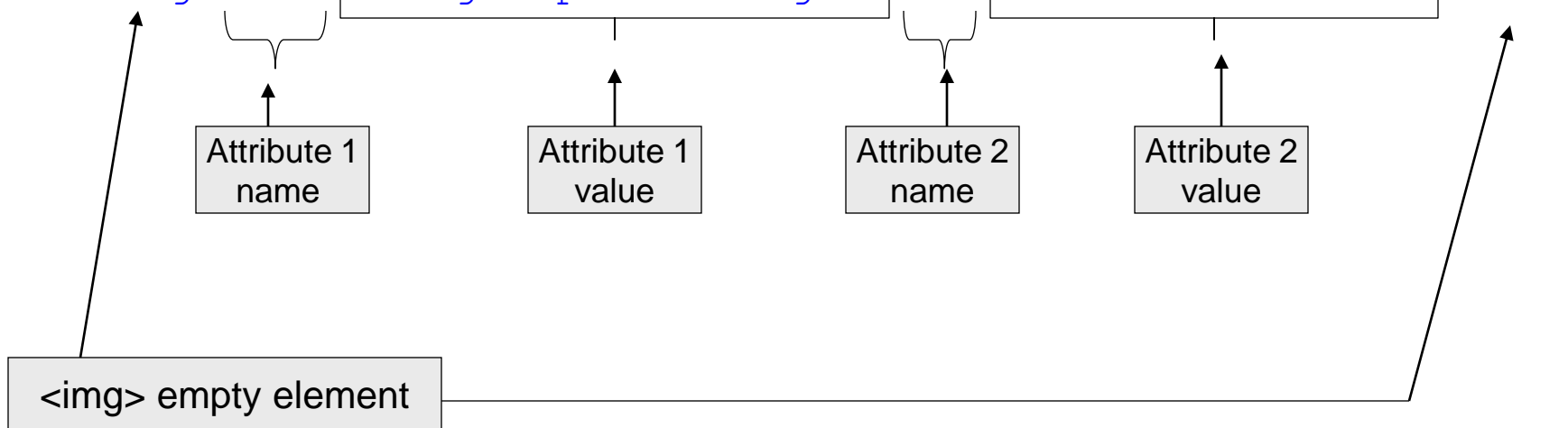
Attributes

```
<a href="http://mark.com">Click here</a>
```



```

```



Attribute Rules

- We'll see more rules later, but here are a few rules about XHTML attributes:
 1. Attributes are always contained within the start tag of an element.
 2. Names must be in lowercase letters.
 3. Attributes must have a value that is surrounded by quotes.



XHTML Core Attributes

- In the XHTML 1.0 specification, there are a set of **core attributes** that can be used with most XHTML elements.
- They cannot be used in `<base>`, `<head>`, `<html>`, `<meta>`, `<param>`, `<script>`, `<style>`, and `<title>` elements.
- The core attributes are:
 1. **id** – document wide unique id.
 2. **class** – list of classes of the element.
 3. **style** – associated style information.
 4. **title** – advisory title amplification.



XHTML Comments

- **Comments** in XHTML are notations that are ignored by programs and parsers.
- The syntax of an XHTML comment is identical to HTML and XML comments.
- You can use comments to document your code, add additional information about a piece of data, add visual breaks, or add information that other people working on or using your document would find useful.
- The following is an XHTML comment:

```
<!-- This is a comment -->
```



The XML Declaration

- The XML declaration defines which version of XML the document is using.
- Stating which version of XML is being used is important in order not to confuse parsers and programs as more versions of XML are developed.
- **The XML declaration always appears as the first line in an XML document. It cannot be preceded by any blank lines or white space.**
- The XML declaration tag begins with `<?xml` and ends with `?>`.
- The XML declaration can contain the following three attributes: `version`, `encoding`, and `standalone`.
- Although the XML declaration is optional in XML and XHTML, it is good practice to at least declare the `version` of XML being used:

```
<?xml version="1.0"?>
```



The XML Declaration

- The second piece of information that the declaration can contain is the **encoding** attribute, which defines the character set that the document uses.
- A character set is a grouping of characters, such as the Latin character set, a character set of symbols, or a character set of Greek letters.
- The encoding attribute is optional in the XML declaration, and if no character set is defined, the default is UTF-8, which is the 8-bit Unicode character encoding scheme.
 - All XML processors are required to handle UTF-8, which is why it is the default character set. (UTF is an acronym for Universal character set Transformation Format.)
- Other character sets include UTF-16, UTF-32, and ISO-10646-UCS-2. There are literally thousands of character sets available for use.



The XML Declaration

- The third piece of information that the declaration can contain is the **standalone** attribute. The value of this attribute must be either **yes** or **no**.
- This attribute is also optional, and the default value is **no**.
- This attribute tells the processor whether this document contains all of the pertinent information within itself or relies on external **Document Type Definitions (DTDs)** for its declarations.
- Setting the value to **yes** tells the processor that everything needed to process the document is within the document, and to ignore any references to external files. Setting the value to **no** tells the processor that the document can reference external files.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
```



The Three Flavors of XHTML 1.0

- The XHTML 1.0 specification comes in three “flavors” or versions. XHTML authors can choose the version that best fits the needs of their documents.
- This is done by inserting a **Document Type Definition (DOCTYPE) declaration**. This is part of the document prolog that we’ll look at more closely later.
- The DTD is the specification from the W3C that defines the language, including the elements and attributes. The declaration was optional for HTML documents, but is required for XHTML documents, and it needs to match one of the DTDs for the three flavors of XHTML.
- The following three pages define each of the three flavors and when they should be used, and shows the DOCTYPE declaration for each and where you can view the DTD on the Web.



XHTML Transitional Version

- XHTML Transitional is currently the most used version of XHTML 1.0. This version most closely resembles HTML 4.0.1 and is the best choice when documents need to use HTML's presentational elements or when pages need to be developed without using style sheets.
- Use this version if you want to convert existing HTML pages to XHTML.
- The caveat to the Transitional version is that it contains support for certain elements and attributes that are being deprecated, or phased out. It also does not contain support for frames.
- For more details on the Transitional flavor of XHTML see: <http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd>.
- The declaration is:

```
<!DOCTYPE html  
    PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```



XHTML Frameset Version

- XHTML Frameset should be used when your documents need to use the frame elements that partition the browser into multiple independent windows.
- The element set for this version contains all of the elements from XHTML Transitional plus the elements needed to support frames, such as `<frame>` and `<frameset>`.
- For more details on the Frameset flavor of XHTML see: <http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd>.
- The declaration is:

```
<!DOCTYPE html
```

```
    PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
```

```
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```



XHTML Strict Version

- XHTML Strict most closely represents the future of XHTML. The element set for XHTML Strict contains a subset of the elements from XHTML Transitional, but does not include support for strictly presentational elements or elements that will not likely be included in future versions of XHTML.
- In the future, XHTML documents will separate presentation from content and use style sheets to define presentation formatting such as font types, colors, and styles.
- Use XHTML Strict with Cascading Style Sheets (CSS). We will learn all about CSS later in the course.
- For more details on the Strict flavor of XHTML see: <http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd>.
- The declaration is:

```
<!DOCTYPE html  
    PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```



Creating Your First XHTML Document

- We'll begin creating our first XHTML document by examining a plain text document that we would like to markup with formatting elements.
- Using NotePad or some similar text editor, create a file with the name “[unformatted.html](#)”, that includes the text shown below (type it exactly as it appears below):

Course Name: Web Based Information Technology

Course Number: CIS 4004

Instructor: Dr. Mark Llewellyn

Class Meets: Monday, Wednesday and Friday, 11:30am – 12:20pm, HEC 118

Course Description: Digital libraries, Media formats, Compression, Streaming Media, Mobile Internet and WML, Emerging technologies. Capacity planning for web services.

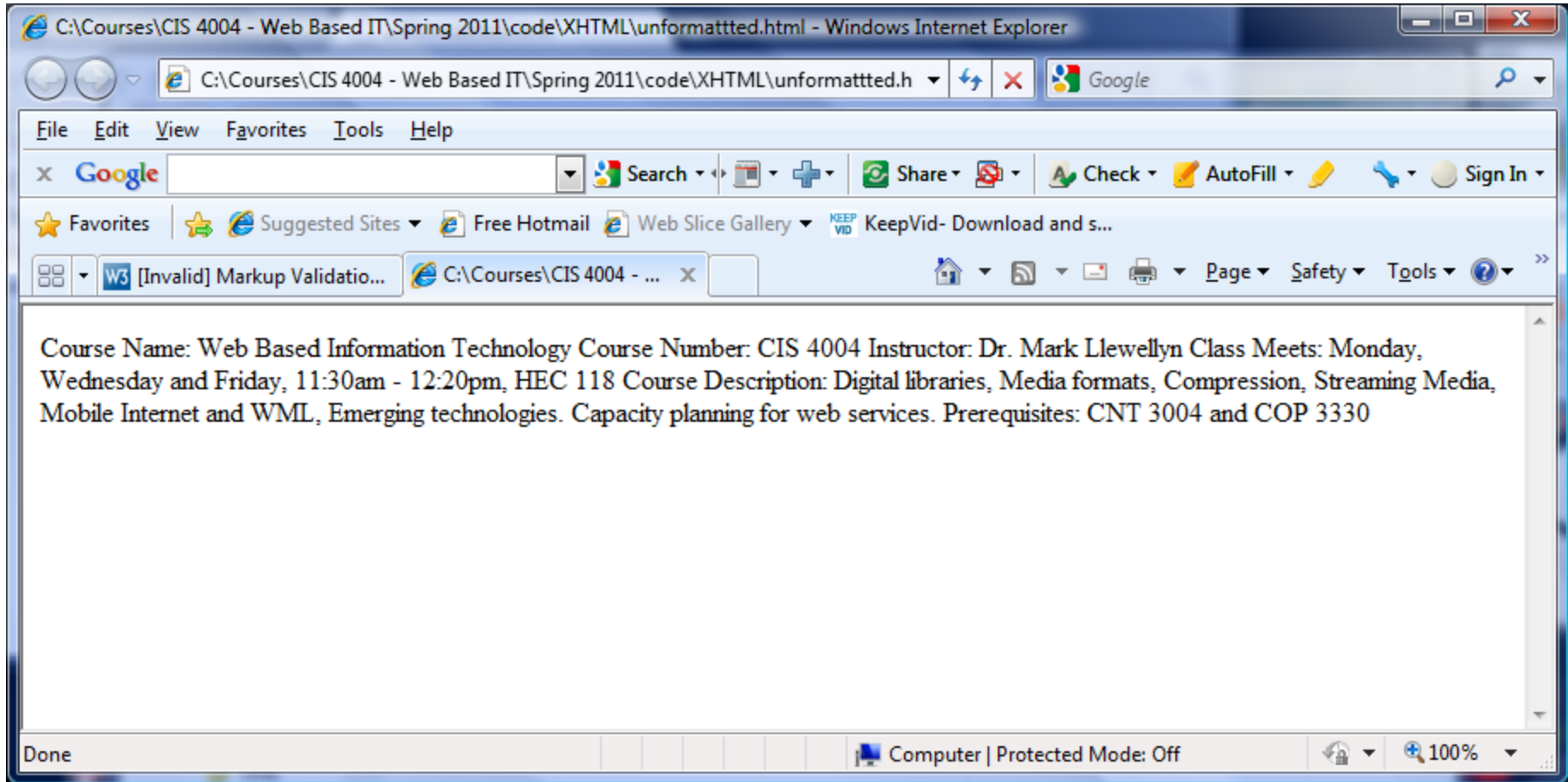
Prerequisites: CNT 4004 and COP 3330

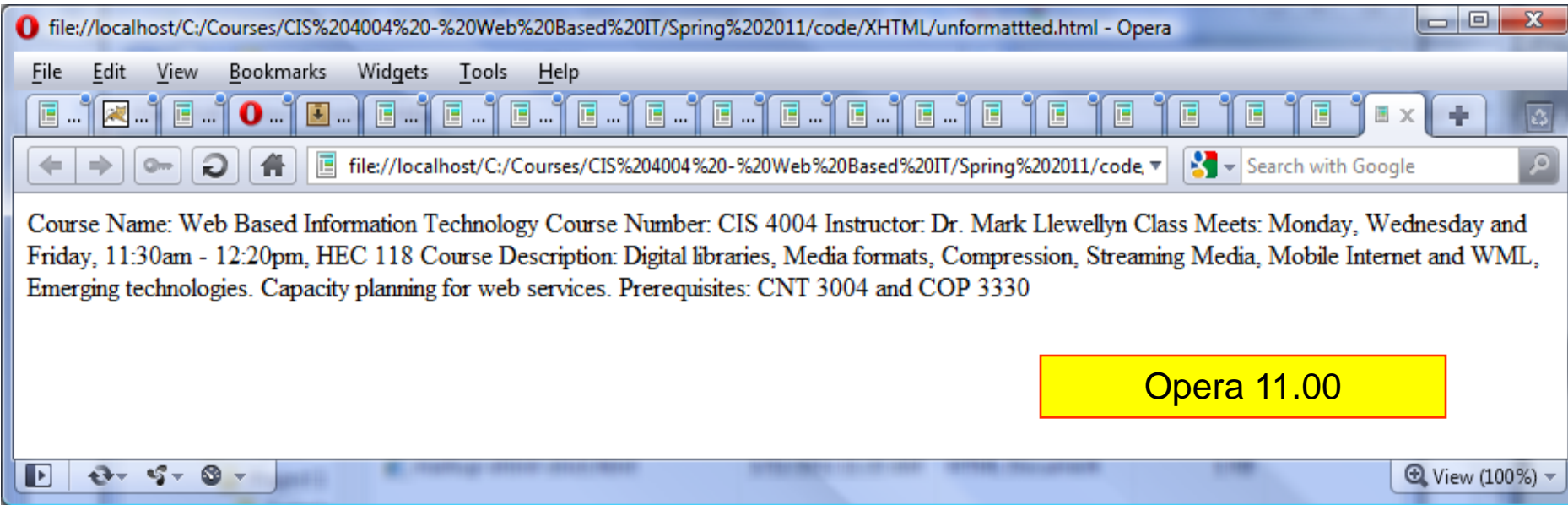
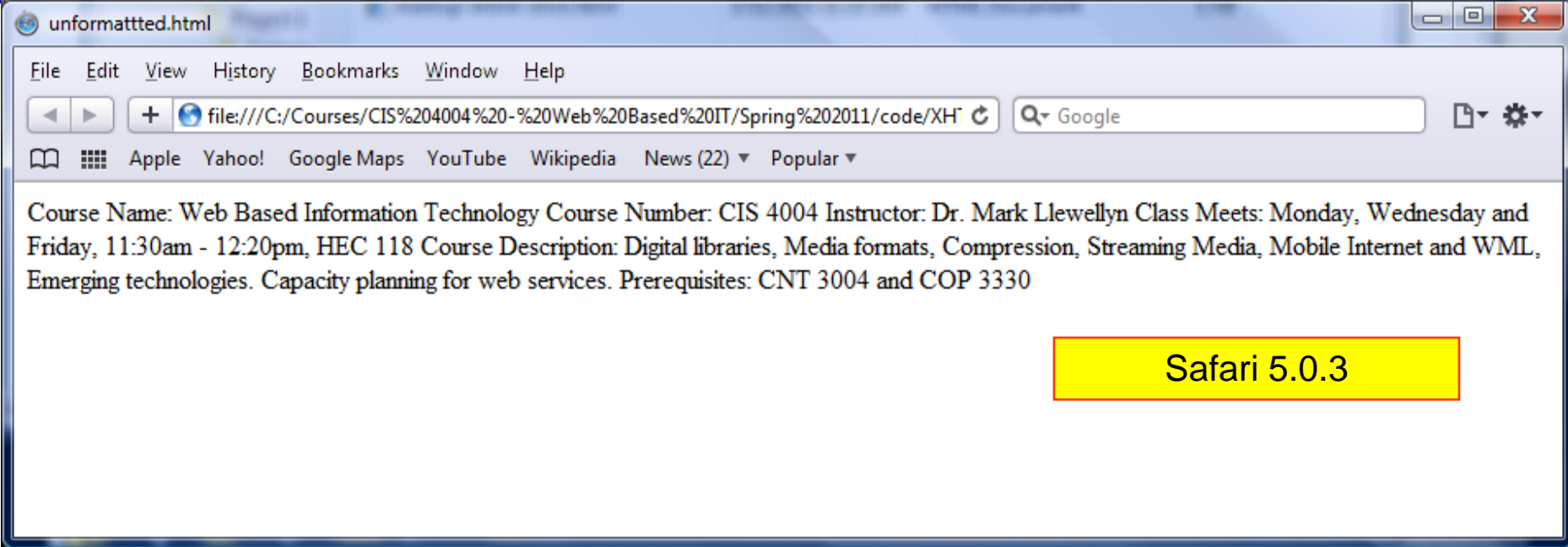
Unformatted Text Document – name it “[unformatted.html](#)”

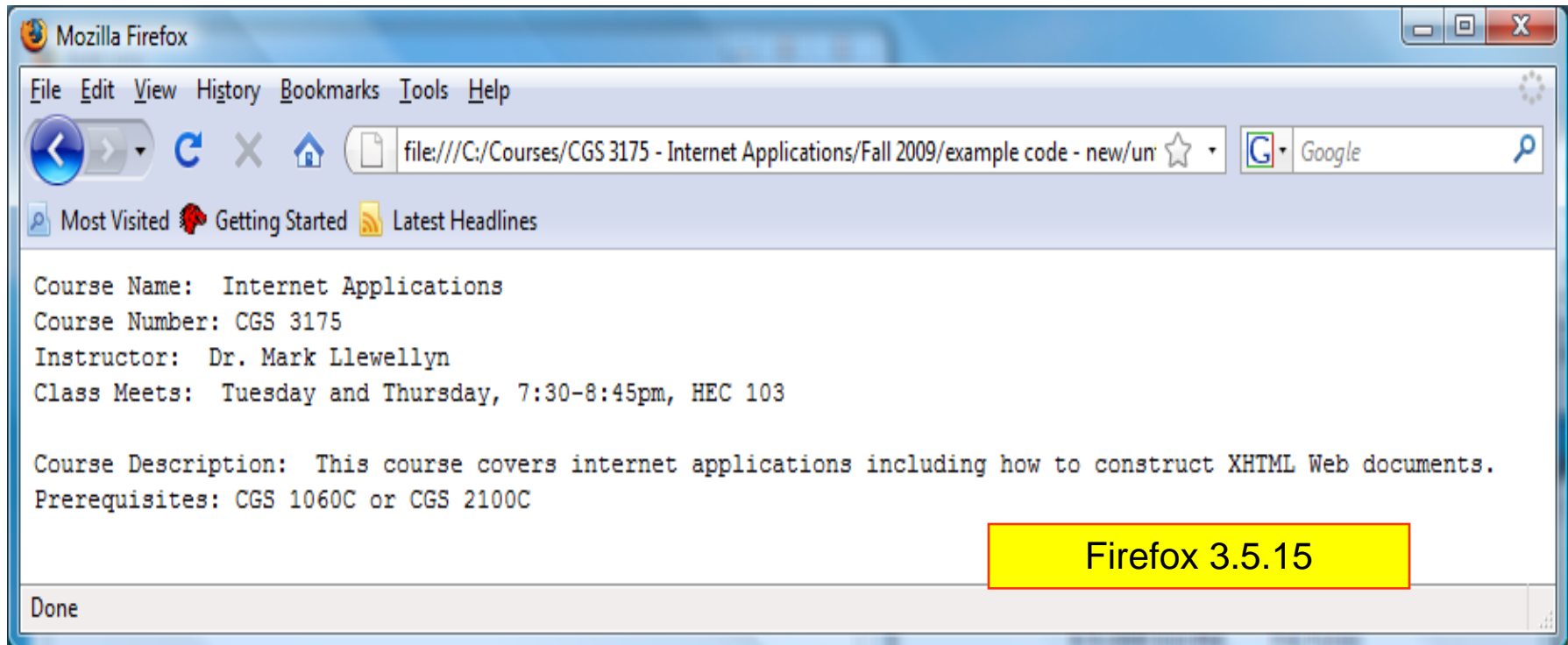


Creating Your First XHTML Document

- Next, open your Web browser and load the file “unformatted.html” that you just created. The screen shot below, shows what the file looks like in Internet Explorer. 8 The following two pages show the same file in several other popular browsers.







Creating Your First XHTML Document

- The screen shot on the previous page illustrates how the Web browser can open the document, but without formatting instructions, it does not know how to correctly format the document.
- Web browsers ignore all whitespace characters, including line breaks, so without the proper markup, our document displays as just a block of text.
- Next, we'll add some formatting elements (markup) to our document. For comparison purposes, we'll markup the document in both HTML and XHTML, and then look at the differences between the two.
- **NOTE:** Do not type the line numbers in your document, they are for only used in these notes to make references to specific lines.



The HTML Version

The document starts with an opening tag for the HTML element

This is followed by the header section and the title of the document

```
1. <HTML>
2.   <HEAD>
3.     <TITLE>Web Based Information Technology Spring 2011</TITLE>
4.   </HEAD>
5.   <BODY>
6.     <STRONG>Course Name: </STRONG> Web Based Information Technoloty <BR>
7.     <STRONG>Course Number: </STRONG> CIS 4004 <BR>
8.     <STRONG>Instructor: </STRONG> Dr. Mark Llewellyn <BR>
9.     <STRONG>Class Meets: </STRONG> Monday, Wednesday and Friday, 11:30am-12:20pm,
      HEC 118 <BR>
10.    <P>
11.    <STRONG>Course Description: </STRONG>
12.    <P>Digital libraries, Media formats, Compression, Streaming Media, Mobile
      Internet and WML, Emerging technologies. Capacity planning for web
      services.</P>
13.    <STRONG>Prerequisites: </STRONG>
14.    <UL>
15.      <LI> CNT 3004 and,
16.      <LI> COP 3330
17.    </UL>
18.  </BODY>
19. </HTML>
```

The main body of the document contains a mix of markup elements and content.

The document always ends with the closing tag for the HTML element



```
C:\Courses\CIS 4004 - Web Based IT\Spring 2011\code\XHTML\markup-html.html - Notepad++
File Edit Search View Format Language Settings Macro Run TextFX Plugins Window ?
markup-html.html
1 <HTML>
2   <HEAD>
3     <TITLE>Web Based Information Technology Spring 2011</TITLE>
4   </HEAD>
5   <BODY>
6     <STRONG>Course Name: </STRONG> Web Based Information Technology <BR>
7     <STRONG>Course Number: </STRONG> CIS 4004 <BR>
8     <STRONG>Instructor: </STRONG> Dr. Mark Llewellyn <BR>
9     <STRONG>Class Meets: </STRONG> Monday, Wednesday and Friday, 11:30am-12:20pm, HEC 118<BR>
10    <P>
11      <STRONG>Course Description: </STRONG> Digital libraries, Media formats, Compression, Streaming
12      Emerging technologies. Capacity planning for web services.
13    <P>
14    <STRONG>Prerequisites: </STRONG>
15    <UL>
16      <LI> CNT 3004 and,
17      <LI> COP 3330
18    </UL>
19  </BODY>
20 </HTML>
21
```

The HTML version as it appears in Notepad++

Hyper Text Markup Language nb char : 736 nb line : 21 Ln : 4 Col : 11 Sel : 0 UNIX ANSI INS



The screenshot shows the Amaya web editor interface. The main window displays a rendered HTML page with the following content:

Course Name: Web Based Information Technology
Course Number: CIS 4004
Instructor: Dr. Mark Llewellyn
Class Meets: Monday, Wednesday and Friday, 11:30am-12:20pm, HEC 118
Course Description: Digital libraries, Media formats, Compression, Streaming Media, Mobile Internet and WML, Emerging technologies. Capacity planning for web services.
Prerequisites:

- CNT 3004 and,
- COP 3330

The right-hand pane shows the 'Elements' and 'Style' panels. The 'Elements' panel includes a toolbar with icons for HTML, XML, and various text and image elements. The 'Style' panel shows a 'Theme' dropdown set to 'No theme', a font family dropdown set to 'Arial', and a font size dropdown set to '12'. Below the 'Style' panel is the 'Apply class' section, which currently shows '(no class)'. The status bar at the bottom of the window displays 'Finished!' and 'Text'.

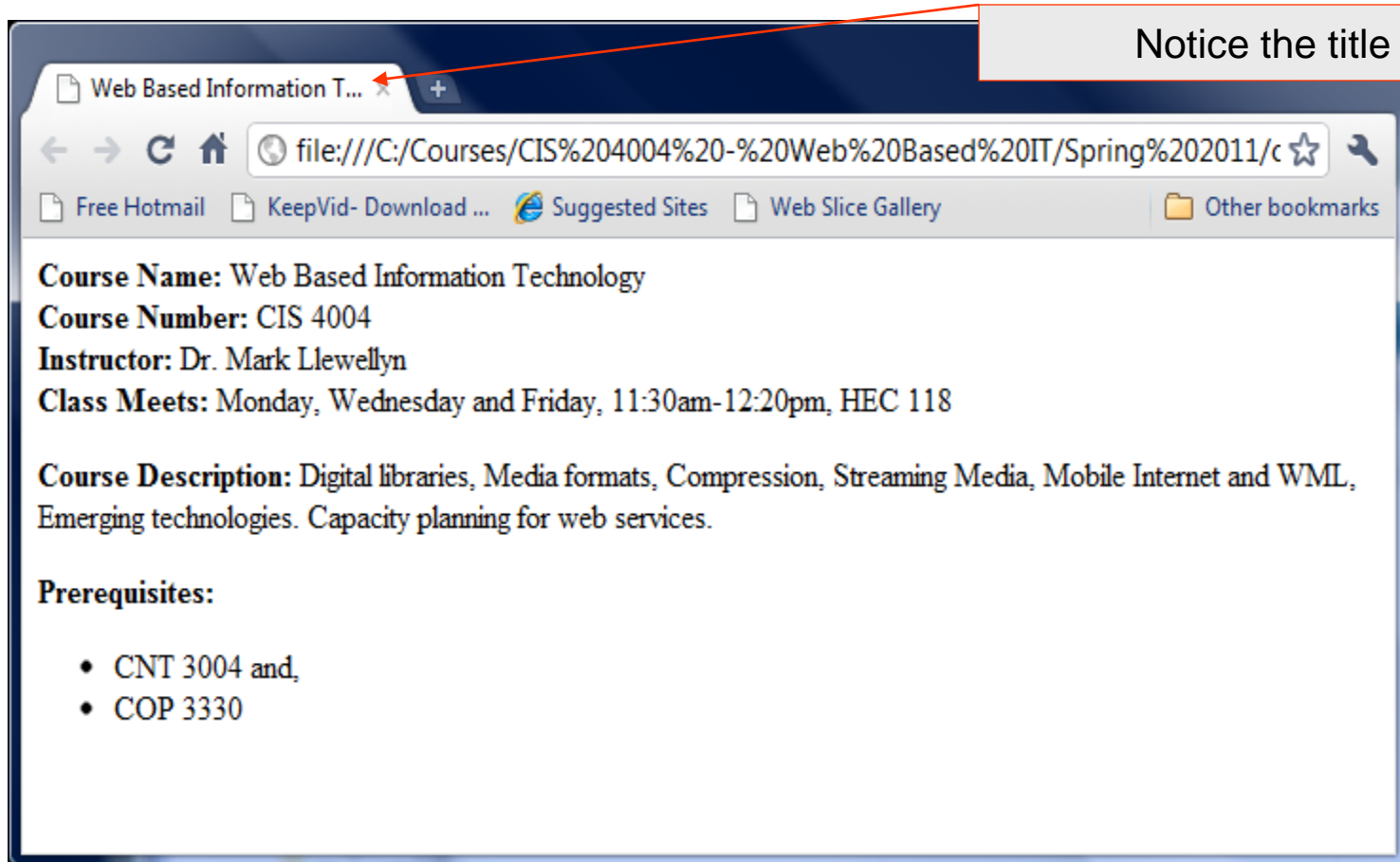
The URL in the address bar is: C:\Courses\CIS 4004 - Web Based IT\Spring 2011\code\XHTML\markup-html.html

The HTML version as it appears in Amaya with split screen showing both rendered and source versions.



Viewing The HTML Version

- Create a new file that contains the text from the previous page (remember – do not type in the line numbers). I called my version of this file: “markup-html.html”, but you can call it whatever you would like. Next, open your Web browser and load the file “markup-html.html” that you just created. The screen shot below, shows what the file looks like in Chrome.



The XHTML Version (Transitional)

Begin with XHTML document heading

```
1. <?xml version="1.0" encoding="UTF-8" standalone="no" ?>
2. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3.     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4. <html xmlns="http://w3.org/199/xhtml">
5.   <head>
6.     <title>Web Based Information Technology Spring 2011
7.   </head>
```

The document starts with an opening tag for the <html> element.

```
8.   <body>
9.     <strong>Course Name: </strong> Web Based Information Technology <br />
10.    <strong>Course Number: </strong> CIS 4004 <br />
11.    <strong>Instructor: </strong> Dr. Mark Llewellyn <br />
12.    <strong>Class Meets: </strong> Monday, Wednesday and Friday, 11:30am - 12:20pm, HEC 118 <br />
13.    <p />
14.    <strong>Course Description: </strong>Digital libraries, Media formats, Compression, Streaming
15.    Media, obile Internet and WML, Emerging technologies. Capacity planning for web
16.    services.
17.    <p />
18.    <strong>Prerequisites: </strong>
19.    <ul>
20.      <li> CNT 3004 and,</li>
21.      <li> COP 3330 </li>
22.    </ul>
23.  </body>
24. </html>
```

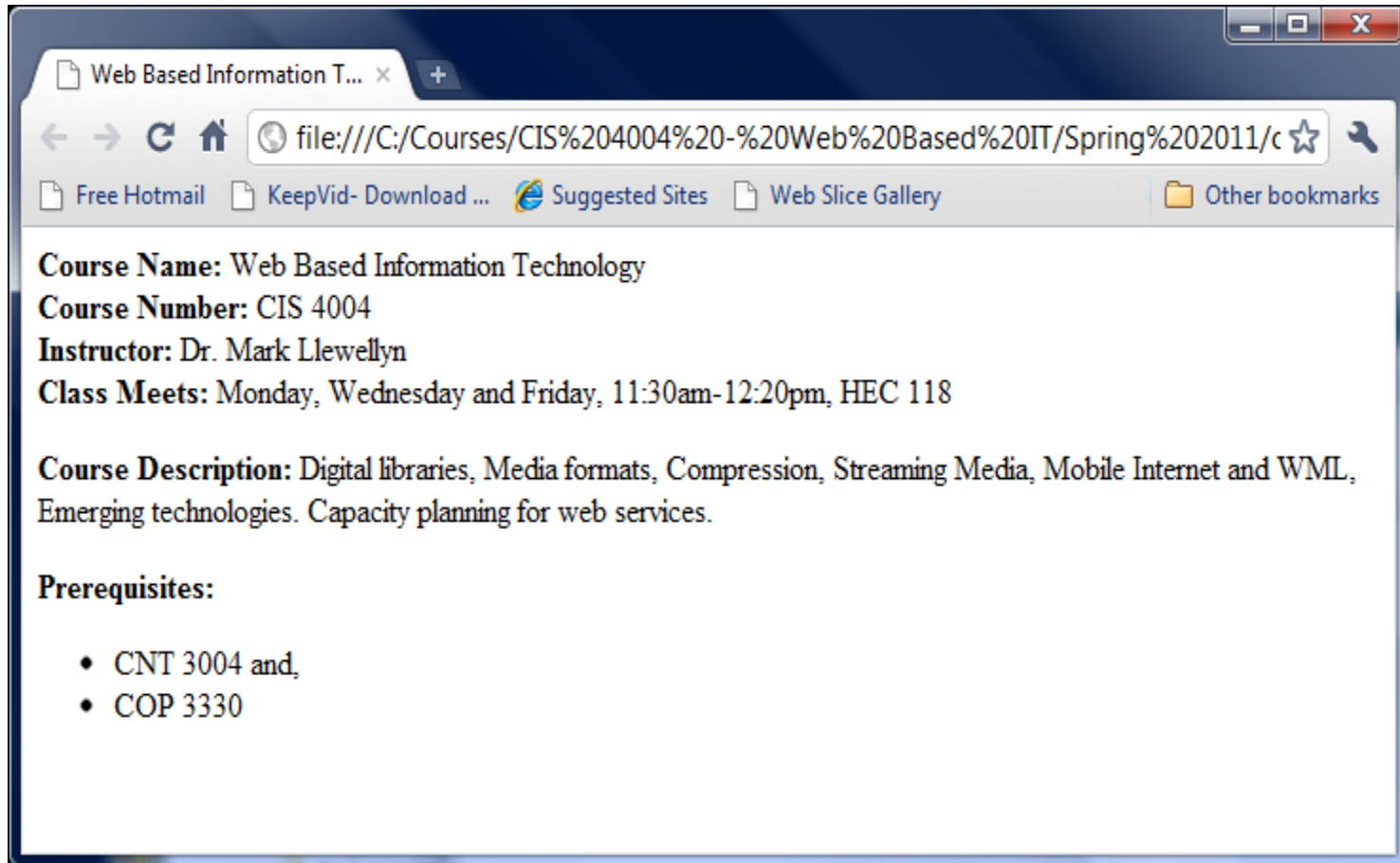
The main body of the document contains a mix of markup elements and content.

The document ends with a closing tag for the <html> element



Viewing The XHTML Version

- Create a new file that contains the text from the previous page (remember – do not type in the line numbers). I called my version of this file: “markup-xhtml.html”, but you can call it whatever you would like. Next, open your Web browser and load the file “markup-xhtml.html” that you just created. The screen shot below, shows what the file looks like in Chrome .



Differences Between HTML And XHTML

- Looking at the two examples on pages 37 and 42, you probably noticed that the code is very similar. Both HTML and XHTML are platform- and vendor-independent. Both used elements and tags to describe pieces of data within a document.
- As we mentioned earlier, they use the same set of element names based on the element set from HTML 4. However, XHTML has much stricter syntax and is more powerful and flexible than HTML.
- The following three pages list some of the major differences between XHTML and HTML. We'll explore all of these differences and many more as the semester progresses.



Differences Between HTML And XHTML

1. XHTML documents contain the XML and DOCTYPE declarations at the top of the document. The XML declaration is optional but the DOCTYPE declaration is required. The DOCTYPE declaration was optional in HTML. These are lines 1 and 2 in the XHTML document on page 42.
2. XHTML documents must be well formed, meaning that they need to adhere to the syntax rules for the language. HTML, however, does not strictly require that documents be well formed.
3. XHTML is not dependent on a single document type or set of markup elements, like HTML. XHTML can be extended or used in conjunction with other markup languages.



Differences Between HTML And XHTML

4. XHTML element and attribute names must be lowercase. XHTML elements and attributes are case sensitive, while HTML elements and attributes are not. In our examples on pages 37 and 42, notice that the HTML elements are all uppercase: `<HTML>`, ``, `</BODY>`. This was done simply as a matter of style. These tags could have been written in lowercase or a combination of upper and lowercase, and HTML would still have interpreted them correctly: `<html>`, ``, `</Body>`. The XHTML document, on the other hand, must have all of its tags and attributes in lowercase.
5. For nonempty elements, XHTML requires end tags. An empty element is an element that does not contain an end tag. This is not a requirement for HTML, as the HTML element set contains a subset of elements that do not have end tags. In the XHTML code on page 42, notice that these elements are written a little differently: `
` and `<p />`, In XHTML, all elements must either have an end tag or end in `/>`.



Differences Between HTML And XHTML

6. Attribute values must always be quoted in XHTML. This was not a requirement in HTML. The following is valid in HTML:

```
<img src=picture.gif>
```

The attribute src has a value of picture.gif assigned to it. However, the same line in an XHTML document places quotes around the value of the attribute:

```

```

- The W3C provides a great deal of information about the differences between HTML and XHTML. See <http://www.w3.org/TR/xhtml1/#diffs>



Rules and Tools For Building XHTML Documents

- Because XHTML is an XML application, you must adhere to the same syntax rules as any XML language.
- If you are used to writing HTML code, these rules may seem a bit intimidating, as HTML does not force developers to adhere so strictly to syntax rules. If you are new to writing markup code, that's fine...you'll be learning the correct way to markup code from the start.
- An XHTML document that adheres to XML syntax is said to be **well-formed**. A document that is not well-formed will generate an error in a parser program.



Syntax Rules For Well-formed XML Documents

1. All XHTML documents must contain the root element `<html>` and cannot contain more than one root element.
2. All elements must have a start and end tag, such as:

```
<h1> . . . </h1>
```

The exception is an empty tag, which must have a forward slash (/) before the end tag. An example of the shorthand notation is: `
`.

3. Elements must be nested properly and cannot overlap. Each element must be contained completely inside of its parent element. This rule is the same as for mathematical functions such as: $(x * [y + z])$. Notice that there are two parts to this equation, one is surrounded by the parentheses and the other by brackets. The subequation $[y + z]$ is entirely contained within the outer equation, which is delineated by the parentheses. It is illegal to write the equation like this: $(x * [y + z])$.

Example of illegally nested elements:

```
<h1> <strong> . . .</h1> </strong>
```

Example of legally nested elements:

```
<h1> <strong>. . . </strong> </h1>
```



Syntax Rules For Well-formed XML Documents

4. All attributes must have a value, and that value must be enclosed in quotes. Both double and single quotes are allowed and can be nested, as long as you are consistent and you do not use the same type of quote twice. Here are some valid examples:

Single quotes: ``

Double quotes: ``

Nested quotes:

```

```

5. Attributes must be placed in the start tag of an element, and no attribute can appear more than once. Here is an invalid example:

```
<a href="page2.html" href="page3.html"> Click here</a>
```

6. Element names are case sensitive, for example, `<TITLE>` and `</title>` are both legal beginning and ending tags in HTML, because it is not case sensitive. In XHTML, all tags must be lowercase, so `<TITLE>` would produce an error.



Well-Formed Versus Valid Documents

- The terms well-formed and valid in reference to writing good XHTML are not the same. They are both very important pieces of developing code that will be compatible with future tools, but they are distinctly different in the kinds of error they test for.
- A well-formed document, as we've just seen, must adhere to a set of structural rules. These rules apply to the syntax of how the code is written, but does not pay attention to the code itself.
 - For example, if you include the names of elements or attributes that are not part of the HTML 4.01 specification, but you follow the syntax rules we just covered, your document will pass the well-formed test, even though you have used elements that are not part of the specification.
- For a document to be valid, it must not only be well-formed, but also conform to the rules of the DTD that it lists in its DOCTYPE declaration. The DTD contains the set of all valid element and attribute names that can be used for that document type. Recall the XHTML Strict DTD:

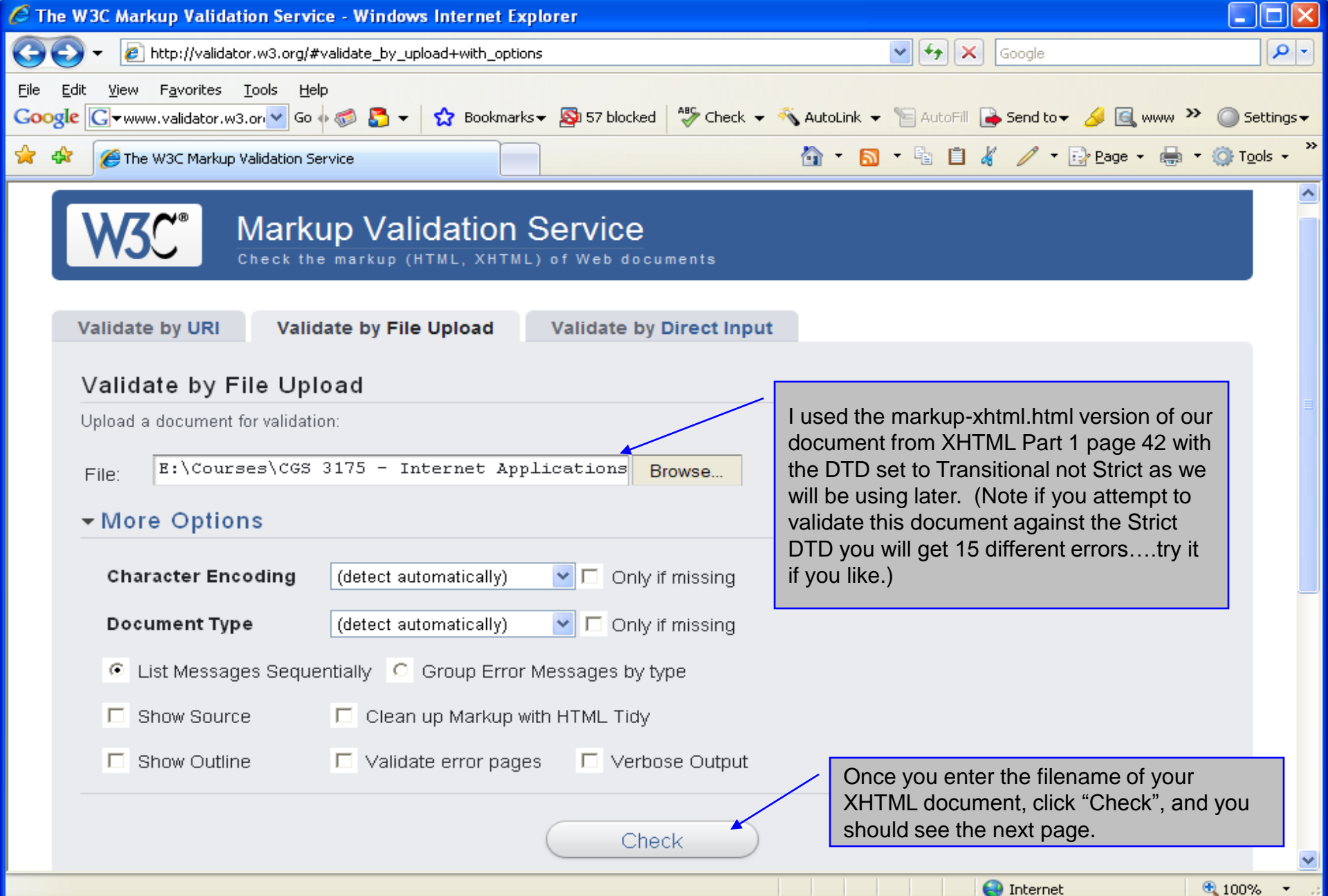
```
<!DOCTYPE html  
    PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```



Validating An XHTML Document

- While there are many free and commercial programs available, one of the simplest ways to check if your XHTML documents are well-formed and valid is to use the free on-line validating tool from the W3C Web site. It can check your document either from your computer or from a website.
- The W3C markup validation service is available at: <http://validator.w3.org>.
- The next page shows the first step in using the file upload version of the validator.





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:

More Options

Character Encoding: Only if missing

Document Type: 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

I used the markup-xhtml.html version of our document from XHTML Part 1 page 42 with the DTD set to Transitional not Strict as we will be using later. (Note if you attempt to validate this document against the Strict DTD you will get 15 different errors....try it if you like.)

Once you enter the filename of your XHTML document, click "Check", and you should see the next page.



[Valid] Markup Validation of markup-xhtml.html - W3C Markup Validator - Windows Internet Explorer

http://validator.w3.org/check

File Edit View Favorites Tools Help

Google Search Share Check AutoFill Sign In

W3C [Valid] Markup Validation of markup-xhtml.html ...

W3C[®] Markup Validation Service
Check the markup (HTML, XHTML, ...) of Web documents

Jump To: [Congratulations](#) - [Icons](#)

This document was successfully checked as XHTML 1.0 Transitional!

Result:	Passed	
File :	<input type="text"/>	<input type="button" value="Browse..."/>
	<i>Use the file selection box above if you wish to re-validate the uploaded file markup-xhtml.html</i>	
Encoding :	utf-8	(detect automatically) ▾
Doctype :	XHTML 1.0 Transitional	(detect automatically) ▾

Done Internet | Protected Mode: Off 100%



XHTML Code With Errors – Can You Find Them?

```
<?xml version="1.0" encoding="UTF-8 standalone="no" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Web Based Information Technology Spring 2011
  </head>
  <body>
    <strong>Course Name: </strong> Web Based Information Technology <br />
    <strong>Course Number: </strong> CIS 4004 <br />
    <strong>Instructor: </strong> Dr. Mark Llewellyn <br />
    <strong>Class Meets: </strong> Monday, Wednesday and Friday, 11:30am -
12:20pm, HEC 118<BR />
    <p />
    <strong>Course Description: </strong>Digital libraries, Media formats,
    Compression, Streaming Media, Mobile Internet and WML, Emerging
    technologies. Capacity planning for web services.
    <p />
    <strong attribute="yes">Prerequisites: </strong>
    <ul>
      <li> CNT 3004 and,</li>
      <li> COP 3330</li>
    </ul>
  </body>
</html>
```

Missing end tag </title>

end tag is capitalized

No attribute named "attribute"



[Invalid] Markup Validation of markup-xhtml - with errors.html - W3C Markup Validator - Windows Internet Explorer

http://validator.w3.org/check

File Edit View Favorites Tools Help

Google Search Share Check AutoFill Sign In

W3C [Invalid] Markup Validation of markup-xhtml - wi...

W3C[®] Markup Validation Service
Check the markup (HTML, XHTML, ...) of Web documents

Jump To: [Validation Output](#)

Errors found while checking this document as XHTML 1.0 Transitional!

Result:	3 Errors	
File :	<input type="text"/>	<input type="button" value="Browse..."/>
<i>Use the file selection box above if you wish to re-validate the uploaded file markup-xhtml - with errors.html</i>		
Encoding :	utf-8	(detect automatically) ▾
Doctype :	XHTML 1.0 Transitional	(detect automatically) ▾

Done Internet | Protected Mode: Off 100%



[Invalid] Markup Validation of markup-xhtml - with errors.html - W3C Markup Validator - Windows Internet Explorer

http://validator.w3.org/check

File Edit View Favorites Tools Help

Google Search Share Check AutoFill Sign In

W3 [Invalid] Markup Validation of markup-xhtml - wi...

✘ Line 7, Column 10: end tag for "title" omitted, but OMITTAG NO was specified

```
</head>
```

You may have neglected to close an element, or perhaps you meant to "self-close" an element, that is, ending it with "/>" instead of ">".

◆ Line 6, Column 6: start tag was here

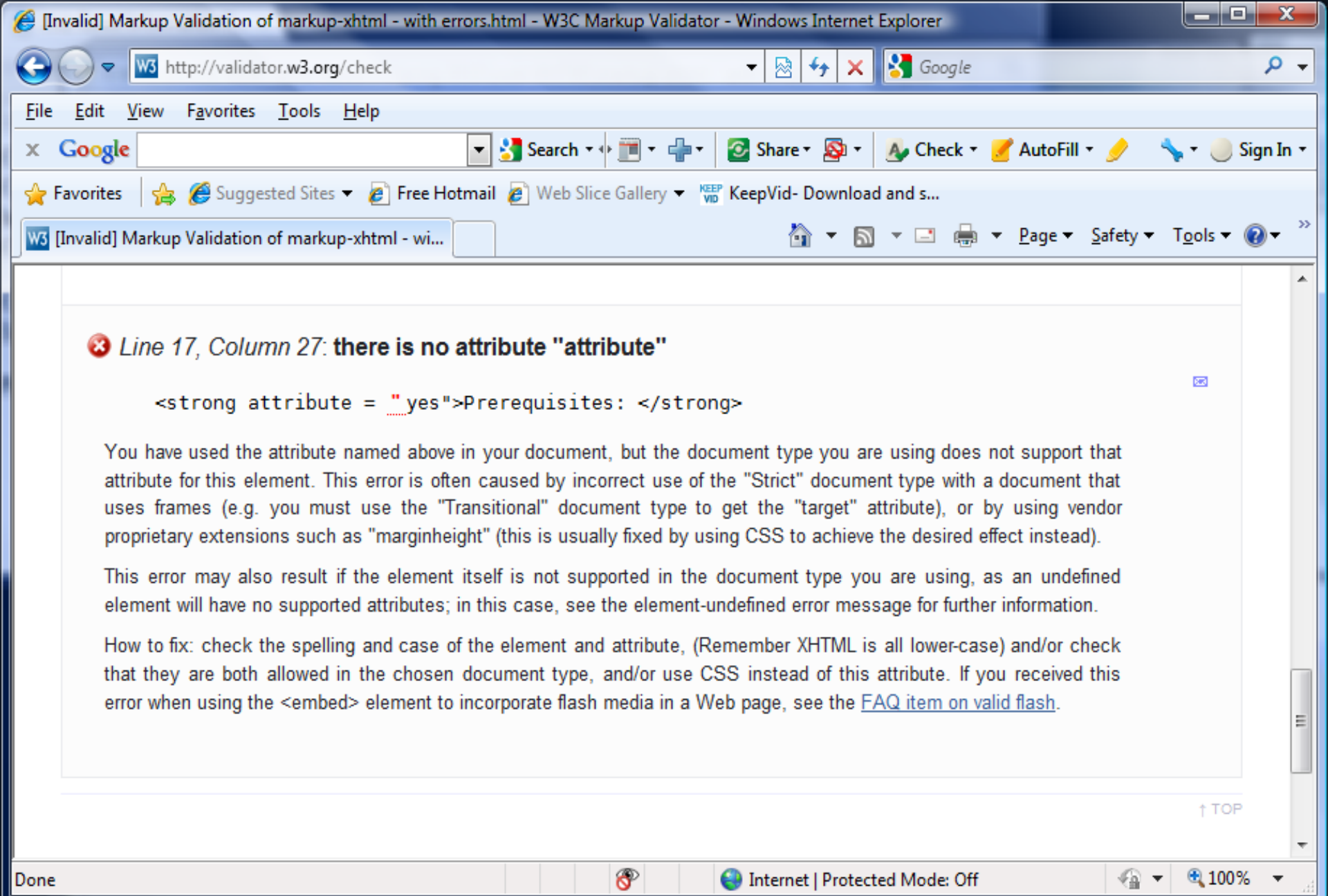
```
<title>Web Based Information Technology Spring 2011
```

✘ Line 12, Column 98: element "BR" undefined

```
... Meets: </strong> Monday, Wednesday and Friday, 11:30am-12:20pm, HEC 118 <BR />
```

Done Internet | Protected Mode: Off 100%





✖ Line 17, Column 27: there is no attribute "attribute"

```
<strong attribute = "yes">Prerequisites: </strong>
```

You have used the attribute named above in your document, but the document type you are using does not support that attribute for this element. This error is often caused by incorrect use of the "Strict" document type with a document that uses frames (e.g. you must use the "Transitional" document type to get the "target" attribute), or by using vendor proprietary extensions such as "marginheight" (this is usually fixed by using CSS to achieve the desired effect instead).

This error may also result if the element itself is not supported in the document type you are using, as an undefined element will have no supported attributes; in this case, see the element-undefined error message for further information.

How to fix: check the spelling and case of the element and attribute, (Remember XHTML is all lower-case) and/or check that they are both allowed in the chosen document type, and/or use CSS instead of this attribute. If you received this error when using the <embed> element to incorporate flash media in a Web page, see the [FAQ item on valid flash](#).

↑ TOP



```
C:\Courses\CIS 4004 - Web Based IT\Spring 2011\code\XHTML\markup-xhtml-strict.html - Notepad++
File Edit Search View Format Language Settings Macro Run TextFX Plugins Window ?
markup-html.html markup-xhtml.html markup-xhtml - with errors.html markup-xhtml-strict.html
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4 <html xmlns="http://www.w3.org/1999/xhtml">
5 <head>
6 <title>Web Based Information Technology Spring 2011</title>
7 </head>
8 <body>
9 <div>
10 <strong>Course Name: </strong> Web Based Information Technology<br />
11 <strong>Course Number: </strong> CIS 4004 <br />
12 <strong>Instructor: </strong> Dr. Mark Llewellyn <br />
13 <strong>Class Meets: </strong> Monday, Wednesday and Friday, 11:30am-12:20pm, HEC 118 <br />
14 <p />
15 <strong>Course Description: </strong> Digital libraries, Media formats, Compression, Stream:
16 Mobile Internet and WML, Emerging technologies. Capacity planning for web services.
17 <p />
18 <strong>Prerequisites: </strong>
19 <ul>
20 <li> CNT 3004 and,</li>
21 <li> COP 3330</li>
22 </ul>
23 </div>
24 </body>
25 </html>
26
Hyper Text Markup Language nb char : 989 nb line : 27 Ln : 23 Col : 11 Sel : 0 Dos\Windows ANSI INS
```

Note: Strict DTD

Note: <div> block required

A XHTML Strict DTD version



```
C:\Courses\CIS 4004 - Web Based IT\Spring 2011\code\XHTML\markup-xhtml-strict - version 2.html - Notepad++
File Edit Search View Format Language Settings Macro Run TextFX Plugins Window ?
markup-html.html markup-xhtml.html markup-xhtml - with errors.html markup-xhtml-strict.html markup-xhtml-strict - version 2.html
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4 <html xmlns="http://www.w3.org/1999/xhtml">
5 <head>
6 <title>Web Based Information Technology Spring 2011</title>
7 </head>
8 <body>
9 <div>
10 <span style="font-weight:bold">Course Name: </span> Web Based Information Technology <br />
11 <span style="font-weight:bold">Course Number: </span> CIS 4004 <br />
12 <span style="font-weight:bold">Instructor: </span> Dr. Mark Llewellyn <br />
13 <span style="font-weight:bold">Class Meets: </span> Monday, Wednesday and Friday, 11:30am -
14 <p />
15 <span style="font-weight:bold">Course Description: </span> Digital libraries, Media formats
16 Mobile Internet and WML, Emerging technologies. Capacity planning for web services.
17 <p />
18 <span style="font-weight:bold">Prerequisites: </span>
19 <ul>
20 <li> CNT 3004 and,</li>
21 <li> COP 3330</li>
22 </ul>
23 </div>
24 </body>
25 </html>
26
Hyper Text Markup Language nb char : 1114 nb line : 27 Ln : 25 Col : 8 Sel : 0 UNIX ANSI INS
```

Note: Strict DTD

In-line CSS – much more later

An alternate version of the XHTML Strict DTD version using in-line CSS.



Course Name: Web Based Information Technology

Course Number: CIS 4004

Instructor: Dr. Mark Llewellyn

Class Meets: Monday, Wednesday and Friday, 11:30am - 12:20pm, HEC 118

Course Description: Digital libraries, Media formats, Compression, Streaming Media, Mobile Internet and WML, Emerging technologies. Capacity planning for web services.

Prerequisites:

- CNT 3004 and,
- COP 3330

Chrome rendering of the XHTML Strict DTD version



A Few Words On Text Editors

- Creating an XHTML document can be done in any text editor, such as Notepad.
- If you must use a word processing program, be sure that it allows you to enter text only documents and be sure that you set the program for this mode when entering XHTML documents. You do not want hidden word processing characters to appear in your XHTML documents because most browsers will not be able to correctly interpret these codes. This will lead to questionable display characteristics.
- Notepad++ is a good text editor that was developed to be syntax friendly for many different programming/markup languages including XHTML. You can download Notepad++ at: <http://notepad-plus.sourceforge.net/uk/site.htm>.
- The W3 organization has also developed a web-friendly editor called Amaya. The current stable version of Amaya is 11.3.1. You can download Amaya at www.w3c.org/Amaya. The Amaya environment is a graphical one that may take a bit of getting used to and many of its features will not become apparent until you begin to work on more complex XHTML documents. We'll revisit Amaya later in the semester.



Windows Internet Explorer window showing the Notepad++ homepage. The address bar displays `http://notepad-plus.sourceforge.net/uk/site.htm`. The browser interface includes a menu bar (File, Edit, View, Favorites, Tools, Help), a search bar with "notepad++" entered, and a toolbar with buttons for Search, Bookmarks, and Check. A notification box indicates "2 pop-ups blocked" and provides instructions to allow pop-ups by pressing 'Ctrl' while clicking.

The homepage content features a green background with a lizard image. The main heading is "made with Notepad++" with the subtext "The latest version : 5.4.5". A navigation menu includes links for "About NP++", "News", "Download", "Screenshots", "Shop", and "Links". The "About Notepad++" section contains the following text:

Notepad++ is a free (as in "free speech" and also as in "free beer") source code editor and Notepad replacement that supports several languages. Running in the MS Windows environment, its use is governed by [GPL License](#).

Based on a powerful editing component [Scintilla](#), Notepad++ is written in C++ and uses pure Win32 API and STL which ensures a higher execution speed and smaller program size. By optimizing as many routines as possible without losing user friendliness, Notepad++ is trying to reduce the world carbon dioxide emissions. When using less CPU power, the PC can throttle down and reduce power consumption, resulting in a greener environment.

This project is mature. However, there may be still some bugs and missing features that are being worked on. If you have any questions or suggestions about this project, please post them in the [forums](#). Also, if you wish to make a feature request, you can post it there as well. But there's no guarantee that I'll implement your request.

You're encouraged to [translate Notepad++](#) into your native tongue if there's not already a translation present in the [Download Section](#). And if you want, help [translating Notepad++ official site](#) into your native tongue would be greatly appreciated.

I hope you enjoy Notepad++ as much as I enjoy coding it.

Here are the features of Notepad++:

A yellow box in the bottom right corner of the page contains the text "Notepad++ Homepage".



Amaya Home Page - Windows Internet Explorer

http://www.w3.org/Amaya/

File Edit View Favorites Tools Help

Google amaya Search Share Check AutoFill Sign In

Favorites Suggested Sites Free Hotmail Web Slice Gallery KeepVid- Download an

W3 [Valid] Markup Valid... C:\Courses\CIS 4004 ... Amaya Home Page X

Amaya Homepage showing location of download

Welcome to Amaya

Translations: [be](#), [da](#), [de](#), [es](#), [hi](#), [is](#), [ja](#), [kr](#), [pl](#), [ru](#), [sq](#), [sr](#), [th](#), [uk](#), [vn](#), [zh-hans](#), [zh-hant](#)

W3C's Editor/Browser

Amaya is a Web editor, i.e. a tool used to create and update documents directly on the Web. Browsing features are seamlessly integrated with the editing and remote access features in a uniform environment. This follows the original vision of the Web as a space for collaboration and not just a one-way publishing medium.

Work on Amaya started at W3C in 1996 to showcase Web technologies in a fully-featured Web client. The main motivation for developing Amaya was to provide a framework that can integrate as many W3C technologies as possible. It is used to demonstrate these technologies in action while taking

Done Internet | Protected Mode: Off 100%



Amaya [11]

[Web Site](#)[W3C](#)[INRIA](#)[Palette](#)

Amaya is a Web client that acts both as a browser and as an authoring tool. It has been designed by [W3C](#) and [INRIA](#) with the primary purpose of demonstrating new Web technologies and helping users to generate valid Web pages. Thanks to the European (FP6) [Palette](#) project, Amaya 11 is much easier to use, through its new user interface and its innovative templating system.

With Amaya, you can manipulate rich Web pages containing forms, tables, and the most advanced features from XHTML. You can create and edit complex [mathematical expressions](#) and [graphics](#) within Web pages. You can style your documents using [Cascading Style Sheets](#) (CSS).

Main new features

Major changes since version 10:

- Possibility to create and edit document templates from document skeletons
- More facilities to edit template instances
- The first version of an integrated editor for SVG graphic schemas
- The support of semantic information

Did you know?

Choose your profile

Amaya can be adapted to user's preferences: 5 different editing profiles are provided to adapt the menus and panels to your own needs. Panels can be displayed on the left or on the right side of the window, and each panel can be customized (see Preferences).

Amaya is a structured editor

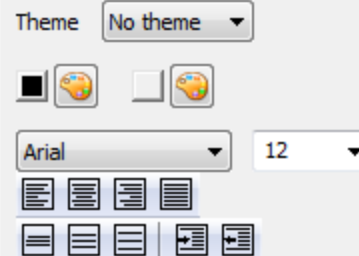
With the F2 or Fsc key you can select the parent element



Elements



Style



Apply class

(no class)
 .body
 .bottom
 .column
 .main
 .section

Finished!

Text



File Edit Views Insert Format Links Tools Help

C:\Program Files\Amaya\amaya\AmayaPage_WX.html

Course Name: Web Based Information Technology
 Course Number: CIS 4004
 Instructor: Dr. Mark Llewellyn
 Class Meets: Monday, Wednesday and Friday, 11:30am - 12:20pm, HEC 118

Course Description: Digital libraries, Media formats, Compression, Streaming Media, Mobile Internet and WML, Emerging technologies. Capacity planning for web services.

Prerequisites:

- CNT 3004 and,
- COP 3330

```

1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4 <html xmlns="http://www.w3.org/1999/xhtml">
5   <head>
6     <title>Web Based Information Technology Spring 2011</title>
7   </head>
8   <body>
9     <div>
10      <span style="font-weight:bold">Course Name: </span> Web Based
  
```

Elements

Style

Theme No theme

Arial 12

Apply class

(no class)

Text



How XHTML Documents Are Structured

- XHTML documents are comprised of a simple three-part framework:
 1. Document prolog
 2. Header section
 3. Body of document.

On the course website (and WebCourses), I've placed a copy of an XHTML template file that you can use as the basis for all of your XHTML documents.

Document prolog declarations

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>  
<!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">
```

The <html> root element

```
<head>  
  <title Strict XHTML Document </title>  
</head>
```

Header information

```
<body>  
  <!-- body of document goes here -->  
</body>  
</html>
```

Open and close <body> tags, between which the main body of the document is contained

XHTML Strict Document Framework



Document Framework Elements

- The elements that make up the framework of XHTML documents do not produce any output in a browser window. Instead, they provide information to the program about the document.
- The elements that makeup the framework are: `<html>`, `<head>`, `<title>`, and `<body>`. We'll look at each of these more closely.



The `<html>` Element

- The `<html>` element is the **root element** of an XHTML document, within which every other element in the document is contained (recall our discussion on nesting of elements).
- The document begins with the `<html>` start tag and ends with the `</html>` end tag. The header and body information of the document are contained in the root element.
- As you may have noticed in the example on page 42 in the previous section of notes, the `<html>` start tag defines an attribute called **xmlns** with a value of `http://www.w3.org/1999/xhtml`. This attribute declares the XHTML namespace and is required in all XHTML documents. If you omit it, it will be inserted automatically by the parser, however, it is recommended that you insert it into the document.



An Aside on Namespaces

- The `<html>` start tag defines an attribute called `xmlns` with a value of `http://www.w3.org/1999/xhtml`.
- The `xmlns` attribute stands for **XML Namespace**, and the value of this attribute defines the namespace that the element names for XHTML belong to.
- Namespaces in XML are collections of element and attribute names, XML allows you to create your own language and your own element and attribute names. This can lead to confusion if your language elements and/or attribute names are the same as someone else's names but have different meanings. Namespaces in XML allows you to specify which namespace the element and attribute names belong to.
- The `xmlns` attribute tells the program parsing the document that the elements and attribute names contained within the `<html>` element belong to the XHTML namespace.
- The value of the `xmlns` attribute, `http://www.w3.org/1999/xhtml`, resembles a Web address, but is simply used to uniquely define the namespace. If you go to this page on the Internet, you will find an informational page provided by the W3C. This is not a requirement of namespace values, it is simply a courtesy provided by the W3C.
- The `xmlns` attribute for the `<html>` element is required for XHTML documents and is specified as a fixed value in each of the three XHTML DTDs. While you can safely omit the attribute from your `<html>` start tag, since the parser will automatically add it, it is recommended to include it.



The <head> Element

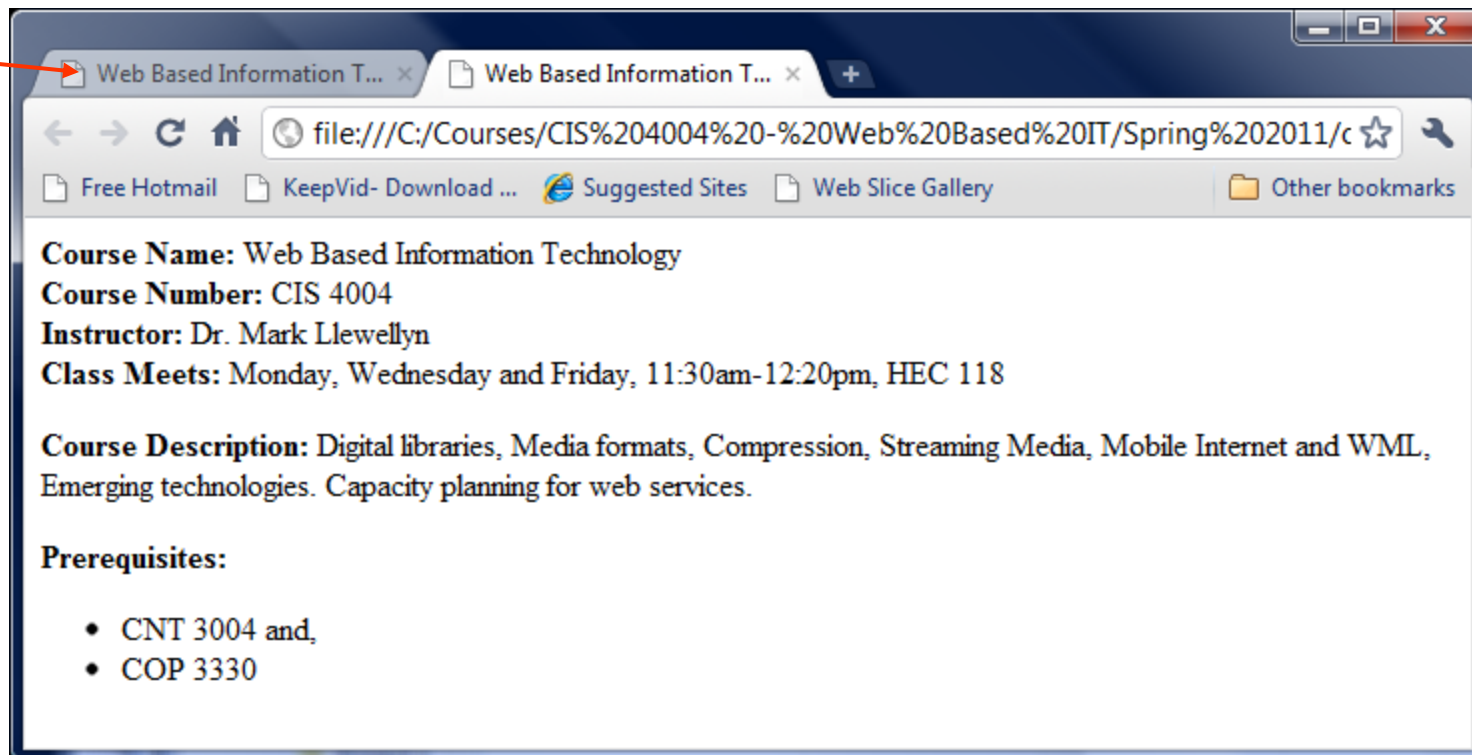
- The start `<head>` tag comes directly after the `<html>` start tag in an XHTML document. This element must be placed inside the `<html>` element.
- It contains information about the document that is mainly used by programs, such as keywords for search engines and link information that defines the relationship this document has to other documents.
- The `<head>` tag also contains the required `<title>` element. The following is a list of elements that can be contained in the `<head>` element – all but the `<title>` element are optional.
 - `<title>` - defines the title of the document.
 - `<base>` - defines the document base URL, which is used for relative links in the document.
 - `<link>` - defines the relationship of this document to other documents.
 - `<meta>` - defines additional information about the document, including the document's content type and special instructions for browsers and search engines.
 - `<script>` - defines links to scripts used within the document, such as Javascript.
 - `<style>` - defines links to style sheets to be used within the document, such as CSS.



The <title> Element

- The <title> element is the only required element within the <head> element. It must be contained within the open and close tags of the <head> element.
- There can only be one <title> element per document. It defines the title of the document that is displayed in the title bar of the browser window as well as the name of bookmarks to that page.
- Most search engines use the content of the <title> element as the text to display on their results pages as well.

title



The `<body>` Element

- The `<body>` element contains the content and all of the markup elements of the document.
- The body of the document is contained between the open `<body>` tag and the ending `</body>` tag.
- All of the other elements we will cover are contained within the `<body>` element.



A Warning About Formatting Elements In XHTML

- Previous versions of HTML included various formatting attributes that would allow you to set the background color for the document and define text attributes in the body element, such as `bgcolor`.
- XHTML Strict uses style sheets to define these formatting attributes, (recall, that this is designed to separate the content from the presentation), so they are not included in the specification.
- If you are using XHTML Transitional or Frameset, you can include additional formatting attributes within the `<body>` element. However, keep in mind the warnings about future compatibility. Style attributes used only in XHTML Transitional and Frameset are:
 - `bgcolor` – sets background color of the document.
 - `text` – sets color of the text in the document.
 - `link` – sets the color of hyperlinks, default color is blue.
 - `vlink` – used to set the color of hyperlinks that have been viewed by the user.
 - `alink` – used to set the color of hyperlinks that are currently active.



Basic Formatting Elements In XHTML

- Now that you understand how XHTML documents are structured, let's start building some Web pages. We'll start with basic formatting elements, show some examples of how to use each of the elements, and create a few documents to illustrate how they look in a Web browser.
- We'll start by looking at block-level formatting elements.



Block-level Formatting Elements – Summary Chart

Element Name	Formatting Style
<code><p>...</p></code>	Paragraph element
<code>
</code>	Line break (empty element)
<code><h1>...</h1></code> to <code><h6>...</h6></code>	Heading elements (1 is largest, 6 is smallest)
<code><hr /></code>	Horizontal rule (empty element)
<code><div>...</div></code>	Section divider



Block-level Formatting Elements

- Documents are broken into logical sections based on the document content to make it easier for users to read.
- The elements shown in the table on the previous page are used to break documents into logical chunks and to label the main content headings.
- These elements are referred to as **block-level elements** because they describe blocks of content.
- We'll examine each of these elements separately.



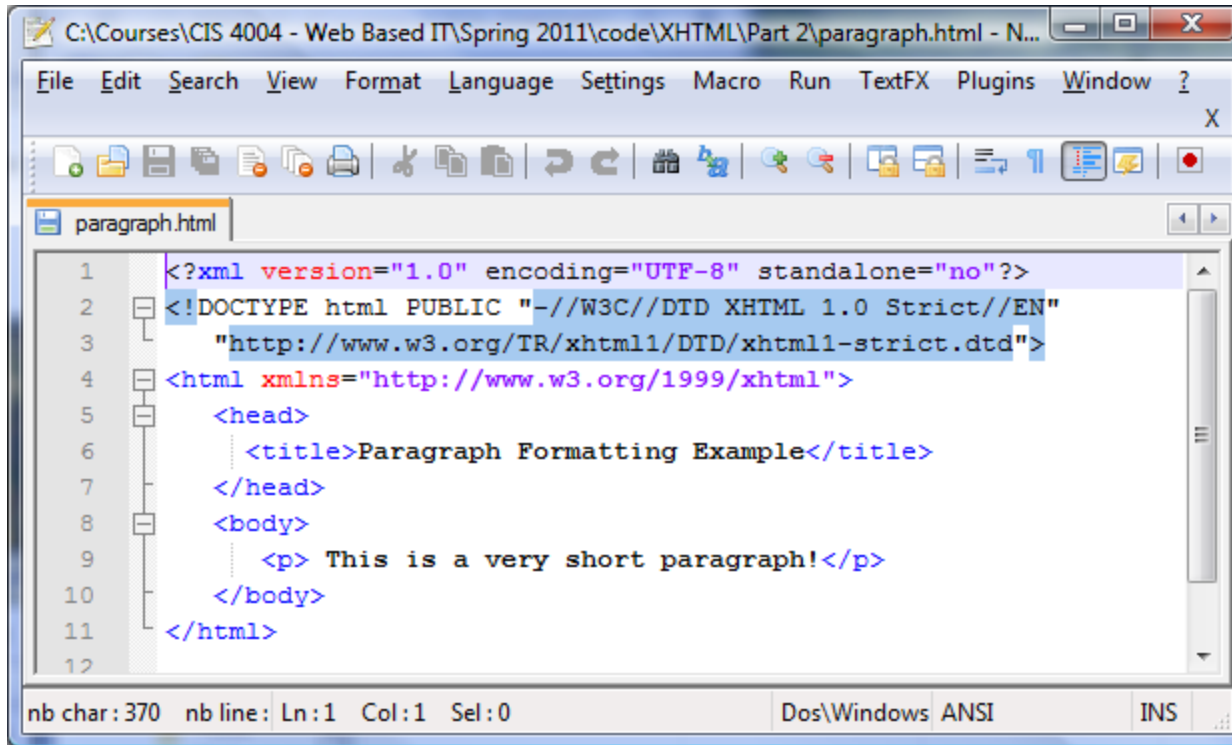
The <p> Element

- The <p> element divides content into **paragraphs**.
- The <p> tag designates the beginning of a paragraph, and the </p> tag ends the paragraph.
- Most browsers will automatically insert a double line return (carriage return) around the paragraph element.
- Example:

```
<p> This is a very short paragraph.</p>
```



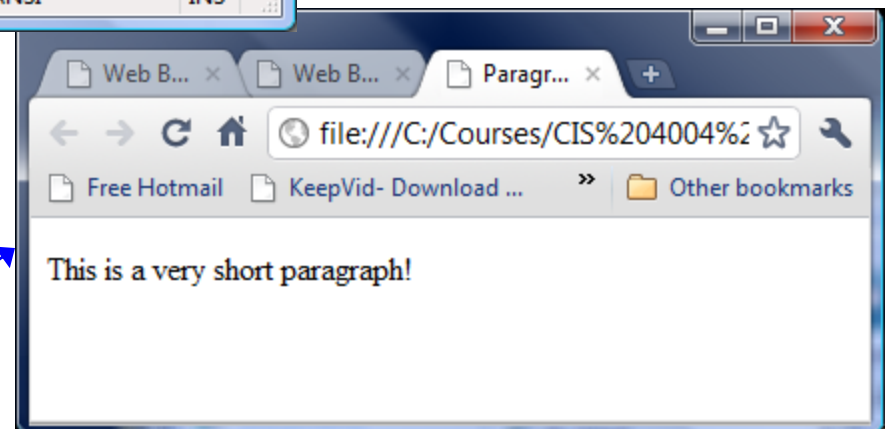
The <p> Element



```
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4 <html xmlns="http://www.w3.org/1999/xhtml">
5   <head>
6     <title>Paragraph Formatting Example</title>
7   </head>
8   <body>
9     <p> This is a very short paragraph!</p>
10  </body>
11 </html>
12
```

XHTML

Rendering



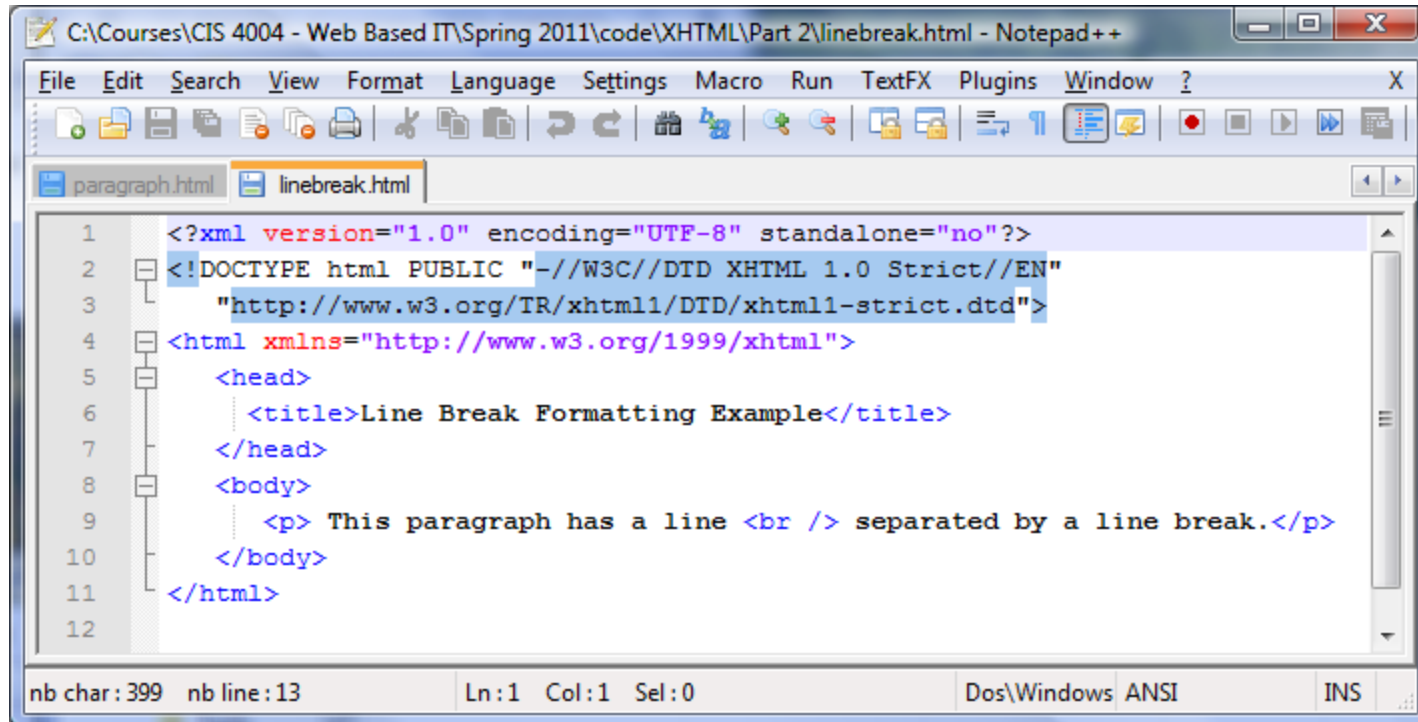
The `
` Element

- The `
` element is the **line break element**. Similar to the `<p>` element, it is used to break up sections of text.
- The `
` element causes the browser to create a single line return (single carriage return).
- The `
` element is an empty element and must end with `/>` in order to conform to the rules of a well-formed document.
- Example:

```
<p> This paragraph has a line <br />  
separated by a line break.</p>
```



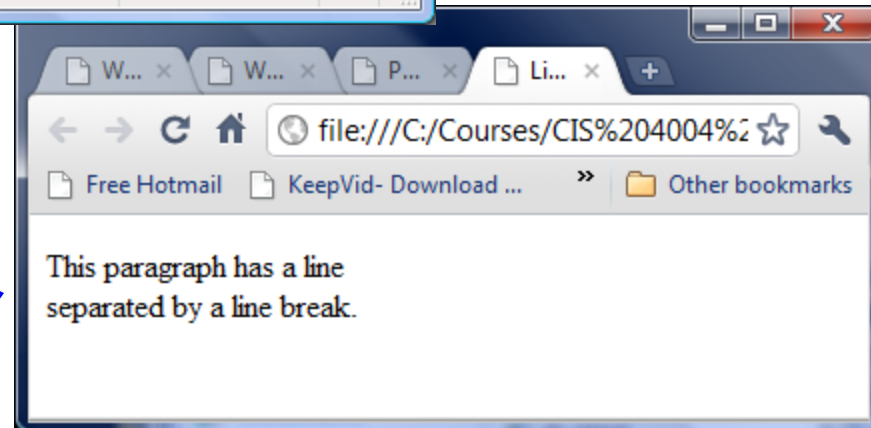
The `
` Element



```
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4 <html xmlns="http://www.w3.org/1999/xhtml">
5   <head>
6     <title>Line Break Formatting Example</title>
7   </head>
8   <body>
9     <p> This paragraph has a line <br /> separated by a line break.</p>
10  </body>
11 </html>
12
```

XHTML

Rendering



The <h1> . . . <h6> Elements

- These elements are the **heading elements**. They are used to label section headings of a document.
- There are six heading levels: <h1>, <h2>, <h3>, <h4>, <h5>, and <h6>.
- The <h1> head element should be used to label the top-most heading, and the rest of the elements should be used for subheads, much like a table of contents hierarchy.
- The browser will display the font for each of these levels differently, starting with a larger font for <h1> and progressively getting smaller as the heading number increases.
- **Example:**

```
<h1> This is a level 1 heading</h1>
```

```
<h2> This is a level 2 heading</h2>
```

```
<h3> This is a level 3 heading</h3>
```

```
<h4> This is a level 4 heading</h4>
```

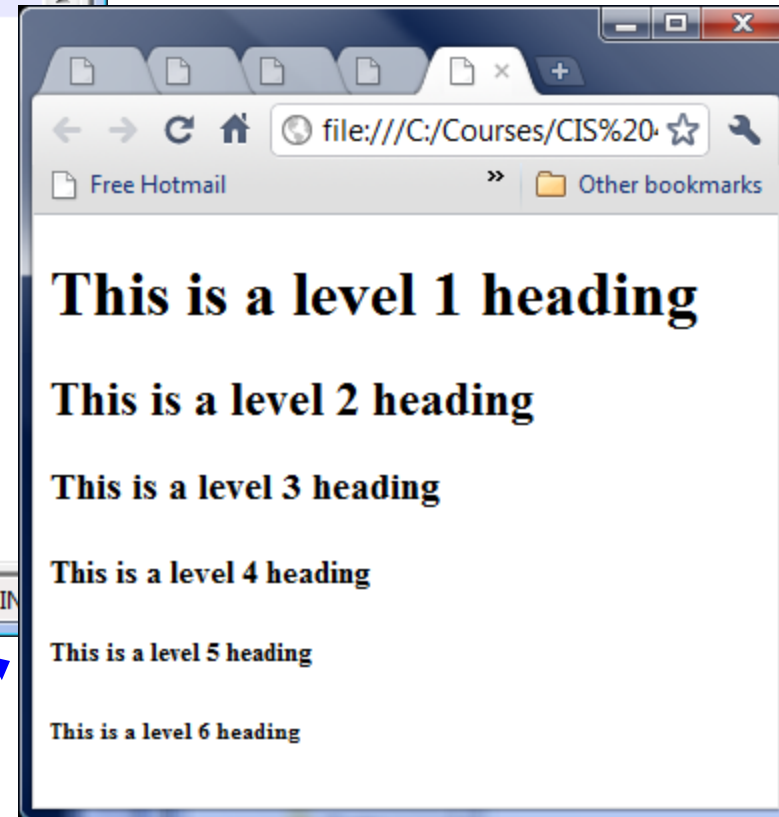
```
<h5> This is a level 5 heading</h5>
```

```
<h6> This is a level 6 heading</h6>
```



The <h1> . . . <h6> Elements

```
C:\Courses\CIS 4004 - Web Based IT\Spring 2011\code\XHTML\Part 2\heading.html - Not...
File Edit Search View Format Language Settings Macro Run TextFX Plugins Window ?
X
paragraph.html linebreak.html heading.html
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4 <html xmlns="http://www.w3.org/1999/xhtml">
5 <head>
6 <title>Header Formatting Example</title>
7 </head>
8 <body>
9 <h1> This is a level 1 heading</h1>
10 <h2> This is a level 2 heading</h2>
11 <h3> This is a level 3 heading</h3>
12 <h4> This is a level 4 heading</h4>
13 <h5> This is a level 5 heading</h5>
14 <h6> This is a level 6 heading</h6>
15 </body>
16 </html>
nb char: 578 nb line Ln: 1 Col: 1 Sel: 0 Dos\Windows ANSI IN
```



XHTML

Rendering



The `<hr />` Element

- The `<hr />` element is the **horizontal rule element**, used to create a visible horizontal line on the Web page to indicate a section break or change in content.
- XHTML Transitional and Frameset provide a set of attributes that can be used with this element to customize the rule. In XHTML Strict customization of the line is done via CSS.
- Example:

There is a horizontal line between this line `<hr />` and this line.

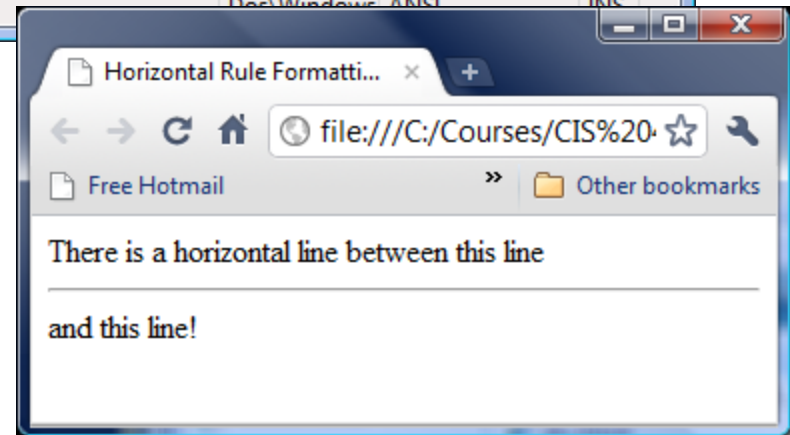


The `<hr />` Element

```
C:\Courses\CIS 4004 - Web Based IT\Spring 2011\code\XHTML\Part 2\horizontal rule.html - Notepad++
File Edit Search View Format Language Settings Macro Run TextFX Plugins Window ?
paragraph.html linebreak.html heading.html horizontal rule.html
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4 <html xmlns="http://www.w3.org/1999/xhtml">
5 <head>
6 <title>Horizontal Rule Formatting Example</title>
7 </head>
8 <body>
9 <div>There is a horizontal line between this line <hr /> and this line!</div>
10 </body>
11 </html>
12
13
HyperText Markup Language: 413 characters, 13 lines. Ln:1 Col:1 Sel:0
```

XHTML

Rendering



The <div> Element

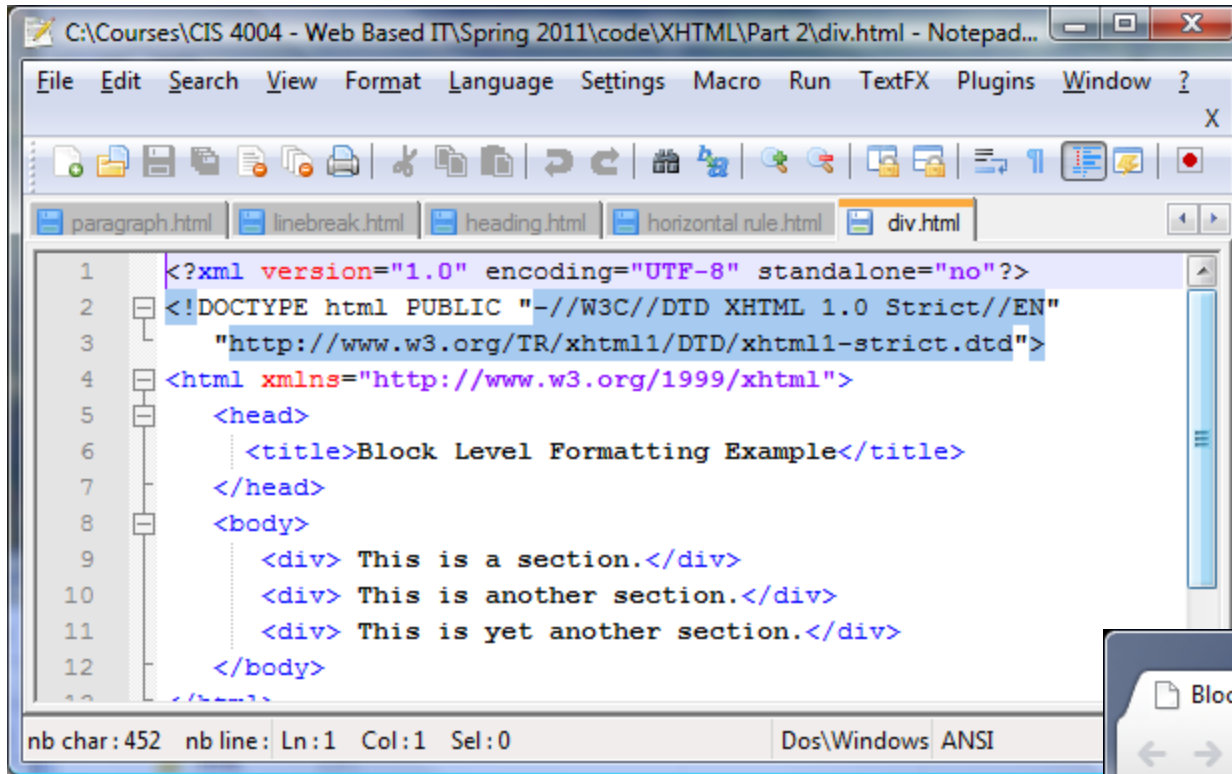
- The <div> element is used to divide sections of content. This element is used to label sections of the document, and can contain any number of other elements.
- This element can use the id and class core XHTML attributes to identify the various sections of the document to be used with parser programs.
- Example:

```
<div> This is a section.</div>
```

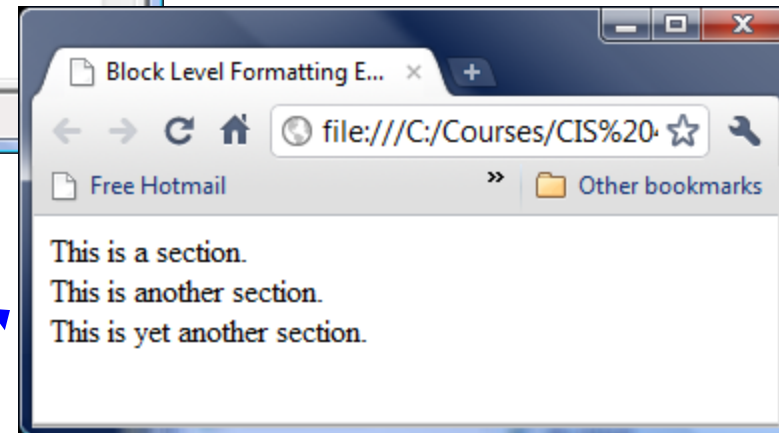
```
<div> This is another section.</div>
```



The <div> Element



```
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4 <html xmlns="http://www.w3.org/1999/xhtml">
5   <head>
6     <title>Block Level Formatting Example</title>
7   </head>
8   <body>
9     <div> This is a section.</div>
10    <div> This is another section.</div>
11    <div> This is yet another section.</div>
12  </body>
13 </html>
```



XHTML

Rendering



Text Formatting Elements

- Text formatting elements are referred to as **character-level elements** because, unlike the block-level elements, which describe blocks of content, these elements describe the text itself.
- Character-level elements describe the formatting of words or phrases as opposed to sections or paragraphs.
- There are two basic groups of text formatting elements: **presentation styles**, and **logical styles**.
 - Presentational styles describe how the text should be displayed, in bold type or italics, for example.
 - Logical styles describe the meaning of the style more than the actual format.

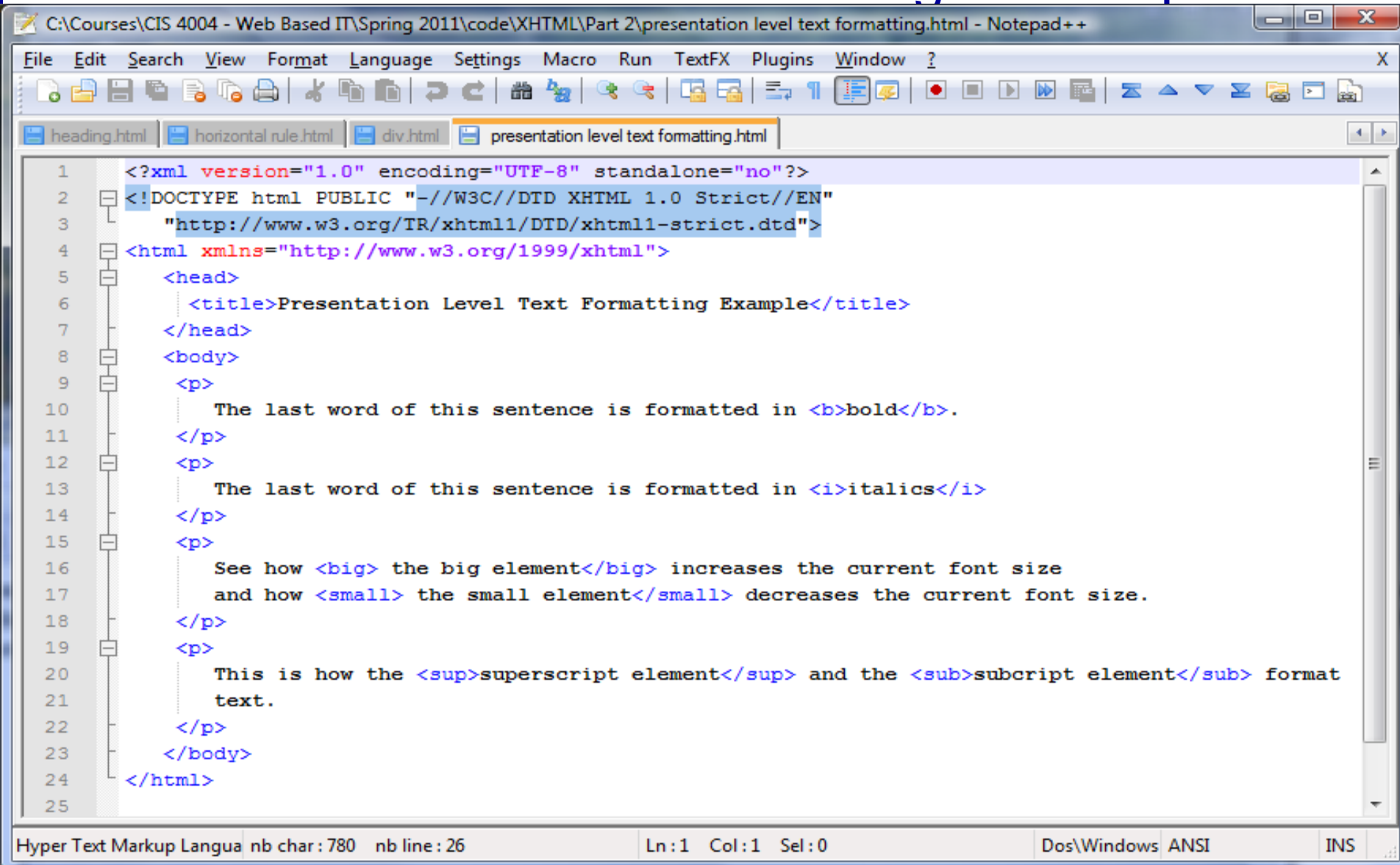


Presentational Text Formatting – Summary Chart

Element Name	Formatting Style
<code>...</code>	Bold font style
<code><big>...</big></code>	Increases current font size
<code><i>...</i></code>	Italic font style
<code><small>...</small></code>	Decrease current font size
<code><sub>...</sub></code>	Subscripted text
<code><sup>...</sup></code>	Superscripted text



Presentation Text Formatting – Example



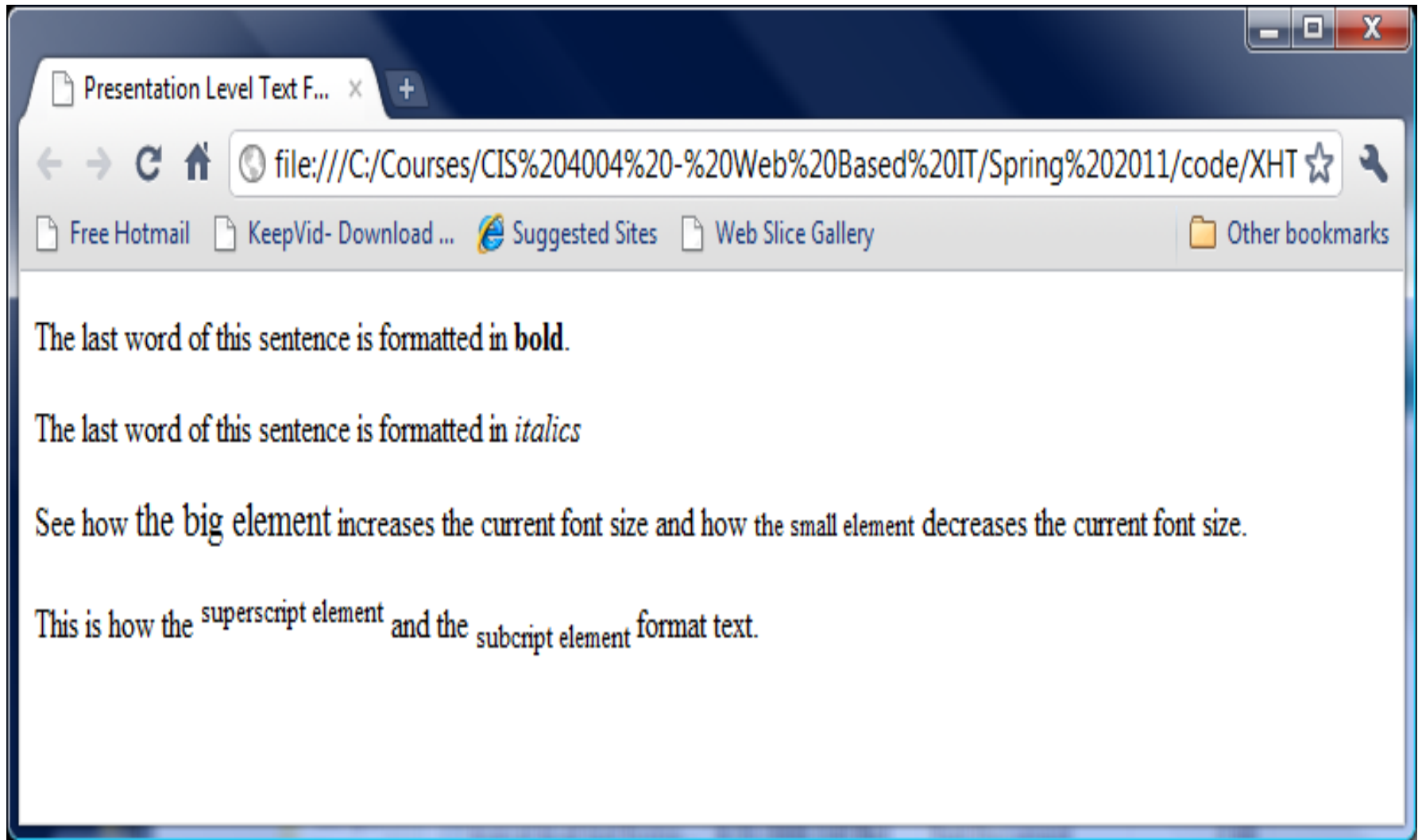
The screenshot shows a Notepad++ window with the following content:

```
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4 <html xmlns="http://www.w3.org/1999/xhtml">
5   <head>
6     <title>Presentation Level Text Formatting Example</title>
7   </head>
8   <body>
9     <p>
10      The last word of this sentence is formatted in <b>bold</b>.
11    </p>
12    <p>
13      The last word of this sentence is formatted in <i>italics</i>
14    </p>
15    <p>
16      See how <big> the big element</big> increases the current font size
17      and how <small> the small element</small> decreases the current font size.
18    </p>
19    <p>
20      This is how the <sup>superscript element</sup> and the <sub>subscript element</sub> format
21      text.
22    </p>
23  </body>
24 </html>
25
```

At the bottom of the window, the status bar displays: Hyper Text Markup Language, nb char : 780, nb line : 26, Ln:1 Col:1 Sel:0, Dos\Windows ANSI, INS.



Presentational Text Formatting – Example



Logical Text Formatting Elements

- Logical text formatting describe the meaning of the style more than the actual format.
- Initially, browsers were left to determine the presentation of these tags as they saw fit, but over time certain standards were developed, and these are unlikely to change in the foreseeable future.
- For example, if you want a certain type, like bold, you should use the `` elements, but the `` element will give you the same effect, because most browsers will interpret `` as ``.
- The table on the next page lists the most commonly used logical elements.

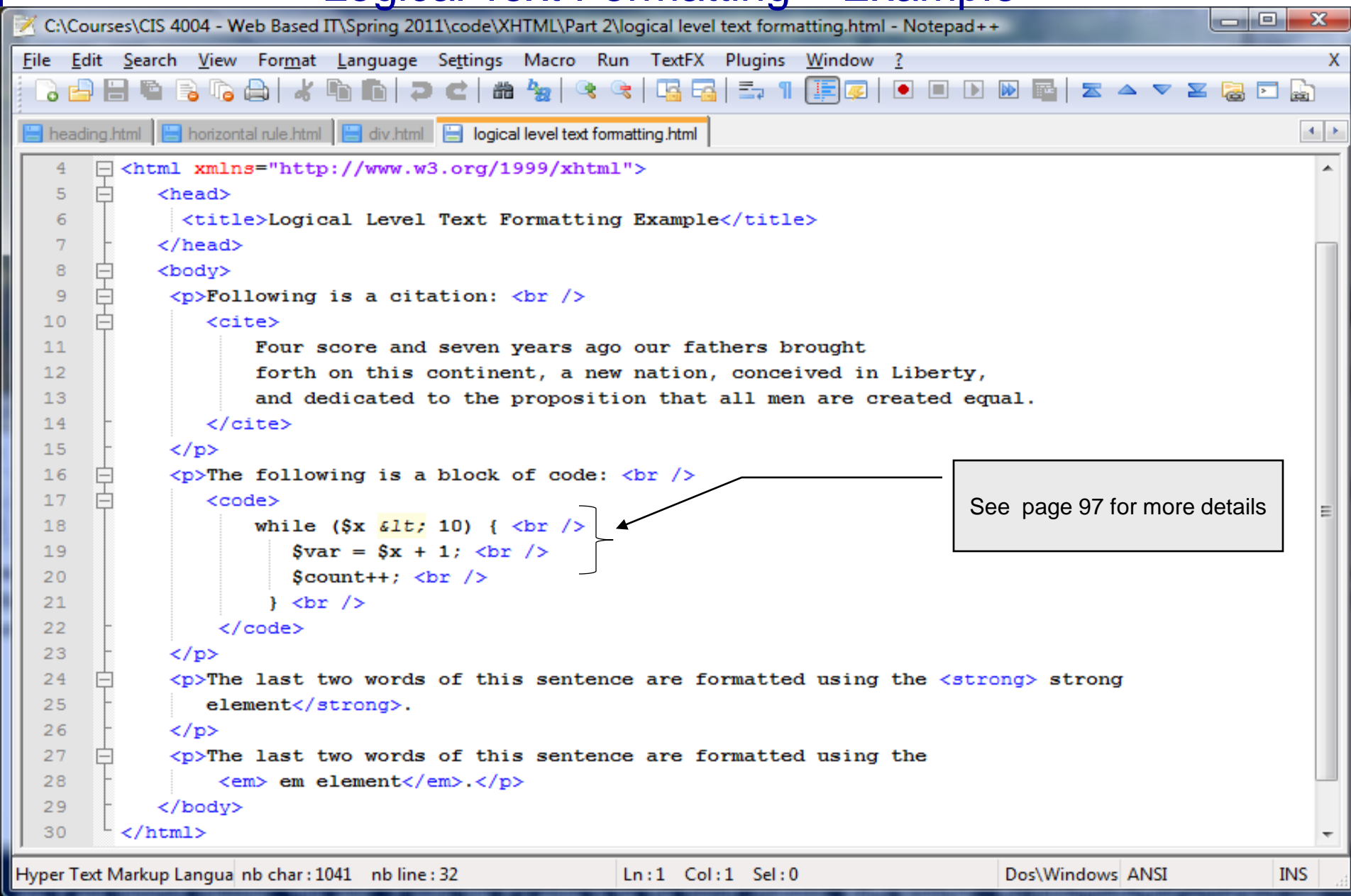


Logical Text Formatting – Summary Chart

Element Name	Formatting Style
<code><cite>...</cite></code>	Defines a citation
<code><code>...</code></code>	Presents computer code examples
<code>...</code>	Emphasis. In most browsers, this is italics
<code>...</code>	Emphasis. In most browsers, this is bold
<code>...</code>	Provides a logical inline grouping with no predefined look.



Logical Text Formatting – Example



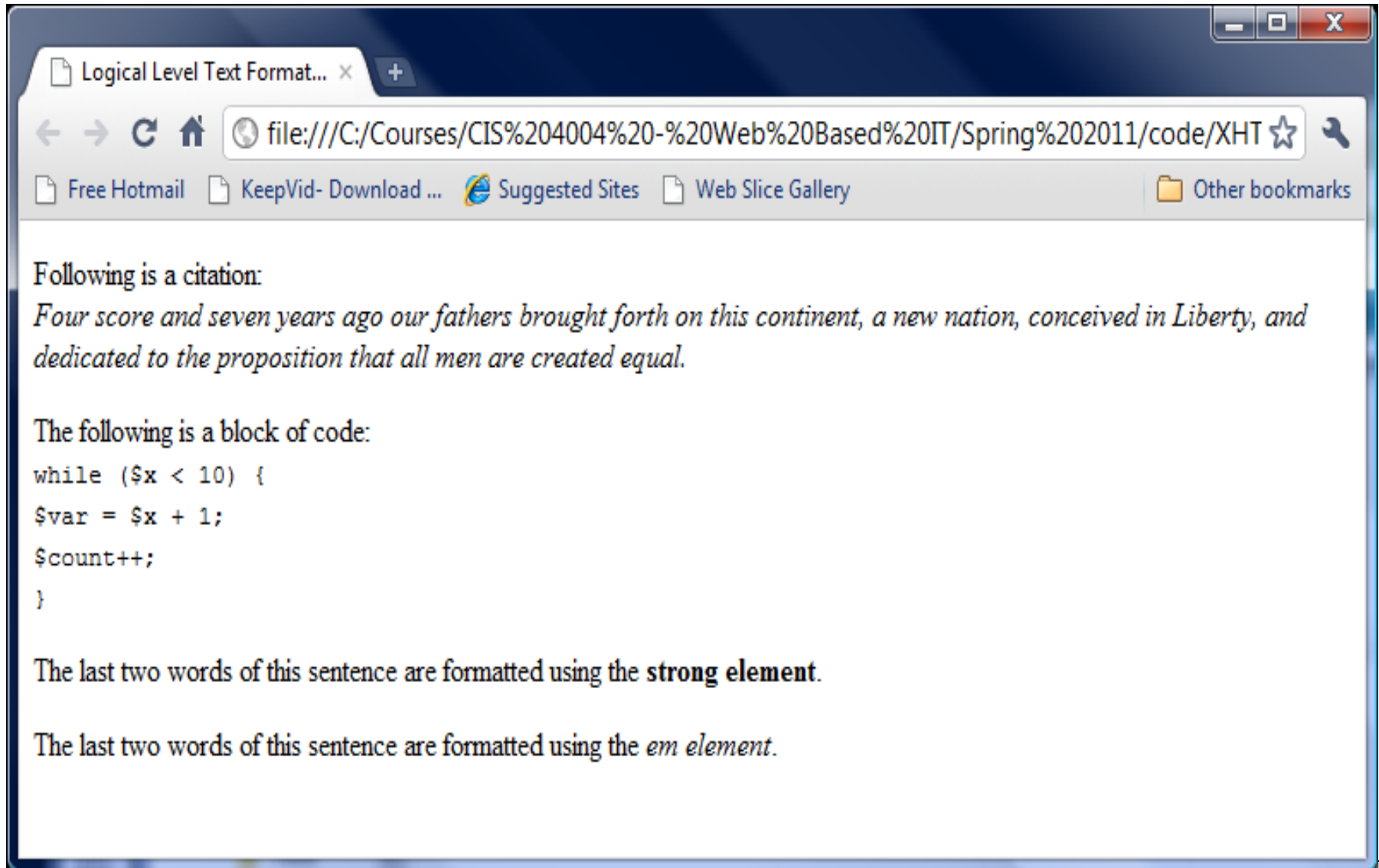
```
4 <html xmlns="http://www.w3.org/1999/xhtml">
5   <head>
6     <title>Logical Level Text Formatting Example</title>
7   </head>
8   <body>
9     <p>Following is a citation: <br />
10    <cite>
11      Four score and seven years ago our fathers brought
12      forth on this continent, a new nation, conceived in Liberty,
13      and dedicated to the proposition that all men are created equal.
14    </cite>
15  </p>
16  <p>The following is a block of code: <br />
17    <code>
18      while ($x &lt; 10) { <br />
19        $var = $x + 1; <br />
20        $count++; <br />
21      } <br />
22    </code>
23  </p>
24  <p>The last two words of this sentence are formatted using the <strong> strong
25    element</strong>.
26  </p>
27  <p>The last two words of this sentence are formatted using the
28    <em> em element</em>.</p>
29  </body>
30 </html>
```

See page 97 for more details

Hyper Text Markup Language nb char : 1041 nb line : 32 Ln:1 Col:1 Sel:0 Dos\Windows ANSI INS



Presentational Text Formatting – Example



Logical Level Text Format... x +

file:///C:/Courses/CIS%204004%20-%20Web%20Based%20IT/Spring%202011/code/XHT

Free Hotmail | KeepVid- Download ... | Suggested Sites | Web Slice Gallery | Other bookmarks

Following is a citation:

Four score and seven years ago our fathers brought forth on this continent, a new nation, conceived in Liberty, and dedicated to the proposition that all men are created equal.

The following is a block of code:

```
while ($x < 10) {  
  $var = $x + 1;  
  $count++;  
}
```

The last two words of this sentence are formatted using the **strong element**.

The last two words of this sentence are formatted using the *em element*.



A Bit More On `<div>` and ``

- The `<div>` and `` elements are most often used in conjunction with style sheets (as we will see later) to avoid the default rendering of elements.
- `<div>` is a block element whereas `` is an inline element. Neither of these elements have a default rendering, which in a sense makes them generic tags. Since they have no default rendering, they are very useful for arbitrary style duties.
- A `<div>` element induces a hard return while the inline `` element does not.
- The following example will more clearly illustrate the differences between these two elements.



A Bit More On <div> and

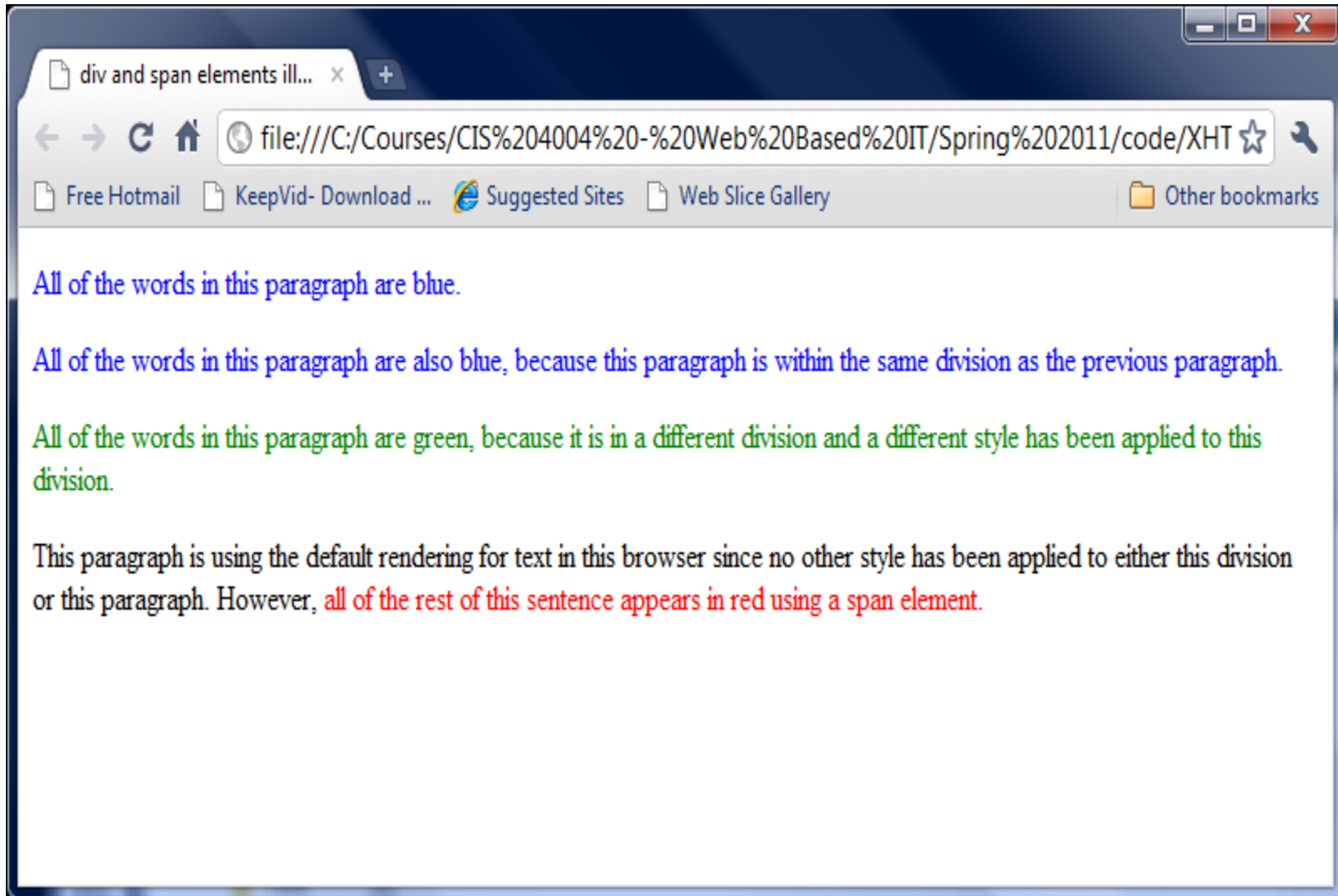
```
C:\Courses\CIS 4004 - Web Based IT\Spring 2011\code\XHTML\Part 2\div and span illustrated.html - Notepad++
File Edit Search View Format Language Settings Macro Run TextFX Plugins Window ?
heading.html horizontal rule.html div and span illustrated.html
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4 <html xmlns="http://www.w3.org/1999/xhtml">
5 <head>
6 <title>div and span elements illustrated </title>
7 </head>
8 <body>
9 <div style="color:blue">
10 <p>All of the words in this paragraph are blue. </p>
11 <p>All of the words in this paragraph are also blue, because this paragraph is
12 within the same division as the previous paragraph. </p>
13 </div>
14 <div style="color:green">
15 <p>All of the words in this paragraph are green, because it is in a different division and
16 a different style has been applied to this division.</p>
17 </div>
18 <div>
19 <p> This paragraph is using the default rendering for text in this browser since no other
20 style has been applied to either this division or this paragraph. However, <span style="col
21 all of the rest of this sentence appears in red using a span element.</span>
22 </p>
23 </div>
24 </body>
25 </html>
26
```

This is an internal style (i.e., CSS)

Hyper Text Markup Language nb char : 1089 nb line : 26 Ln:1 Col:1 Sel:0 Dos\Windows ANSI INS



A Bit More On `<div>` and ``



Character Entities In XHTML

- The character of the less than symbol shown in the example on page 29 is written as `<`.
- Certain characters in XHTML have special meaning to parser, like less than `<` and greater than `>`. These characters identify the beginning and ending of a tag, so if you want to add these characters as literal values, you must use the [character entity code](#) for them.
- A character entity is written in the following syntax: `&code;`. It begins with an ampersand (`&`) character, then the code for the entity, then a semicolon (`;`).
- Hundreds of symbols can be referenced and included on Web pages using entities. Some of the more popular symbols can be referenced by their abbreviations, like less than (`<`) and greater than (`>`), but they can also be referenced using their decimal value in the ASCII Table.
- Some of the most common character entity codes are shown in the table on the next page.



Some Common Character Entity Codes

Symbol	Description	XHTML Code
>	Greater than	> or &62;
<	Less than	< or &60;
®	Trademark	™ or &174;
©	Copyright	© or &169;
¢	Cent sign	¢ or &162;
	Non-breaking space	 or &160



XHTML Lists

- XHTML provides three main types of lists: numbered, bulleted, and definition. These are summarized in the table below.

List type	XHTML element	Item element
Ordered list (numbered)	<code>...</code>	<code>...</code>
Unordered list (bulleted)	<code>...</code>	<code>...</code>
Definition list	<code><dl>...</dl></code>	<code><dt>...</dt></code> and <code><dd>...</dd></code>

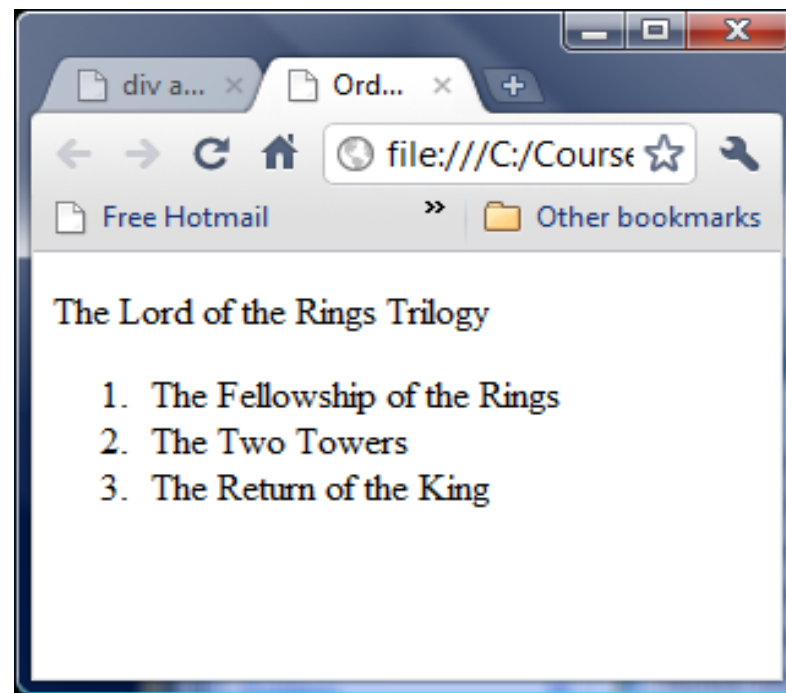


Ordered Lists

- Ordered lists are numbered and are contained within the `...` , **ordered list element**.
- An ordered list may have any number of `` **list items** appearing in the element content.
- A browser will list each of the elements in a number sequential list.
- For example:

```
<p> The Lord of the Rings Trilogy</p>
<ol>
  <li>The Fellowship of the Rings</li>
  <li>The Two Towers</li>
  <li>The Return of the King</li>
</ol>
```

Viewed in a browser →

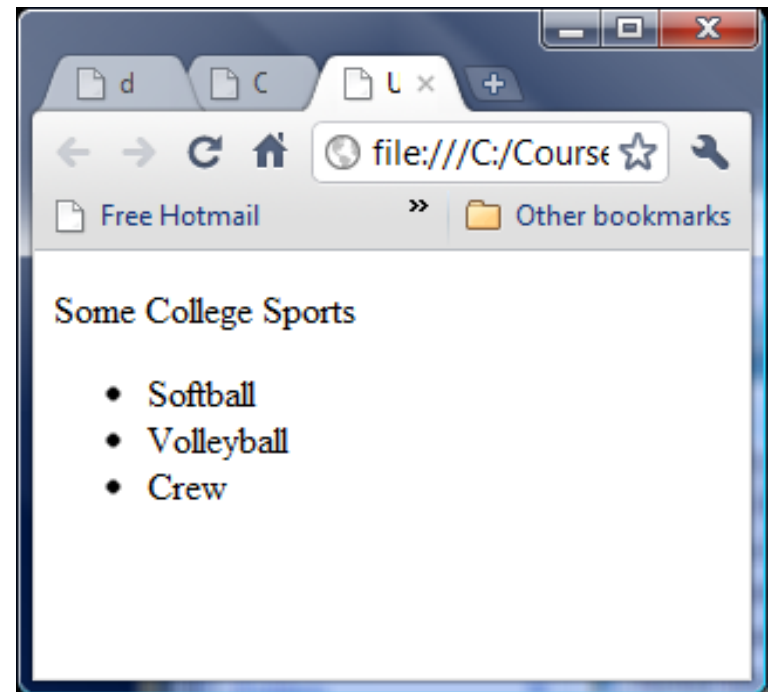


Unordered Lists

- Unordered lists are bulleted instead of numbered. An unordered list is contained within the `...`, **unordered list element**.
- An unordered list may have any number of `` **list items** appearing in the element content.
- A browser will list each of the elements in a bulleted list.
- For example:

```
<p> College Sports</p>
<ul>
  <li>Softball</li>
  <li>Volleyball</li>
  <li>Crew</li>
</ul>
```

Viewed in a browser

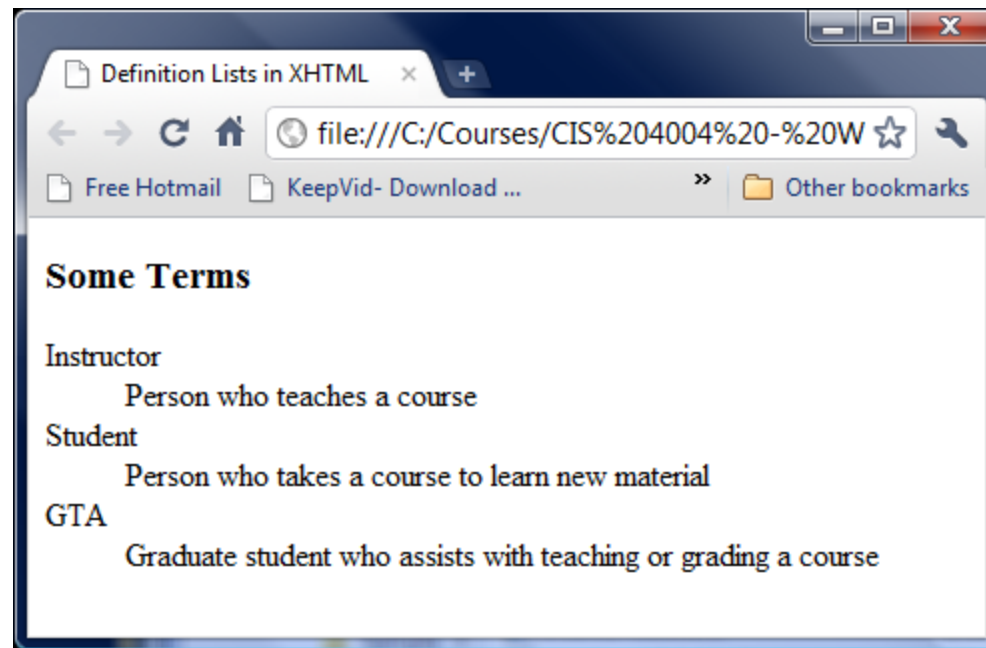


Definition Lists

- Definition lists are lists of terms and their definitions. They are a little different than ordered and unordered lists in that the items are lists in pairs.
- The `<dl>...</dl>` , surround the **definition list**. The name of the term appears between `<dt>` and `</dt>` tags, and the definition is between `<dd>` and `</dd>` tags.
- For example:

```
<h3> Some Terms</h3>
<dl>
  <dt>Instructor</dt><dd>Person who
    teaches a course</dd>
  <dt>Student</dt><dd>Person who
    takes a course to learn new
    material</dd>
  <dt>GTA</dt><dd>Graduate student
    who assists with teaching a
    course</dd>
</dl>
```

Viewed in a browser

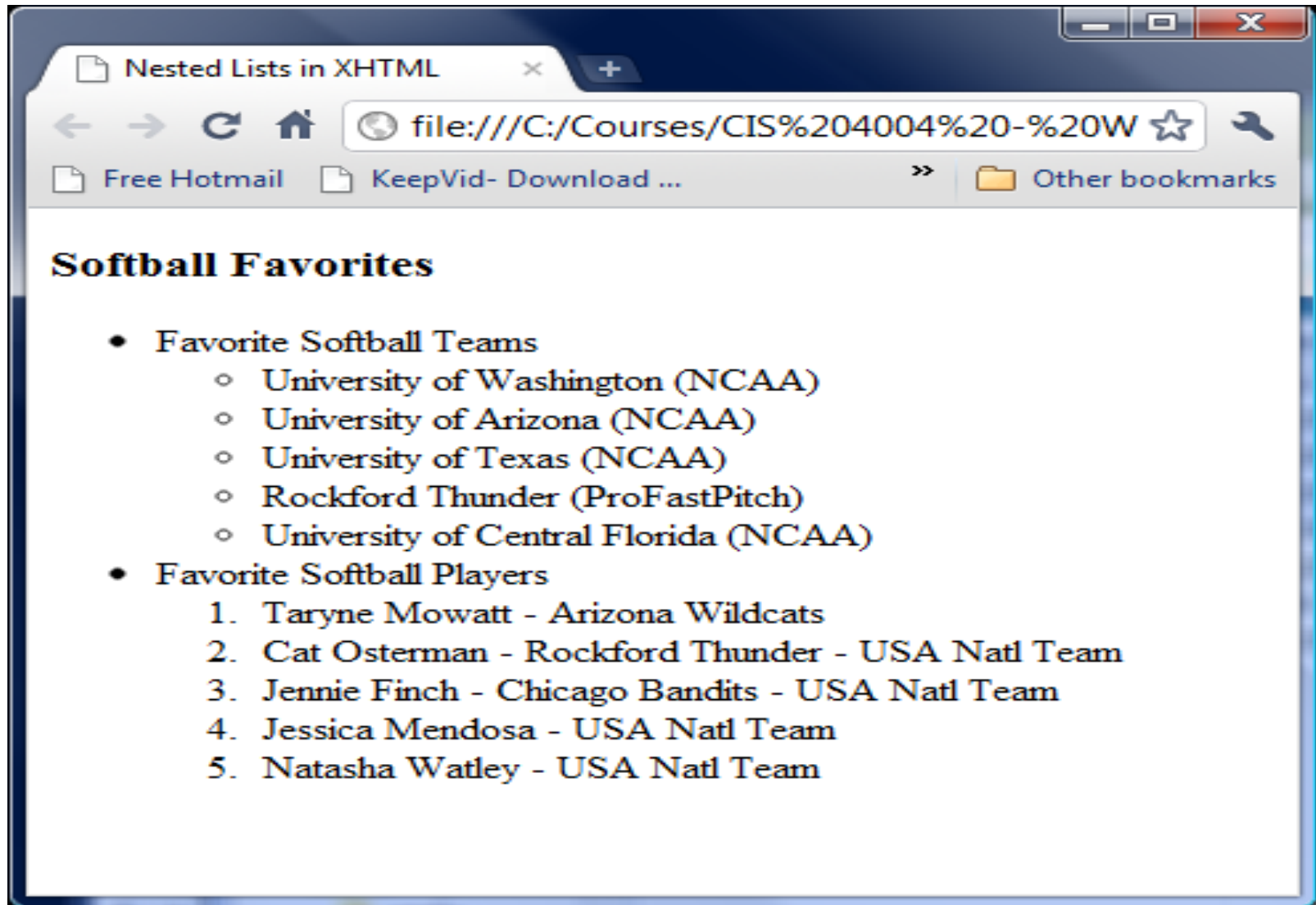


Nesting Lists – Example

```
C:\Courses\CIS 4004 - Web Based IT\Spring 2011\code\XHTML\Part 3\nested lists.html - ...
File Edit Search View Format Language Settings Macro Run TextFX Plugins Window ?
heading.html horizontal rule.html ordered lists.html nested lists.html
4 <html xmlns="http://www.w3.org/1999/xhtml">
5   <head>
6     <title>Nested Lists in XHTML</title>
7   </head>
8   <body>
9     <h3> Softball Favorites</h3>
10    <ul>
11      <li>Favorite Softball Teams
12        <ul>
13          <li>University of Washington (NCAA)</li>
14          <li>University of Arizona (NCAA)</li>
15          <li>University of Texas (NCAA)</li>
16          <li>Rockford Thunder (ProFastPitch)</li>
17          <li>University of Central Florida (NCAA)</li>
18        </ul>
19      </li>
20      <li>Favorite Softball Players
21        <ol>
22          <li>Taryne Mowatt - Arizona Wildcats</li>
23          <li>Cat Osterman - Rockford Thunder - USA Natl Team</li>
24          <li>Jennie Finch - Chicago Bandits - USA Natl Team</li>
25          <li>Jessica Mendosa - USA Natl Team</li>
26          <li>Natasha Watley - USA Natl Team</li>
27        </ol>
28      </li>
29    </ul>
30  </body>
31 </html>
32
```



Nesting Lists – Example



The screenshot shows a web browser window with the title "Nested Lists in XHTML". The address bar contains the file path "file:///C:/Courses/CIS%204004%20-%20W". The browser's bookmark bar shows "Free Hotmail", "KeepVid- Download ...", and "Other bookmarks". The main content area displays the following text:

Softball Favorites

- Favorite Softball Teams
 - University of Washington (NCAA)
 - University of Arizona (NCAA)
 - University of Texas (NCAA)
 - Rockford Thunder (ProFastPitch)
 - University of Central Florida (NCAA)
- Favorite Softball Players
 1. Taryne Mowatt - Arizona Wildcats
 2. Cat Osterman - Rockford Thunder - USA Natl Team
 3. Jennie Finch - Chicago Bandits - USA Natl Team
 4. Jessica Mendosa - USA Natl Team
 5. Natasha Watley - USA Natl Team



Hyperlinks

- A **hypertext link**, or **hyperlink** is an object in a Web page that when clicked will redirect the browser to another Web page or file.
- Usually, hyperlinks take the form of blue, underlined text, or an image.
- Special linking elements are included in the XHTML (also in HTML) specification that allow Web page authors to use images or text within a Web page to create these links to other resources. The resource being linked to by the hyperlink is called the **target resource**.
- In addition to other Web pages, the target resource can be an image file, a multimedia file (such as an audio or video file), another section within the same page, or any Web page or file anywhere on the Internet.
- Hyperlinks provide Web page authors with a powerful means of organizing information and allow them to create very complex, cross-referenced Web sites with clickable tables of contents and menus.



Creating Hyperlinks With The `<a>` Element

- The `<a>` or **anchor element** in XHTML is used to create hyperlinks. These links require the user to perform an action – usually clicking on the link – in order to for the link to do anything. The clickable region of the link can consist of text or images.
- If the user never clicks the linked image or text, the link is never activated. Passively moving the cursor over the hyperlink will not activate it.
- The syntax of an anchor element is:

Value of the `href` attribute is the URL of the target resource.

```
<a href = "http://www.cs.ucf.edu/courses/cgs3175/fall2009/index.html">
```

```
  This is a link. </a>
```

Clickable area of the link in a Web browser.

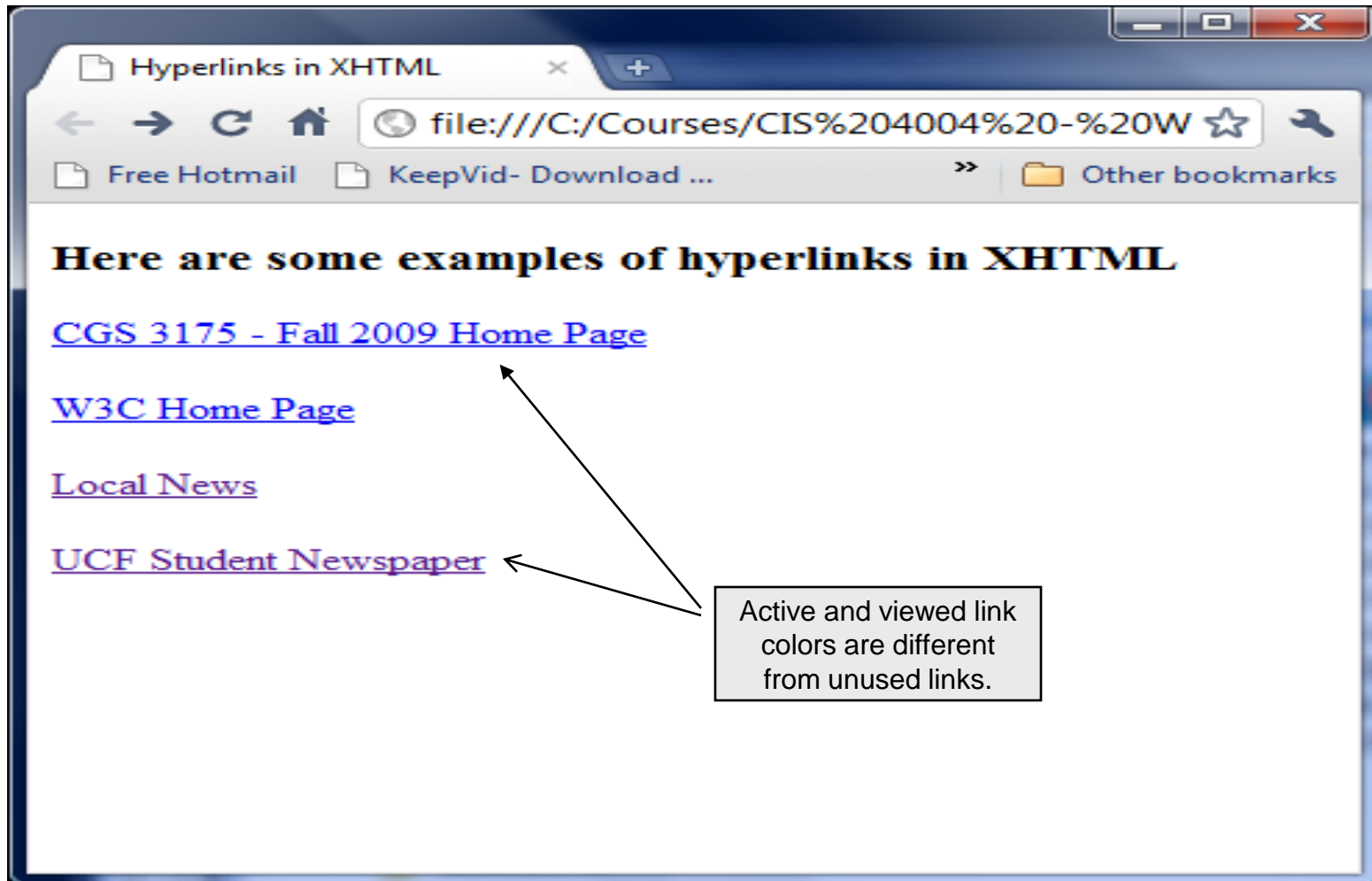


Hyperlink – Example

```
*C:\Courses\CIS 4004 - Web Based IT\Spring 2011\code\XHTML\Part 3\hyperlinks.html - Notepad++
File Edit Search View Format Language Settings Macro Run TextFX Plugins Window ?
heading.html horizontal rule.html hyperlinks.html
1 <?xml version="1.0" encoding="UTF-8" standalone="no" ?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4 <html xmlns="http://www.w3.org/1999/xhtml">
5 <head>
6 <title>Hyperlinks in XHTML</title>
7 </head>
8 <body>
9 <div>
10 <h3> Here are some examples of hyperlinks in XHTML</h3>
11 <a href="http://www.cs.ucf.edu/courses/cis4004/spr2011/index.html">CIS 4004 - Spring 2011 Home Page</a>
12 <br /><br />
13 <a href="http://www.w3c.org">W3C Home Page</a>
14 <br /><br />
15 <a href="http://www.cfnews13.com">Local News</a>
16 <br /><br />
17 <a href="http://www.centralfloridafuture.com">UCF Student Newspaper</a>
18 </div>
19 </body>
20 </html>
21
Hyper Text Markup Language file nb char : 728 nb line : 21 Ln : 11 Col : 94 Sel : 0 Dos\Windows ANSI INS
```



Hyperlink – Example



Relative versus Absolute URLs

- **Relative URLs** are used to link documents that reside on the same Web server. When a relative URL is used, the protocol and domain name are omitted. The link to the target resource is relative to the location of the document containing the link, or the **source document**.
- If the target resource resides in the same directory as the source document, you can use a link containing only the name of the target resource, as in the first example below.
- If the target resource resides in a different directory on the Web server, you must include the subdirectory information in the link, as in the second example below.

```
<a href="newpage.html">Click Here</a>
```

```
<a href="documents/newpage.html">Click Here</a>
```

```
<a href="images/mypicture.jpg">Click Here</a>
```



Relative versus Absolute URLs

- **Absolute URLs** are used to link documents that reside on different Web servers. When an absolute URL is used, the protocol (`http://`) and domain name (`cs.ucf.edu`) and domain name are included to direct the Web browser to the location of the new Web server. The absolute URL does not take into account any location information about the current document and can reference any target resource anywhere on the Internet.
- Below are some examples.

```
<a href="http://www.cs.ucf.edu/courses/cis4004/spr2011/index.html">Click Here</a>
```

```
<a href="http://www.cs.ucf.edu/courses/cis4004/spr2011/background.gif"> Click Here</a>
```



Linking Within A Single Document

- If you are working with a large document, you may want to create links to sections within that document.
- For example, you may want to create a link at the bottom of the document that links back to the top of the document, or a link that will take you to a footnote at the bottom of a page from within the body of the document.
- You see internal linking quite often when viewing on-line tutorials, or documentation in which each chapter is linked from one to the next and even pages within a chapter are linked from one to the next.



Linking Within A Single Document

- In order to create an **internal link**, you will need to first create the anchor at the place where you want the link to link to. The anchor element is used with an attribute called **name**, which identifies the anchor, or **target**.

```
<a name="footnote">Footnote</a>
```

- Next, you need to create a link that looks like the relative links we've already examined, but has a # sign in front of the relative URL to tell the browser that this link exists in the current document. This link would look like:

```
<a href="#footnote">Link to footnote</a>.
```

- This would create an anchor where the footnote resides in the document, and clicking on the link would then take the user to that place within the document. The example on the next couple of pages illustrates internal links.



Linking Within A Single Document

```
C:\Courses\CIS 4004 - Web Based IT\Spring 2011\code\XHTML\Part 3\internal links.html - Notepad++
File Edit Search View Format Language Settings Macro Run TextFX Plugins Window ?
heading.html horizontal rule.html hyperlinks.html internal links.html
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4 <html xmlns="http://www.w3.org/1999/xhtml">
5 <head>
6 <title>Internal links in XHTML</title>
7 </head>
8 <body>
9 <h3> How internal links work in XHTML</h3>
10 <p>
11 This is the body of the document.
12 It isn't really very big, but we'll pretend
13 that it is a very large document. <a href="#footnote1">see footnote 1</a>
14 To really see the effect of the browser "moving" you to the location of
15 the footnote...you either want to make the few paragraphs in this document
16 a lot bigger, or you can shrink the size of your browser window, which is
17 what I did so that it would fit onto the notes reasonably well.
18 </p>
19 <p>
20 This is the second paragraph in our "large" document.
21 It isn't much bigger than the first, but it also will include
22 this footnote.<a href="#footnote2">see footnote 2</a><a name="return2"></a>
23 This example you'll need to run yourself to really experience how the browser
24
```

Hyper Text Markup Language file nb char : 1980 nb line : 45 Ln : 1 Col : 1 Sel : 0 Dos\Windows ANSI INS

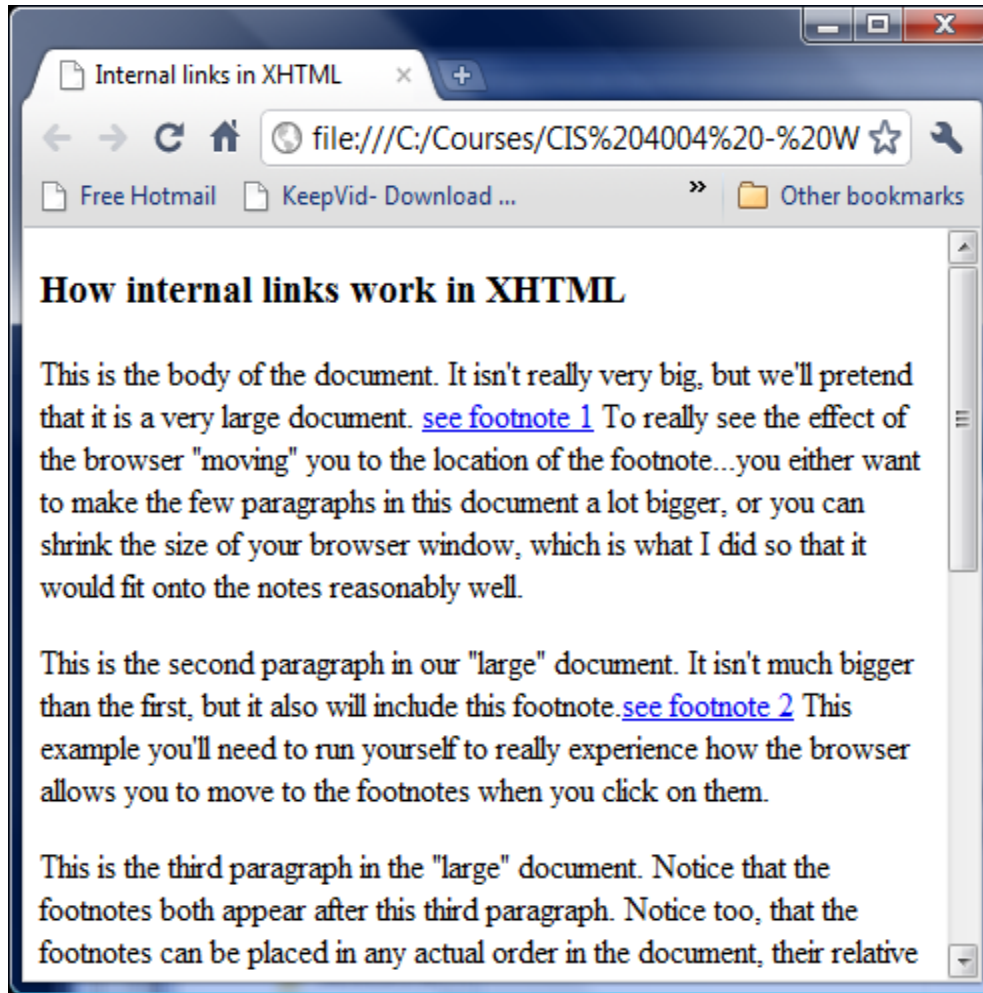


Linking Within A Single Document

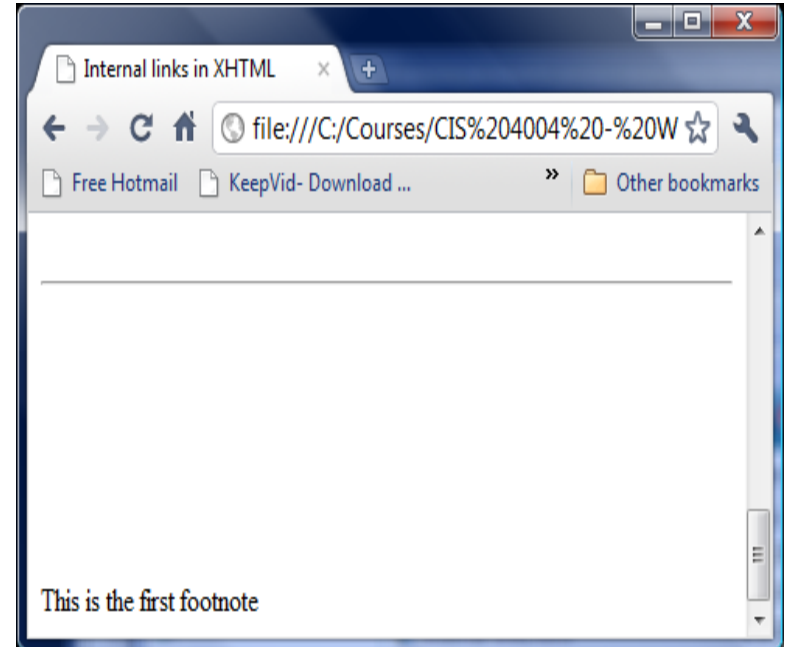
```
C:\Courses\CIS 4004 - Web Based IT\Spring 2011\code\XHTML\Part 3\internal links.html - Notepad++
File Edit Search View Format Language Settings Macro Run TextFX Plugins Window ?
heading.html horizontal rule.html hyperlinks.html internal links.html
22 It isn't much bigger than the first, but it also will include
23     this footnote.<a href="#footnote2">see footnote 2</a><a name="return2"></a>
24 This example you'll need to run yourself to really experience how the browser
25 allows you to move to the footnotes when you click on them.
26 </p>
27 <p>
28     This is the third paragraph in the "large" document.
29     Notice that the footnotes both appear after this third paragraph.
30     Notice too, that the footnotes can be placed in any actual order
31     in the document, their relative order is based on the reference to
32     them in the actual document.
33 </p>
34 <!-- spacing and horizontal rule are for effect only - neither are required. -->
35 <div>
36 <br /> <br /> <hr /> <br /> <br />
37 <span><a name="footnote2">This is the second footnote</a></span>
38     <br /> <br /> <br /> <br /> <br /> <hr /> <br />
39 <br /> <br /> <br /> <br /> <br /> <br />
40 <span><a name="footnote1">This is the first footnote</a></span>
41 </div>
42 </body>
43 </html>
44
45
Hyper Text Markup Language file  nb char : 1980  nb line : 45  Ln : 14  Col : 79  Sel : 0  Dos\Windows ANSI  INS
```



Linking Within A Single Document



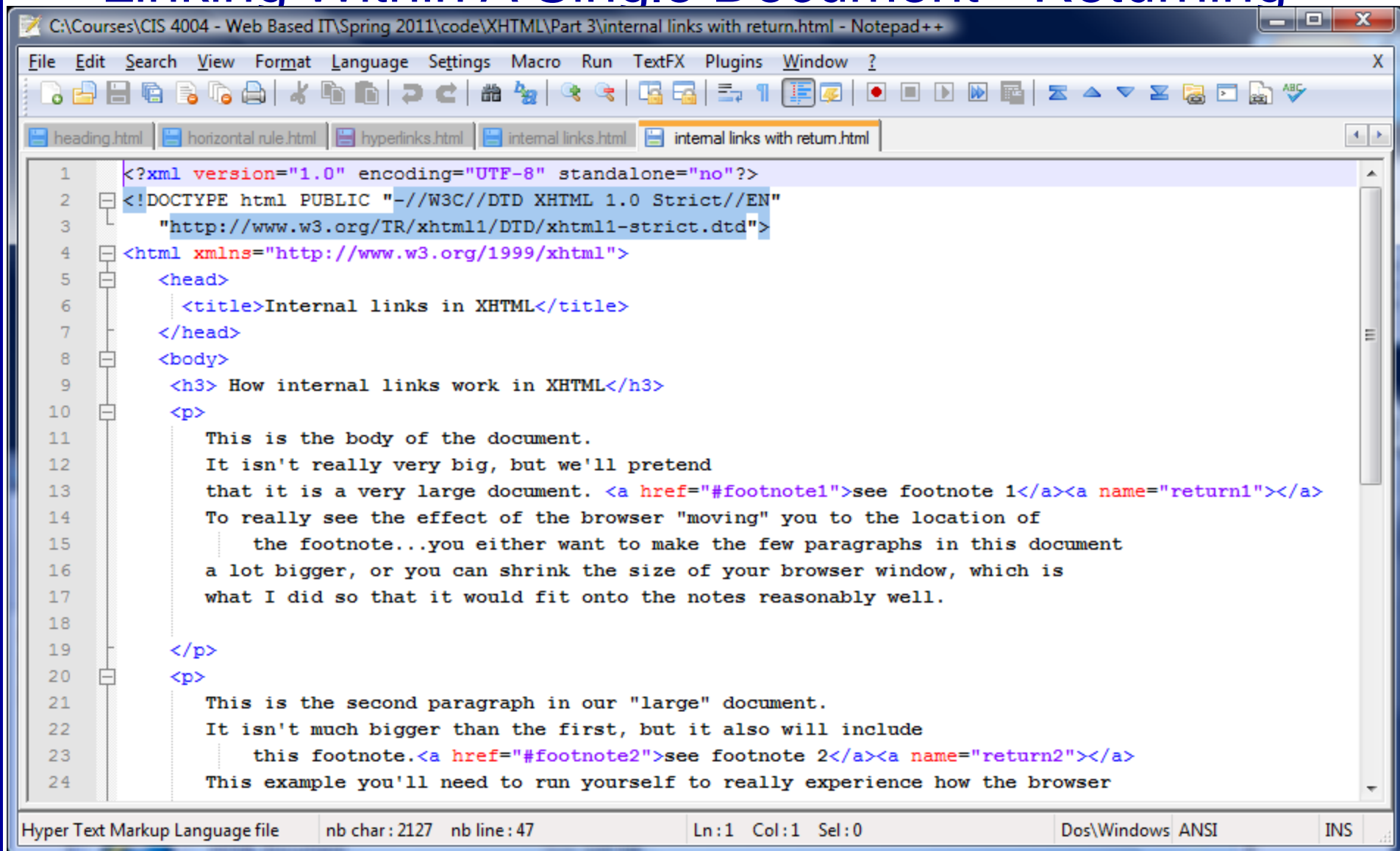
Initial browser window



After clicking footnote #1 link



Linking Within A Single Document - Returning

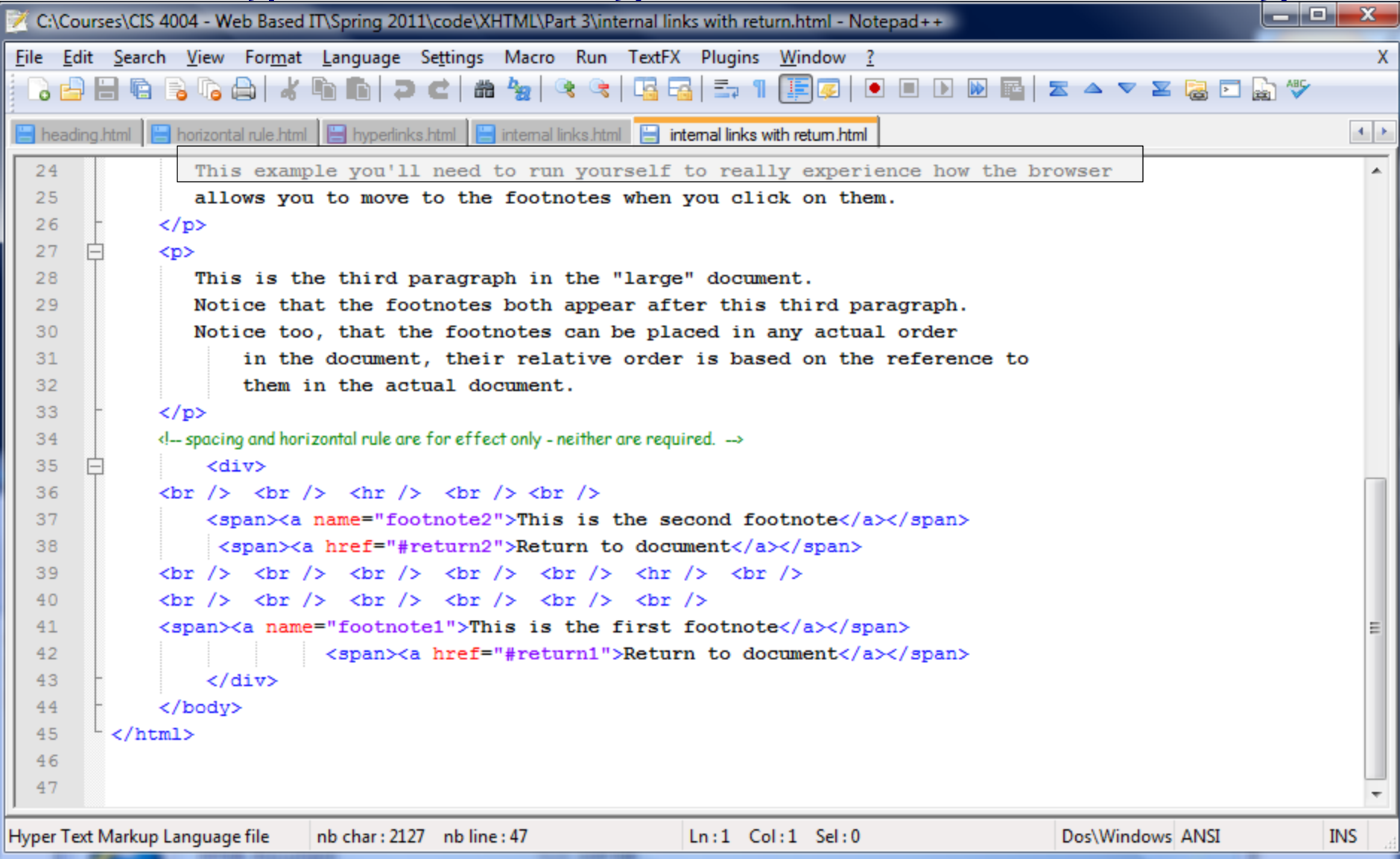


```
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4 <html xmlns="http://www.w3.org/1999/xhtml">
5   <head>
6     <title>Internal links in XHTML</title>
7   </head>
8   <body>
9     <h3>How internal links work in XHTML</h3>
10    <p>
11      This is the body of the document.
12      It isn't really very big, but we'll pretend
13      that it is a very large document. <a href="#footnote1">see footnote 1</a><a name="return1"></a>
14      To really see the effect of the browser "moving" you to the location of
15        the footnote...you either want to make the few paragraphs in this document
16        a lot bigger, or you can shrink the size of your browser window, which is
17        what I did so that it would fit onto the notes reasonably well.
18    </p>
19    <p>
20      This is the second paragraph in our "large" document.
21      It isn't much bigger than the first, but it also will include
22        this footnote.<a href="#footnote2">see footnote 2</a><a name="return2"></a>
23      This example you'll need to run yourself to really experience how the browser
```

Hyper Text Markup Language file nb char : 2127 nb line : 47 Ln:1 Col:1 Sel:0 Dos\Windows ANSI INS



Linking Within A Single Document - Returning



```
24 This example you'll need to run yourself to really experience how the browser
25 allows you to move to the footnotes when you click on them.
26 </p>
27 <p>
28     This is the third paragraph in the "large" document.
29     Notice that the footnotes both appear after this third paragraph.
30     Notice too, that the footnotes can be placed in any actual order
31     in the document, their relative order is based on the reference to
32     them in the actual document.
33 </p>
34 <!-- spacing and horizontal rule are for effect only - neither are required. -->
35     <div>
36     <br /> <br /> <hr /> <br /> <br />
37     <span><a name="footnote2">This is the second footnote</a></span>
38     <span><a href="#return2">Return to document</a></span>
39     <br /> <br /> <br /> <br /> <br /> <hr /> <br />
40     <br /> <br /> <br /> <br /> <br /> <br />
41     <span><a name="footnote1">This is the first footnote</a></span>
42     <span><a href="#return1">Return to document</a></span>
43     </div>
44 </body>
45 </html>
```

Hyper Text Markup Language file nb char : 2127 nb line : 47 Ln : 1 Col : 1 Sel : 0 Dos\Windows ANSI INS



Linking Within A Single Document - Returning

