Tutorial 9: Working with XHTML

Computer Science Department
University of Central Florida

COP 3175 – Internet Applications

Objectives

- Describe the history and theory of XHTML
- Understand the rules for creating valid XHTML documents
- Apply a DTD to an XHTML document
- Understand how to apply the XHTML namespace
- Test an XHTML document under the transitional DTD
- Test an XHTML document under the strict DTD
- Explore the use of character and parsed character data

Tutorial 9 Website

- Wizard Works
  - Company making customized fireworks
  - “Tom”, the head of advertising, designs the website and does a decent job
    - But the site is now old and outdated
  - Tom wants the code to be updated, reflecting current standards and practices
  - Specifically, he wants us to rewrite the code in XHTML rather than HTML
  - And he wants us to find ways to confirm that the code is up to par and meets XHTML standards

Introducing XHTML

- SGML (Standard Generalized Markup Language)
  - Language introduced in the 1980s
    - Describes the structure and content of documents or any type of information readable by machines
  - Device-independent and system-independent
    - documents written in SGML can be used, in theory, on almost any type of device and under almost any type of operating system
  - Bottom line:
    - VERY powerful and flexible markup language

Introducing XHTML

- SGML (Standard Generalized Markup Language)
  - Its strength (power and flexibility) made it a difficult language to learn and apply
    - Official specification is over 150 pages long
    - Covering scenarios even top programmers won't encounter
  - Result:
    - SGML is limited to organizations that can afford the cost and overhead of maintaining complex SGML environment

Introducing XHTML

- HTML as an SGML Application
  - SGML is too complex for the World Wide Web
    - Web page authors need only a simple markup language that is easy to use
  - However, SGML is used to make other markup languages that are based on the SGML architecture
  - One such markup language: HTML
  - Standards get confusing among browsers
  - Can be applied inconsistently

Introducing XHTML

- Problems with HTML
  - Web browsers develop their own unique “flavor” of HTML
    - Providing customers with new and useful features not available in other browsers
  - Example:
    - Netscape introduced the frameset and frame elements
      - Neither were part of the HTML specs at that time
    - Microsoft introduced the iframe for IE
      - Another departure from the official HTML specs at the time
  - Some extensions were welcomed and officiall adopted…others were not

Introducing XHTML

- Problems with HTML
  - Result:
    - Standards get confusing among browsers
      - One kind of HTML for each browser, or, worse, for each version of the browser
    - These innovations/extras certainly increased the scope and power of HTML
      - But they did so at the expense of clarity
    - Web designers could no longer spend all their time on simply making a quality looking website
      - Now they had to spend a lot of time ensuring their sites worked across the various browsers and browser versions

Introducing XHTML

- Problems with HTML
  - Another Problem:
    - HTML can be applied inconsistently (wrongly)
    - Yet, most browsers end up rendering it correctly
  - Example:
    - The following code is wrong:

```
<body>
      <h1>Web Page Title
</body>
```

    - Why?
      - The h1 element has not be closed with an ending </h1> tag
    - Most browsers, however, will still show this correctly

Introducing XHTML

- Problems with HTML
  - Result:
    - Browsers are forgiving of mistakes
      - One may argue that this helps the Web page designer
    - But this causes confusion, as different Web pages end up employing HTML code in markedly different ways
- Solution:
  - It's better for everyone if Web page code adheres to a set of standards for content and structure
  - This solution involved rewriting HTML in terms of XML

Introducing XHTML

- XML and XHTML
  - XML (Extensible Markup Language)
    - Think of this as “SGML light”
    - XML is a language like SGML in that it is used to create markup languages
      - But it doesn’t have the complexity and size of SGML
    - XML has been used to create many markup languages:
      - MathML for mathematical content
      - CML for documenting chemical structures
      - MusicML for describing musical scores
      - and much more

Introducing XHTML

- XML and XHTML
  - XML Example:
    - The following is an excerpt from a MusicML document describing Mozart's Piano Sonata in A Major

```
<work>
        <work-number>K. 331</work-number>
        <work-title>Piano Sonata in A Major</work-title>
</work>
<identification>
        <creator type="composer">Wolfgang Amadeus Mozart</creator>
        <rights>Copyright 2003 Recordare LLC</rights>
</identification>
```

    - Notice that the appearance is very similar to HTML
      - to be expected since they are both markup languages

Introducing XHTML

- XML and XHTML
  - All of this leads to XHTML
  - What is XHTML?
    - It is a reformulation of HTML written in XML
  - The W3C maintains the specifications and standards for XHTML
    - Several versions of XHTML have been released and are in the process of being released
  - The graphic on the next pages summarizes these versions…

Introducing XHTML

Version	Date Released	Description
XHTML 1.0	2001	This version is a reformulation of HTML 4.01 in XML and combines the strength of HTML 4.0 with the power of XML. XHTML brings the rigor of XML to Web pages and provides standards for more robust Web content on a wide range of browser platforms.
XHTML 1.1	2002	A minor update to XHTML 1.0 that allows for modularity and simplifies writing extensions to the language.
XHTML 2.0	not yet released	The latest version, designed to remove most of the presentational features left in HTML. XHTML 2.0 is not backward-compatible with XHTML 1.1.
XHTML 5.0	not yet released	A version of HTML 5.0 written under the specifications of XML; unlike XHTML 2.0, XHTML 5.0 will be backward-compatible with XHTML 1.1.

Introducing XHTML

- XML and XHTML
  - Most widely accepted version is XHTML 1.0
    - specs closely match that of HTML 4.0
  - XHTML 2.0 is still in draft form
    - Not supported by the Web community for many reasons
    - Will not be backwards compatible with earlier versions of XHTML
      - This is a problem, which makes many focus on…
  - XHTML 5.0
    - This, along with HTML 5.0, is were the research and development is currently focused

Introducing XHTML

- Creating an XHTML Document
  - XHTML documents are also considered XML docs
    - As such, the first line of an XHTML file contains a prolog indicating that the document adheres to the syntax rules of XML
  - Form of the XML prolog:

    <?xml version="value" encoding="type" ?>

    - where the version attribute indicates the XML version of the document and the encoding attribute specifies the character encoding
    - XHTML documents are written in XML 1.0
    - Unless you are using special international characters, you can usually set the encoding value to UTF-8

Tutorial 9 Website

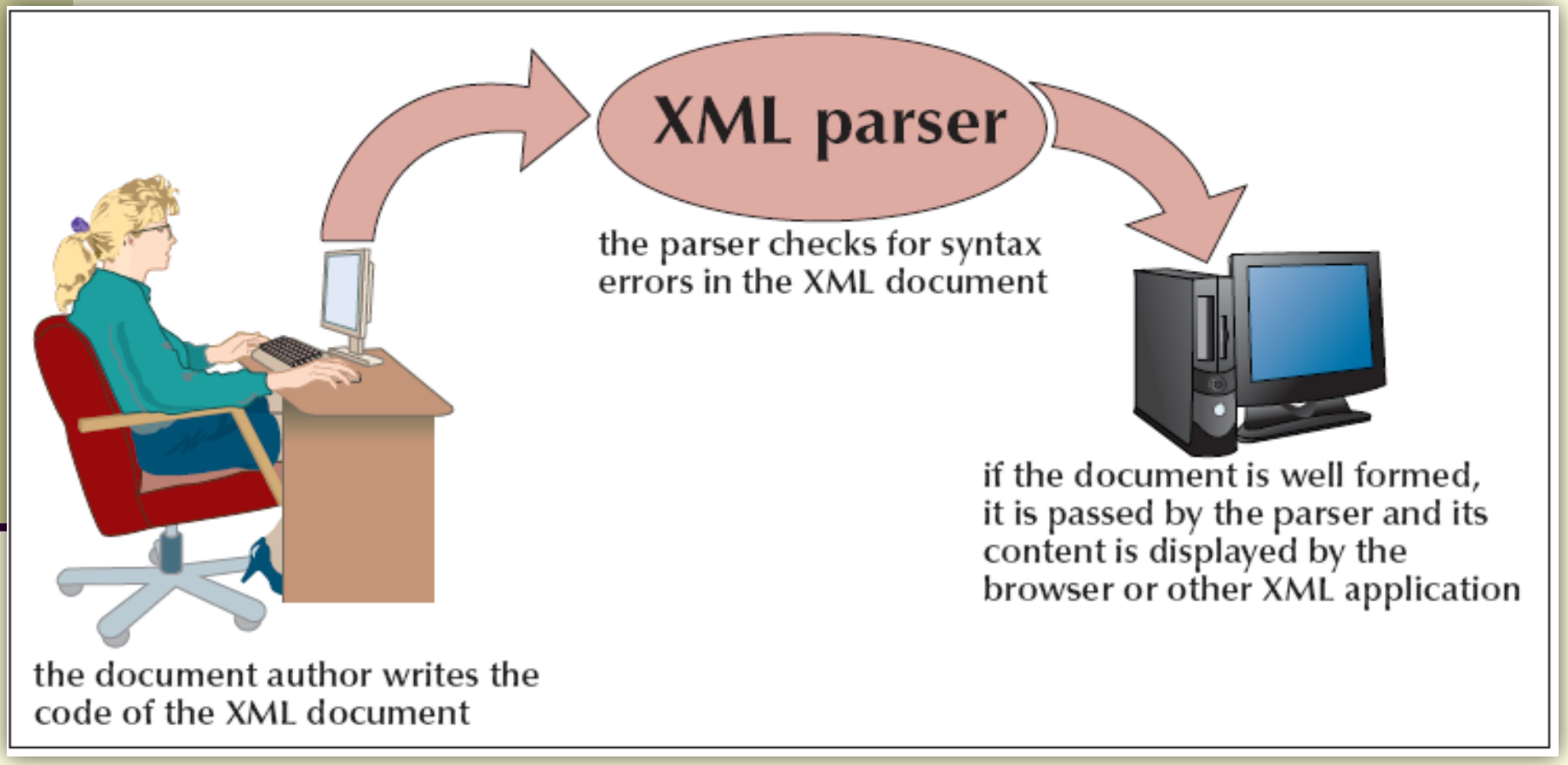
- Insert an XML prolog:
  - First step is pretty easy
    - The XML version is 1.0
    - The encoding is UTF-8
  - We add this to the HTML file:

```
<?xml version="1.0" encoding="UTF-8" ?>
<html>
<head>
```

Creating Well-Formed Documents

- Time to follow the rules!
  - To make XML documents follow specific rules for content and structure, they must be evaluated with an XML parser
    - This is a program that checks the document for errors in syntax and content, and reports the errors it finds
  - An XML document with correct syntax is called a **well-formed** document
  - As mentioned, browsers are forgiving
    - They often accept and render documents that violate HTML syntax as long as the violation isn't too severe
  - However, an XML parser (aka "the Rejecta") will reject any XML document that is not well-formed

Creating Well-Formed Documents



Creating a Well-Formed Document

- Some rules for well-formed XML
  - gotta know “wrong” code from “right” code

Rule	Incorrect	Correct
Element names must be lowercase	`<P>This is a paragraph.</P>`	`<p>This is a paragraph.</p>`
Elements must be properly nested	`<p>This text is <b>bold.</p></b>`	`<p>This text is <b>bold.</b></p>`
All elements must be closed	`<p>This is the first paragraph.` `<p>This is the second paragraph.`	`<p>This is the first paragraph.</p>` `<p>This is the second paragraph.</p>`
Empty elements must be terminated	`This is a line break `	`This is a line break `
Attribute names must be lowercase	`<td ALIGN="right">`	`<td align="right">`
Attribute values must be quoted	`<table width=620>`	`<table width="620">`
Attributes must have values	`<option selected>`	`<option selected="selected">`

Creating a Well-Formed Document

- Another important rule:
  - XHTML documents must also include a single root element that contains all other elements
    - For XHTML, that root element is the html element
- Older Web pages violate a lot of rules!
  - Common problem is **Attribute minimization**
    - Occurs when some attributes lack attribute values
  - XHTML doesn't allow attribute minimization
  - So you have to be aware of how to correct the code, making it XHTML compliant

Attribute minimization in HTML and XHTML

HTML	XHTML
compact	compact="compact"
checked	checked="checked"
declare	declare="declare"
readonly	readonly="readonly"
disabled	disabled="disabled"
selected	selected="selected"
defer	defer="defer"
ismap	ismap="ismap"
nohref	nohref="nohref"
noshade	noshade="noshade"
nowrap	nowrap="nowrap"
multiple	multiple="multiple"
noresize	noresize="noresize"

Creating a Well-Formed Document

- Attribute Minimization:
  - Example:

    ```
    <input type="radio" checked>
    ```

  - In XHTML, this would be rewritten as:

    ```
    <input type="radio" checked="checked" />
    ```

    - So we added the word "checked" as a value of checked
    - And we added the forward slash to properly terminate the <input> element

  - Failure to make these changes will cause the XHTML document to be rejected as not well-formed

Creating Valid XHTML Documents

- XHTML documents must also be valid
  - A valid document is one that is well-formed and one that contains only approved elements, attributes, and other features of the language
  - Example:

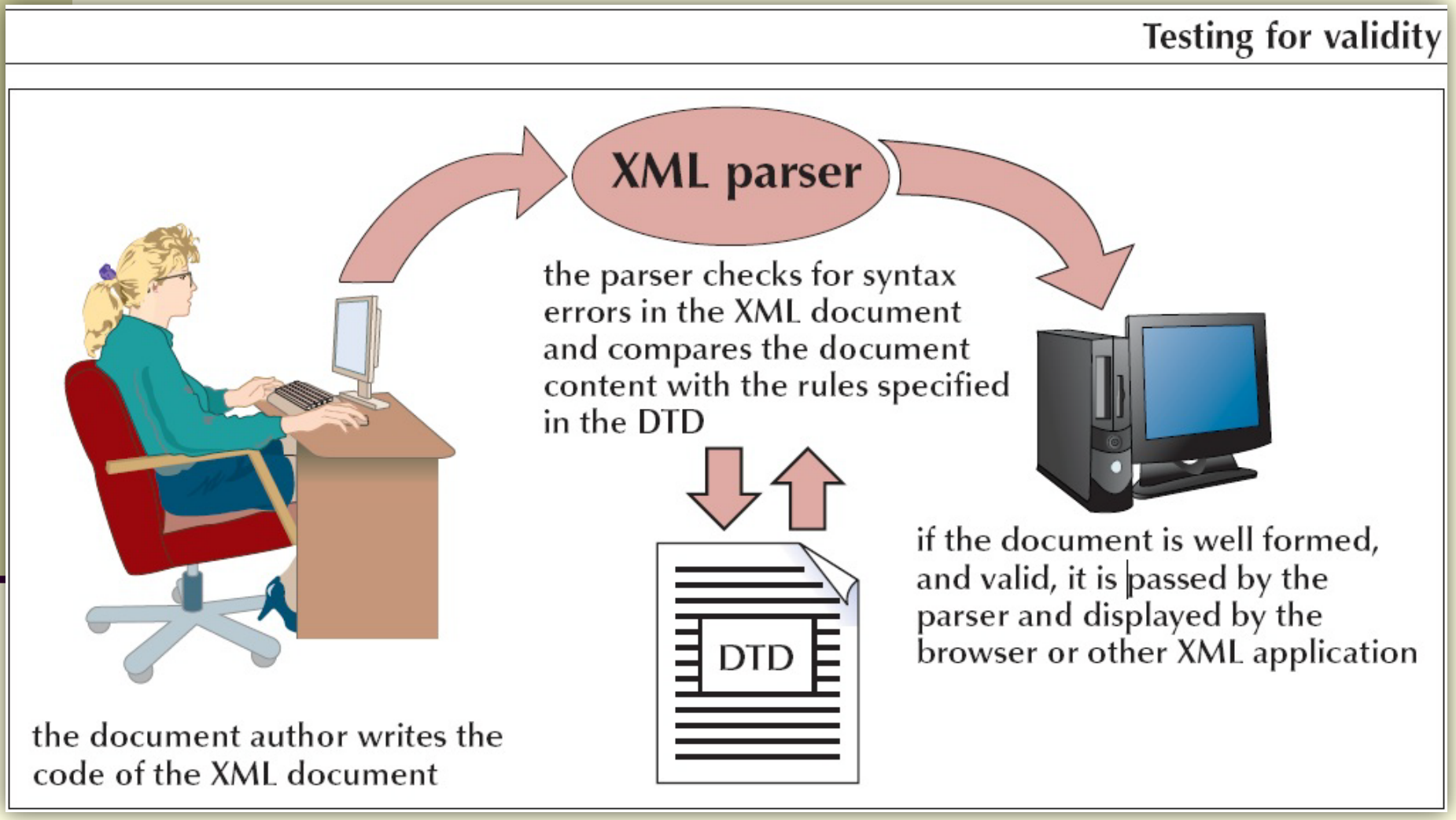
```
<body>
    <mainhead>Title of Page</mainhead>
<body>
```

    - This code is well-formed, because it complies with the syntax rules of XML
    - But, it is not valid, because XHTML does not support the <mainhead> tag

Creating Valid XHTML Documents

- DTD
  - The developers of an XML-based language create a collection of rules that specify the correct content and structure for a document
  - These rules are collectively called the DTD
    - document type definition
  - An XML parser tests the content of the document against the rules of the DTD
    - If it doesn't follow the rules, the parser rejects the document as not valid

Creating Valid XHTML Documents



Creating Valid XHTML Documents

- Transitional, Frameset, and Strict DTDs
  - There are many different DTDs associated with HTML and XHTML documents
    - Some represent older versions of HTML in case you needed to create a document valid only for HTML 2.0
  - For most purposes, you'll focus on three DTDs that are used with XHTML 1.0
    - transitional
    - frameset
    - strict

Creating Valid XHTML Documents

- Transitional DTD
  - Supports many of the presentational features of HTML
    - including deprecated elements and attributes
  - This DTD is best used for older documents that contain deprecated features
  - So if you need support for older browsers, you should use the transitional DTD

Creating Valid XHTML Documents

- Frameset DTD
  - Used for documents containing frames
  - Also supports deprecated elements and attributes
  - So if you need to support older browsers, while at the same time using a framed Web site, you should use the frameset DTD

Creating Valid XHTML Documents

- Strict DTD
  - Does not allow any presentational features or deprecated HTML elements and attributes
  - Also does not support frames or inline frames
  - Best suited for documents that must strictly conform to the latest standards
  - So if you need to support more current browsers and want to weed out any use of deprecated features, and if you don't need to support frames, you should use the strict DTD

Creating Valid XHTML Documents

- Elements **not** allowed under the strict DTD:
  - applet
  - basefont
  - center
  - dir
  - font
  - frame
  - frameset
  - iframe
  - isindex
  - menu
  - noframes
  - s
  - strike
  - u

Creating Valid XHTML Documents

- Strict DTD
  - The strict DTD also enforces a particular structure on the document
  - Example:
    - You cannot place a block level element within an inline element
    - So if you were using the <a> tag to make a link, you couldn't use the <p> tag to put a paragraph inside those <a> tags

Creating Valid XHTML Documents

Child elements prohibited under the strict DTD

Element	Prohibited Children
inline element	block-level elements
body	a, abbr, acronym, b, bdo, big, br, button, cite, code, dfn, em, i, img, input, kbd, label, map, object, q, samp, select, small, span, strong, sub, sup, textarea, tt, var
button	button, form, fieldset, iframe, input, isindex, label, select, textarea
blockquote	a, abbr, acronym, b, bdo, big, br, button, cite, code, dfn, em, i, img, input, kbd, label, map, object, q, samp, select, small, span, strong, sub, sup, textarea, tt, var
form	a, abbr, acronym, b, bdo, big, br, cite, code, dfn, em, form, i, img, kbd, map, object, q, samp, small, span, strong, sub, sup, tt, var
label	label
pre	big, img, object, small, sub, sup
all other page elements	big, small

Creating Valid XHTML Documents

- Strict DTD
  - Example:

```
<body>
   <img src="logo.jpg" alt="Wizard Works" />
</body>
```

    - That code would be rejected because an inline image was a direct child of the body element
  - Corrected code:

```
<body>
   <p><img src="logo.jpg" alt="Wizard Works" /></p>
</body>
```

Creating Valid XHTML Documents

- All DTDs:
  - All three DTDs require the following elements be present in every valid XHTML document:
    - html, head, title, and body
  - The html, head, and body elements are usually expected under HTML
  - However, XHTML requires that every document include the title element as well
    - If the title element is missing “the Rejecta” will reject your document

Creating Valid XHTML Documents

- The Valid Use of Attributes
  - DTDs also include rules for attributes and their use
  - Under the strict DTD, deprecated attributes are not allowed
  - So you must know these elements and their corresponding attributes…

Attributes Prohibited in strict DTD

Element	Prohibited Attributes
a	target
area	target
base	target
body	alink, bgcolor, link, text, vlink,
br	clear
caption	align
div	align
dl	compact
form	name, target
h*n*	align
hr	align, noshade, size, width
img	align, border, hspace, name, vspace
input	align
li	type, value
link	target
map	name
object	align, border, hspace, vspace
ol	compact, start
p	align
pre	width
script	language
table	align, bgcolor
td	bgcolor, height, nowrap, width
th	bgcolor, height, nowrap, width
tr	bgcolor
ul	type, compact

Creating Valid XHTML Documents

- The Valid Use of Attributes
  - Many of these attributes are the so-called presentational attributes
    - The define how the browser should render the elements
  - Why do you think they are prohibited?
    - Remember the goal we always mention of Web design
    - Separate content from structure/appearance
    - Presentation/design should all be done in CSS
  - The strict DTD enforces this

Creating Valid XHTML Documents

- The Valid Use of Attributes
  - The strict DTD requires the use of the id attribute in place of the name attribute
    - These tags:

```
<a name="top">
<form name="order">
<img name="logo">
<map name="parkmap">
```

    - Would have to be rewritten in strict XHTML as:

```
<a id="top">
<form id="order">
<img id="logo">
<map id="parkmap">
```

Creating Valid XHTML Documents

- The Valid Use of Attributes
  - The strict DTD does not support the target attribute
    - Problem: you cannot open links in secondary browser windows if you want your code compliant with XHTML
  - So clearly some attributes are prohibited
  - Also, other attributes are required
    - So given an element, in order for the code to be valid XHTML, that element may require certain attributes
    - The following graphic lists required XHTML attributes

Required XHTML Attributes

Element	Required Attributes
applet	height, width
area	alt
base	href
basefont	size
bdo	dir
form	action
img	alt, src
map	id
meta	content
optgroup	label
param	name
script	type
style	type
textarea	cols, rows

Creating Valid XHTML Documents

- Taking it all in…
  - A LOT of rules!
    - The list is certainly long and perhaps onerous
    - But they reflect good coding practice
  - Using the rules helps guarantees a certain level of quality in the syntax of your Web site
    - You won't have an page without a title
    - You won't have an inline image without alt text
    - And if you make a mistake in syntax, using the DTD enables you to test your document to correct mistakes

Brief Interlude: FAIL Pics

Daily UCF Bike Fail

Creating Valid XHTML Documents

- Inserting the DOCTYPE Declaration
  - You need to specify which DTD should be used by your XHTML document
  - To do this, you add a DOCUTYPE declaration
    - This tells XML parsers what DTD is associated with the document and allows the parsers to work accordingly
    - **Syntax:**
    - **<!DOCTYPE root type “id” “url”>**
      - where *root* is the name of the root element of the document
      - *type* identifies the type of the DTD (PUBLIC or SYSTEM)
      - *id* is an id associated with the DTD
      - and *url* is the location of an external file containing the DTD rules

Creating Valid XHTML Documents

DOCTYPE declarations for different versions of HTML and XHTML

DTD	DOCTYPE
HTML 2.0	<!DOCTYPE html PUBLIC "-//IETF//DTD HTML 2.0//EN">
HTML 3.2	<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
HTML 4.01 strict	<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
HTML 4.01 transitional	<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
HTML 4.01 frameset	<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">
XHTML 1.0 strict	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
XHTML 1.0 transitional	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
XHTML 1.0 frameset	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
XHTML 1.1	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

Tutorial 9 Website

- Insert a DOCTYPE declaration
  - Tom points out that the site has a lot of deprecated code
    - Most of which he'll fix
    - But some of which he may not get to
    - So he wants the site to work with the transitional DTD

```
<?xml version="1.0" encoding="UTF-8" ?>

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

<html>
<head>
```

Setting the XHTML Namespace

- Adding a namespace declaration
  - What is a namespace?
  - Remember:
    - XHTML is only one of hundreds of languages built on the foundation of XML
    - For example, another XML-based language, MathML, is sued for documents containing mathematical content, symbols, equations, etc.
  - Example:
    - A math professor wants to make a Web site
    - He really needs may want some pages using XHTML and others using MathML (or both)

Setting the XHTML Namespace

- Adding a namespace declaration
  - What is a namespace?
    - XML (and through it, XHTML) allows elements and attributes, from several different XML-based languages, to be combined within a single document
    - So our math professor could combine elements of XHTML and MathML in one document
  - The problem is that you need a way of identifying which elements go with which language
  - This is done by using namespaces

Setting the XHTML Namespace

- Adding a namespace declaration
  - So finally, what is a namespace?
    - A namespace is a unique identifier for elements and attributes originating from a particular document type (like XHTML or MathML)
  - Two types of namespaces:
    - default and local (we only focus on default)
    - A default namespace is the namespace that is assumed to be applied to a root element and any element within it
    - This includes, by default, any element within the document

Setting the XHTML Namespace

- Adding a namespace declaration
  - To declare a namespace, you add the xmlns (XML namespace) attribute

    `<root xmlns="namespace">`

    to the markup tag for the documents root element
    - where *root* is the name of the root element and *namespace* is the namespace id
  - Every XML-based language has a namespace id
  - For XHTML, you would add the following:

    `<html xmlns="http://www.w3.org/1999/xhtml">`
    - this namespace uniquely identifies the doc as XHTML

Testing under XHTML Transitional

- Time to test our pages!
  - In order to confirm that your Web pages follow standards, you must submit them to an XML parser
  - There are several parsers available on the Web, with, arguably, the most famous on at the W3C:
    - http://validator.w3.org/
    - You can either upload files or simply provide a link the specific page

Testing under XHTML Transitional

- Errors Reported:
  - Note:
    - Many errors may end up being reported
    - This doesn't mean that all of them are true errors
    - In some cases, one mistake may result in multiple errors being reported
    - And fixing that one mistake can result in several of the "errors" being resolved
  - Suggestion:
    - Start with the most obvious errors first, and hopefully it will help get rid of others in the process

Testing under XHTML Transitional

- Testing under different DTDs
  - Once your page works under the transitional DTD, it's a good idea to get it working under strict DTD
  - So you must change the DOCTYPE to reflect your choice of strict DTD:

```
<?xml version="1.0" encoding="UTF-8" ?>

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

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
```

  - And then revalidate!
  - Note: expect a lot more errors as well!

Testing under XHTML Transitional

- Testing under different DTDs
  - Once your page works under the transitional DTD, it's a good idea to get it working under strict DTD
  - So you must change the DOCTYPE to reflect your choice of strict DTD:

```
<?xml version="1.0" encoding="UTF-8" ?>

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

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
```

  - And then revalidate!
  - Note: expect a lot more errors as well!

Testing under XHTML Transitional

- Passing Validation:
  - Once the page has passed validation, you may want to make note of this on your Web page
  - The W3C provides code (a link and a picture) that you can paste into your document to advertise that you beat “the Rejecta”
  - NOTE:
    - This is REQUIRED for the final project to have BOTH validation icons (for HTML and CSS) and to have those icons working

Tutorial 9 Website

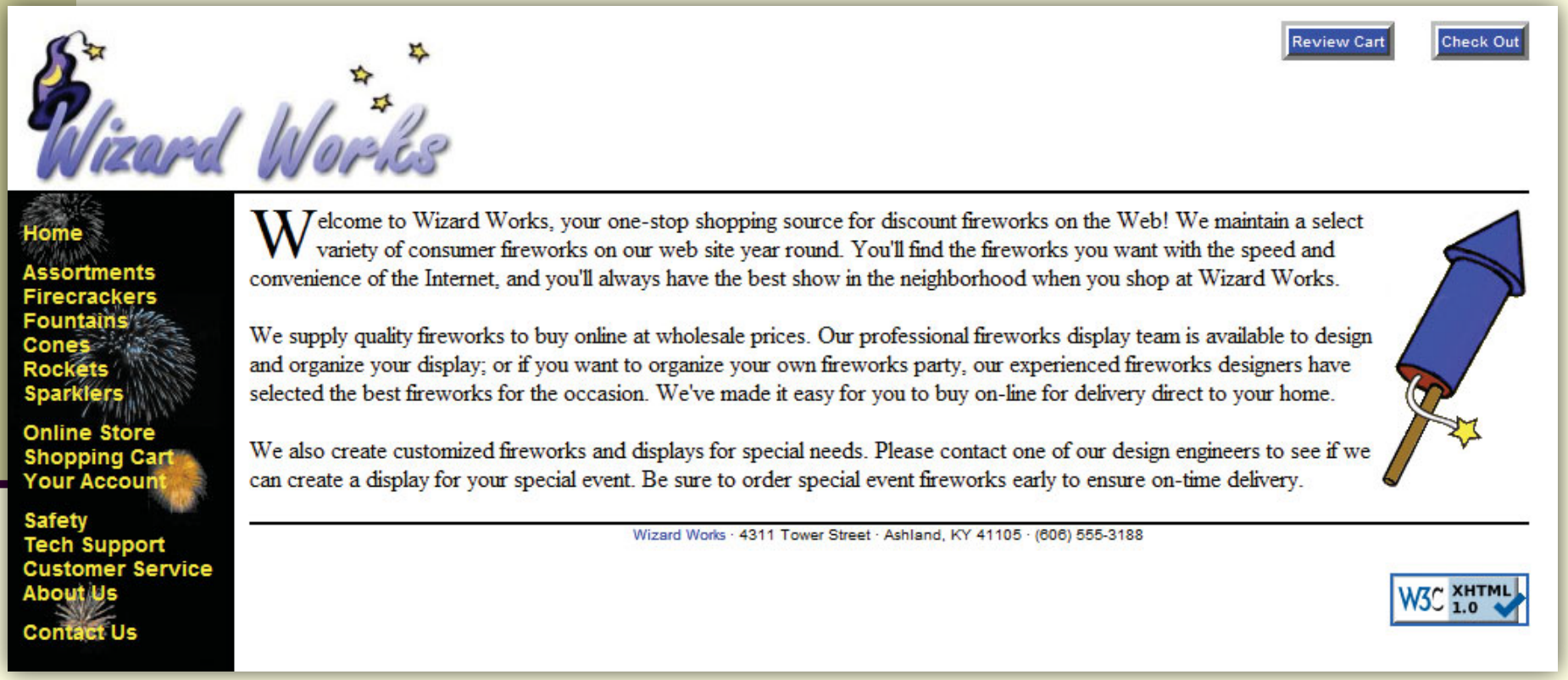
- Go through Part 2 of the tutorial and fix Tom's site
  - Once it is valid, we add the W3C valid icon:

```
    <address>
       <span style="color: blue">Wizard Works</span> &#183;
       4311 Tower Street &#183;
       Ashland, KY  41105 &#183;
       (606) 555-3188
    </address>

    <p>
       <a href="http://validator.w3.org/check?uri=referer">
          <img src="http://www.w3.org/Icons/valid-xhtml10-blue"
               alt="Valid XHTML 1.0 Strict" height="31" width="88" />
       </a>
    </p>

</div>
```

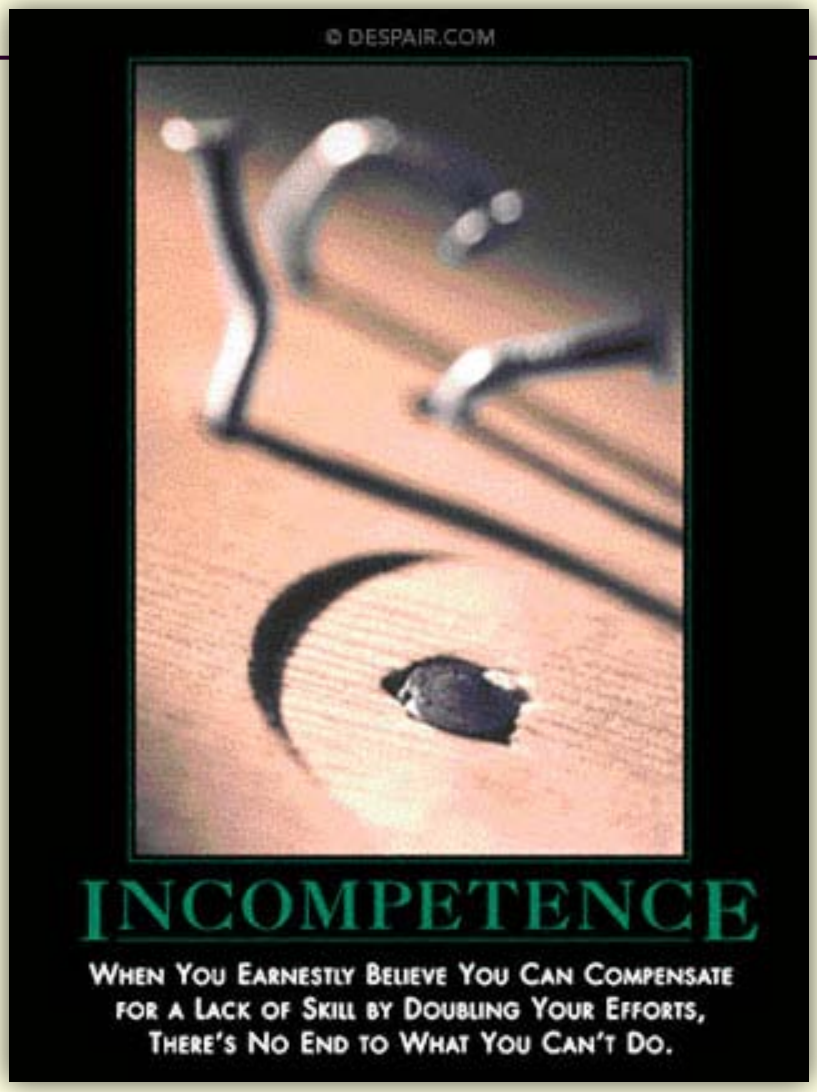
Tutorial 9 Website – Result



Tutorial 9: Working with XHTML

WASN'T THAT MONUMENTAL!

Daily Demotivator



Tutorial 9: Working with XHTML

Computer Science Department
University of Central Florida

COP 3175 – Internet Applications