

CSI 3140

WWW Structures, Techniques and Standards

**Markup Languages:  
XHTML 1.0**

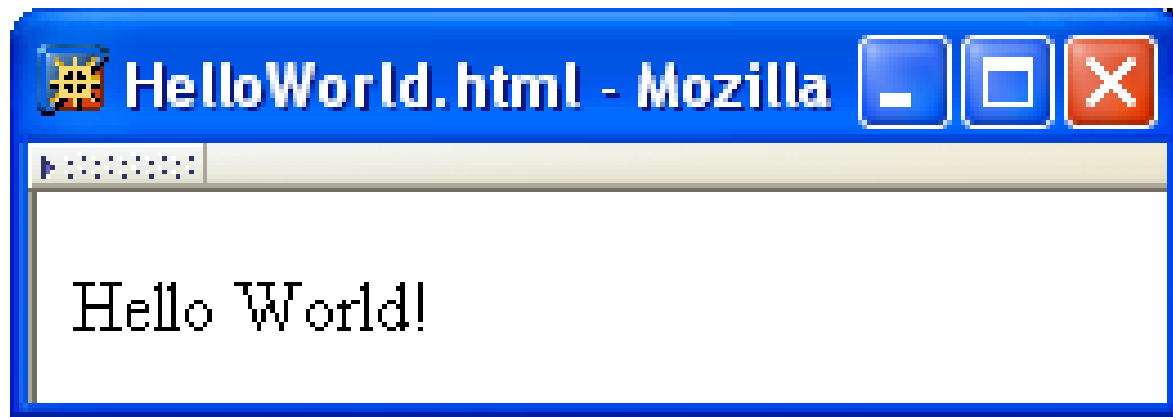
# HTML “Hello World!”

Document  
Type  
Declaration

```
<!DOCTYPE html
    PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>
      HelloWorld.html
    </title>
  </head>
  <body>
    <p>
      Hello World!
    </p>
  </body>
</html>
```

Document  
Instance

# HTML “Hello World”

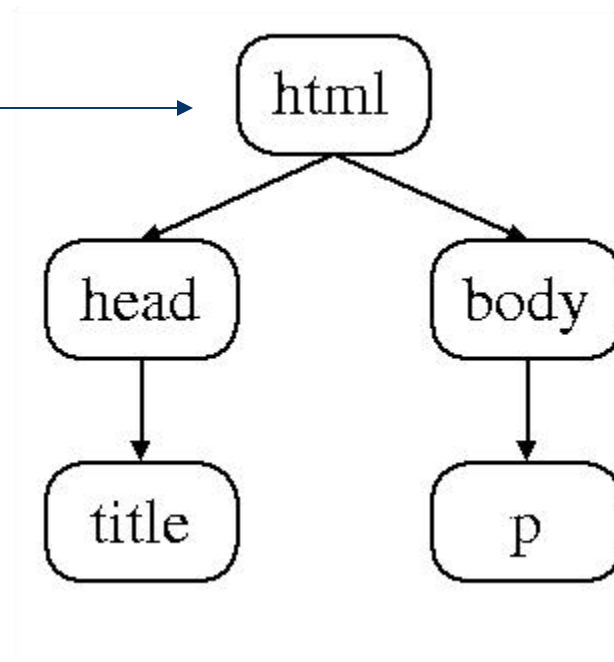


# HTML Tags and Elements

- ◆ Any string of the form `< ... >` is a *tag*
- ◆ All tags in document instance of Hello World are either **end tags** (begin with `</`) or **start tags** (all others)
  - Tags are an example of markup, that is, text treated specially by the browser
  - Non-markup text is called character data and is normally displayed by the browser
- ◆ String at beginning of start/end tag is an **element name**
- ◆ Everything from start tag to matching end tag, including tags, is an **element**
  - **Content** of element excludes its start and end tags

# HTML Element Tree

Root  
Element



# HTML Root Element

- ◆ Document type declaration specifies name of root element:

```
<!DOCTYPE html
```

- ◆ Root of HTML document must be *html*

- ◆ XHTML 1.0 (standard we will follow) requires that this element contain the xml namespace *xmlns* attribute specification (name/value pair)

↓

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

# HTML head and body Elements

- ◆ The *body* element contains information displayed in the browser client area
- ◆ The *head* element contains information used for other purposes by the browser:
  - title (shown in title bar of browser window)
  - scripts (client-side programs)
  - style (display) information
  - etc.

# HTML History

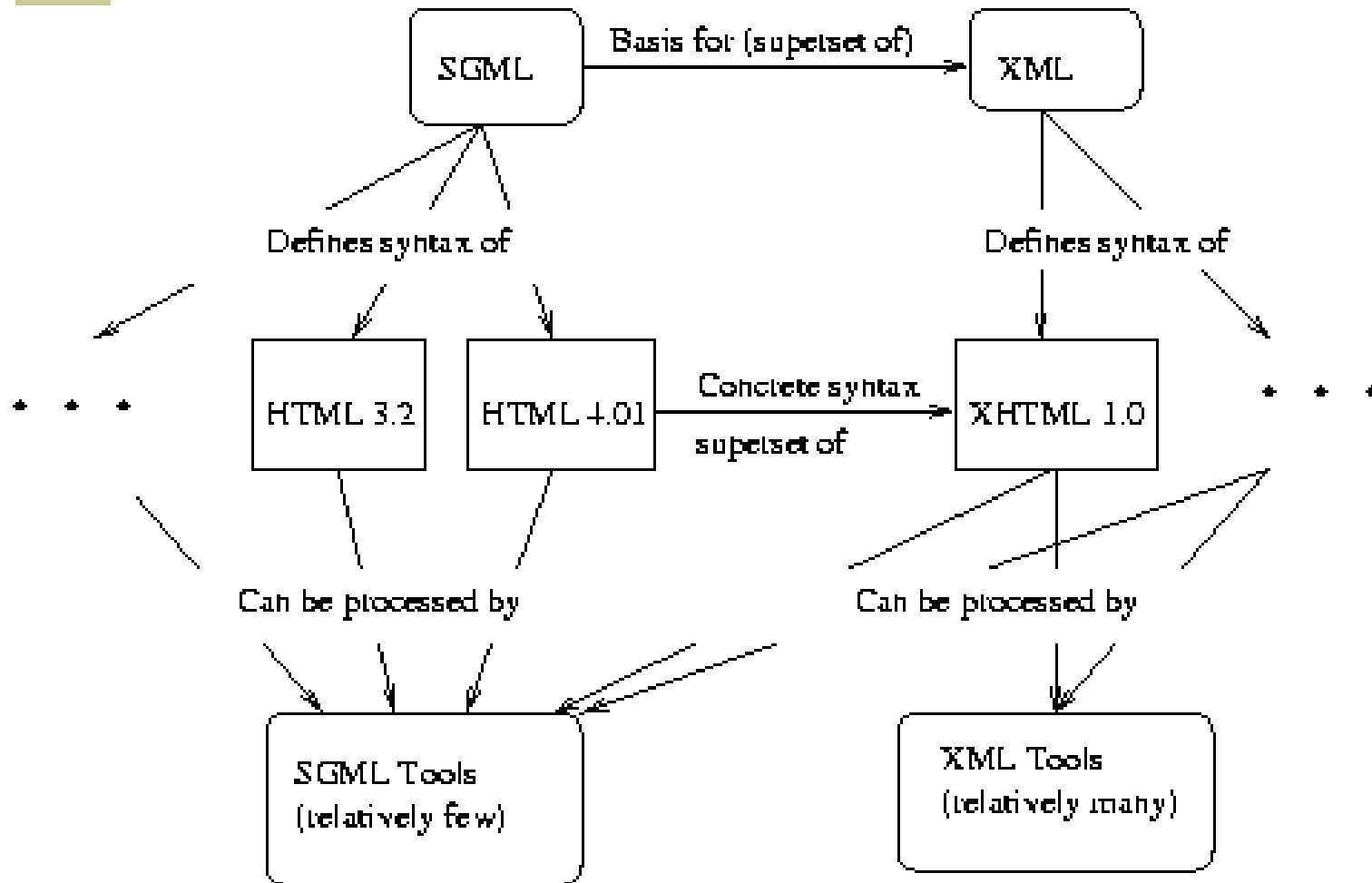
- ◆ 1990: HTML invented by Tim Berners-Lee
- ◆ 1993: Mosaic browser adds support for images, sound, video to HTML
- ◆ 1994-~1997: “**Browser wars**” between Netscape and Microsoft, HTML defined operationally by browser support
- ◆ ~1997-present: Increasingly, World-Wide Web Consortium (W3C) recommendations define HTML



# HTML Versions

- ◆ HTML 4.01 (Dec 1999) syntax defined using Standard Generalized Markup Language (SGML)
- ◆ XHTML 1.0 (Jan 2000) syntax defined using Extensible Markup Language (XML)
- ◆ Primary differences:
  - HTML allows some tag omissions (e.g., end tags)
  - XHTML element and attribute names are lower case (HTML names are case-insensitive)
  - XHTML requires that attribute values be quoted

# SGML and XML



# HTML “Flavors”

- ◆ For HTML 4.01 and XHTML 1.0, the document type declaration can be used to select one of three “flavors”:
  - **Strict**: W3C ideal
  - **Transitional**: Includes deprecated elements and attributes (W3C recommends use of *style sheets* instead)
  - **Frameset**: Supports frames (subwindows within the client area)

# HTML Frameset

The screenshot shows a Mozilla browser window displaying the Java 2 Platform SE v1.4.2 documentation for the `Applet` class. The window title is "Applet (Java 2 Platform SE v1.4.2) - Mozilla". The page layout includes a left-hand navigation pane and a main content area.

**Navigation Pane (Left):**

- Java™ 2 Platform Std. Ed. v1.4.2
- All Classes
- Packages
  - [java.applet](#)
  - [java.awt](#)
- java.applet
  - Interfaces
    - [AppletContext](#)
    - [AppletStub](#)
    - [AudioClip](#)
  - Classes
    - [Applet](#)

**Main Content Area:**

Overview Package **Class** Use Tree Deprecated Index Help *Java™ 2 Platform Std. Ed. v1.4.2*

PREV CLASS [NEXT CLASS](#) [FRAMES](#) [NO FRAMES](#)

SUMMARY: [NESTED](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#) DETAIL: [FIELD](#) | [CONSTR](#) | [METHOD](#)

---

java.applet

## Class Applet

[java.lang.Object](#)

- [java.awt.Component](#)
  - [java.awt.Container](#)
    - [java.awt.Panel](#)
      - java.applet.Applet**

All Implemented Interfaces:  
[Accessible](#), [ImageObserver](#), [MenuContainer](#), [Serializable](#)

Screen shots are reproduced by permission of Sun Microsystems Inc. All rights reserved.

# HTML Document Type Declarations

- ◆ 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 Frameset:

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

- ◆ HTML 4.01 Transitional:

```
<!DOCTYPE HTML  
PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"  
    "http://www.w3.org/TR/html4/loose.dtd">
```

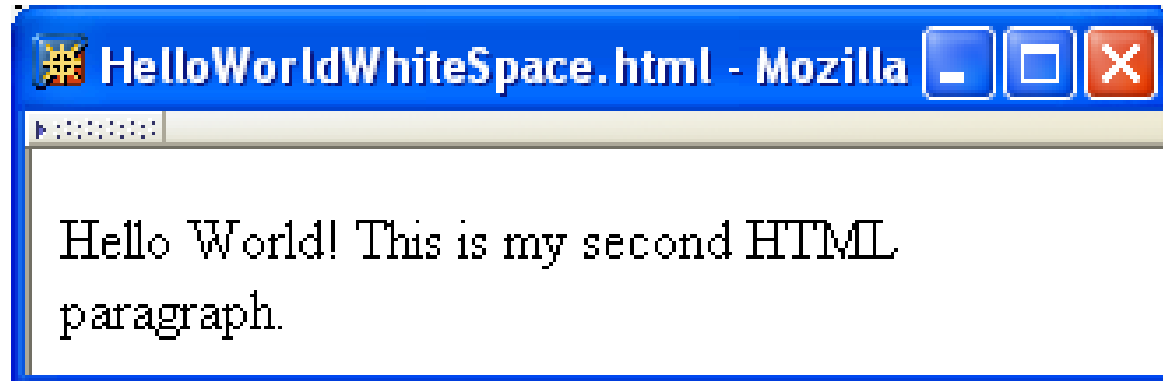
# XHTML White Space

- ◆ Four white space characters: carriage return, line feed, space, horizontal tab
- ◆ Normally, character data is normalized:
  - All white space is converted to space characters
  - Leading and trailing spaces are trimmed
  - Multiple consecutive space characters are replaced by a single space character

# XHTML White Space

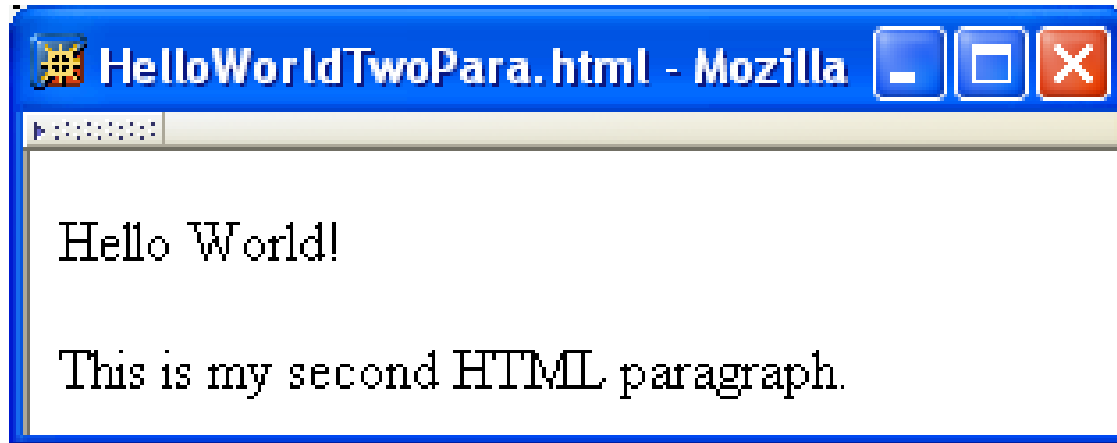
```
<body>
  <p>
    Hello World!

    This is my second HTML paragraph.
  </p>
</body>
```



# XHTML White Space

```
<p>  
  Hello World!  
</p>  
<p>  
  This is my second HTML paragraph.  
</p>
```





# Unrecognized HTML Elements

```
<!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>
    <titl>
      HelloWorldBadElt.html
    </title>
  </head>
  <body>
    <p>
      Hello World!
    </p>
  </body>
</html>
```

Misspelled  
element name →

# Unrecognized HTML Elements

Belongs here

*title* character data



# Unrecognized HTML Elements



*title* character  
data

Displayed  
here

# Unrecognized HTML Elements

- ◆ Browsers ignore tags with unrecognized element names, attribute specifications with unrecognized attribute names
  - Allows evolution of HTML while older browsers are still in use
- ◆ Implication: an HTML document may have errors even if it displays properly
- ◆ Should use an HTML validator to check syntax

# Unrecognized HTML Elements

## Example for non-frame browsers (old)

```
<HTML>
  <HEAD>
    <TITLE>A simple frameset document</TITLE>
  </HEAD>
  <FRAMESET cols="20%, 80%">
    <FRAME src="contents_of_frame1.html" />
    <FRAME src="contents_of_frame2.html" />
    <NOFRAMES>
      <P>This doc contains frames</P>
    </NOFRAMES>
  </FRAMESET>
</HTML>
```

# HTML References

- ◆ Since `<` marks the beginning of a tag, how do you include a `<` in an HTML document?
- ◆ Use markup known as a reference
- ◆ Two types:
  - Character reference specifies a character by its Unicode code point
    - For `<`, use `&#60;`; or `&#x3C;`; or `&#x3c;`
  - Entity reference specifies a character by an HTML-defined name
    - For `<`, use `&lt;`

# HTML References

TABLE 2.2: Example entity and character references.

Character	Entity Reference	Character Reference (decimal)
<	&lt;	&#60;
>	&gt;	&#62;
&	&amp;	&#38;
"	&quot;	&#34;
'	&apos;	&#39;
©	&copy;	&#169;
ñ	&ntilde;	&#241;
α	&alpha;	&#945;
∀	&forall;	&#8704;

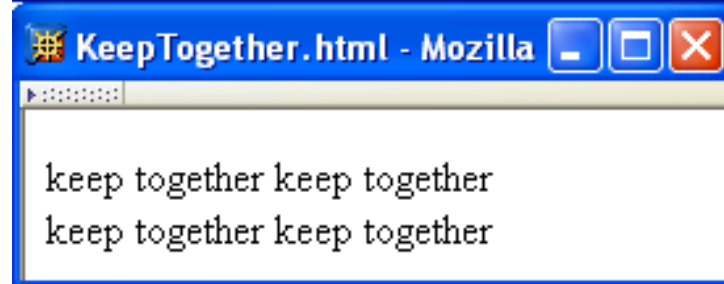
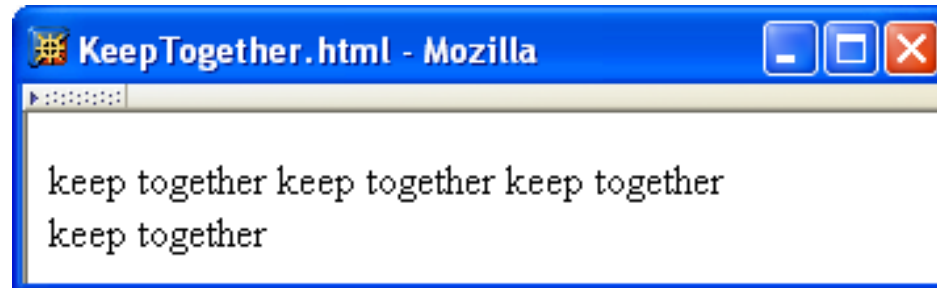
# HTML References

- ◆ Since `<` and `&` begin markup, within character data or attribute values these characters must *always* be represented by references (normally `&lt;`; and `&amp;`;) )
- ◆ Good idea to represent `>` using reference (normally `&gt;`;) )
  - Provides consistency with treatment of `<`
  - Avoids accidental use of the reserved string `]]>`



# HTML References

- ◆ Non-breaking space ( *&nbsp;* ) produces space but counts as part of a word
  - Ex: keep together keep together ...

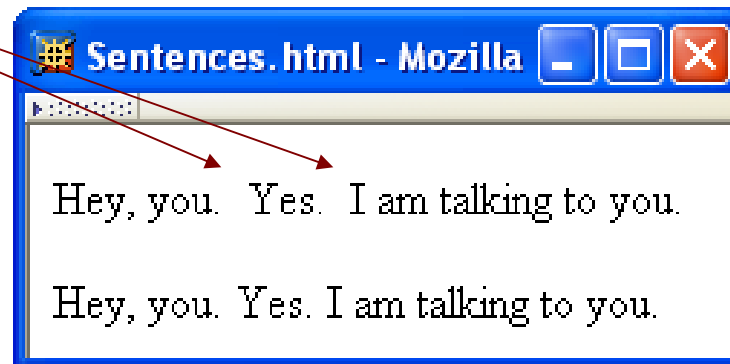


# HTML References

- ◆ Non-breaking space often used to create multiple spaces (not removed by normalization)

```
<p>  
  Hey, you.&nbsp; Yes.&nbsp; I am talking to you.  
</p>  
<p>  
  Hey, you.  Yes.  I am talking to you.  
</p>
```

&nbsp; + space  
displays as two  
spaces



# XHTML Attribute Specifications

## ◆ Example:

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

## ◆ Syntax:

- Valid attribute names specified by HTML recommendation (or XML, as in `xml:lang`)
- Attribute values must be quoted (matching single or double quotes)
- Multiple attribute specifications are space-separated, order-independent

# XHTML Attribute Values

- ◆ Can contain embedded quotes or references to quotes

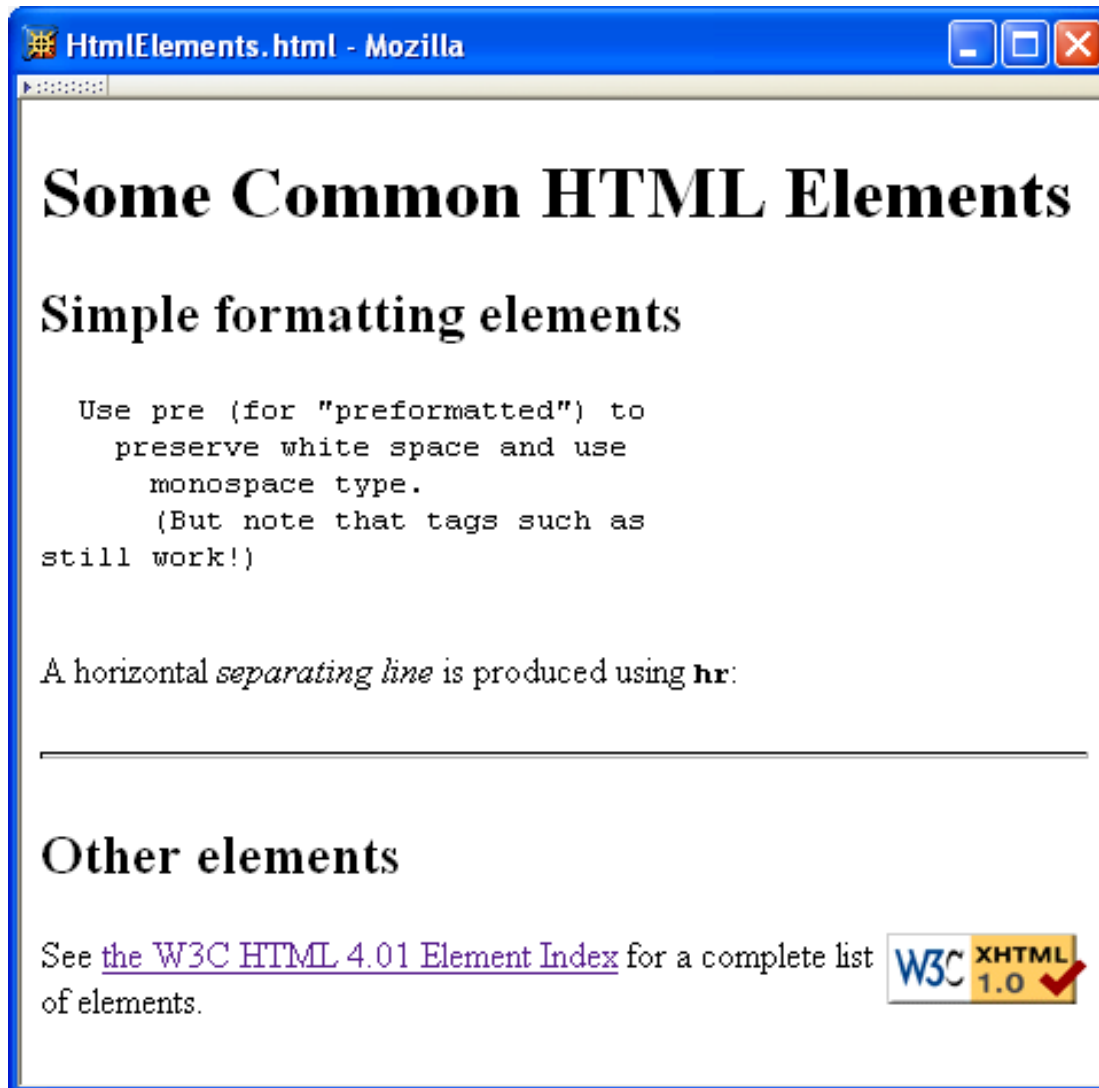
✓ value = "Ain't this grand!"

✓ value = "He said, &quot;She said&quot;, then sighed."

✗ value = "He said, "She said", then sighed."

- ◆ May be normalized by browser
  - Best to normalize attribute values yourself for optimal browser compatibility

# Common HTML Elements



The screenshot shows a Mozilla browser window titled "HtmlElements.html - Mozilla". The page content is as follows:

## Some Common HTML Elements

### Simple formatting elements


Use `pre` (for "preformatted") to preserve white space and use monospace type.  
(But note that tags such as `hr` still work!)

A horizontal *separating line* is produced using `hr`:

---

### Other elements

See [the W3C HTML 4.01 Element Index](#) for a complete list of elements.



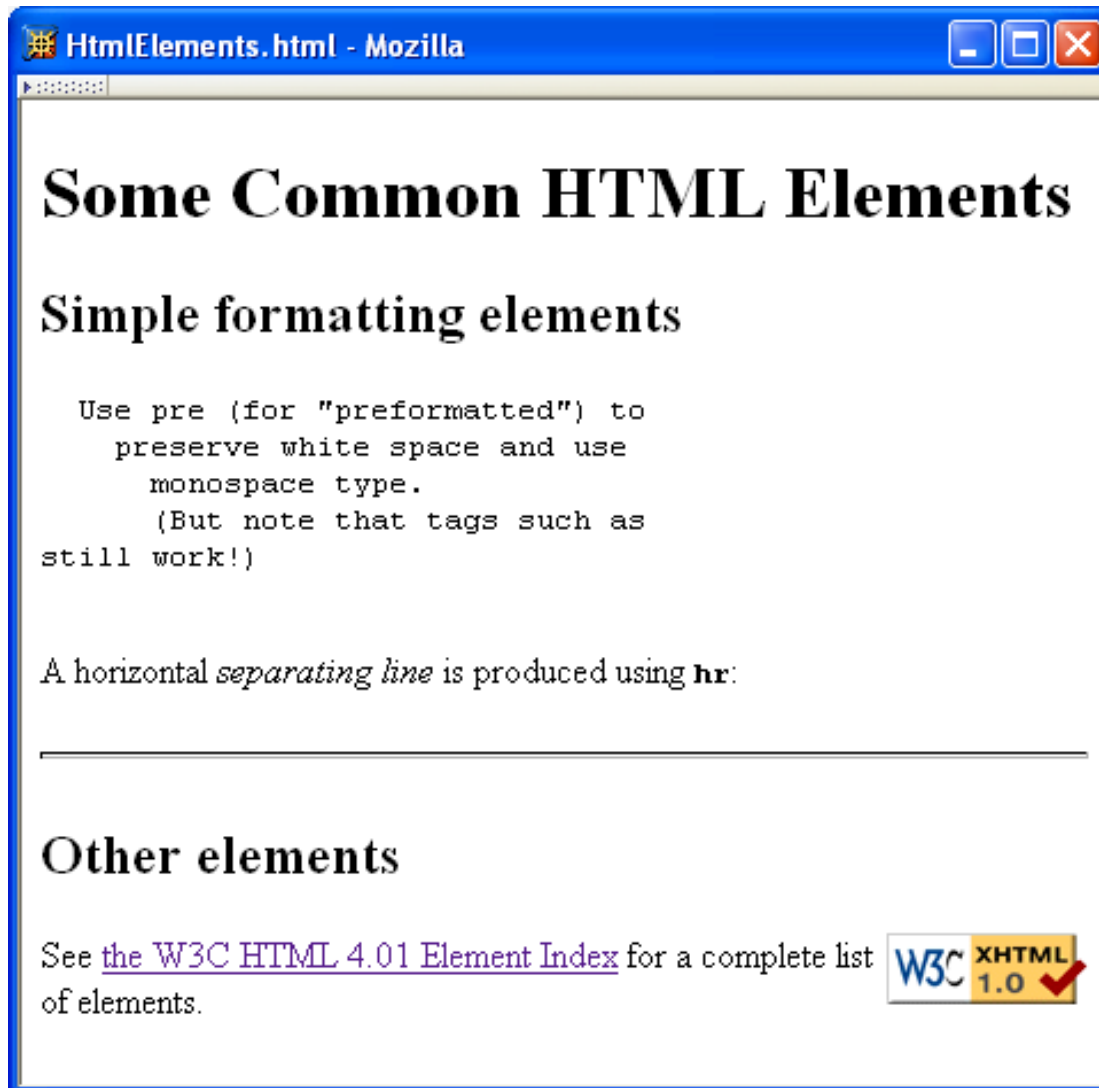
# Common HTML Elements

- ◆ Headings are produced using h1, h2, ..., h6 elements:

```
<h1>  
    Some Common HTML Elements  
</h1>  
<h2>  
    Simple formatting elements  
</h2>
```

- ◆ Should use h1 for highest level, h2 for next highest, etc.
  - Change style (next chapter) if you don't like the "look" of a heading

# Common HTML Elements



HtmlElements.html - Mozilla

## Some Common HTML Elements

### Simple formatting elements


Use `pre` (for "preformatted") to preserve white space and use monospace type.  
(But note that tags such as `pre` still work!)

A horizontal *separating line* is produced using `hr`:

---

### Other elements

See [the W3C HTML 4.01 Element Index](#) for a complete list of elements.



# Common HTML Elements

- ◆ Use *pre* to retain format of text and display using monospace font:

```
<pre>
Use pre (for "preformatted") to
preserve white space and use
monospace type.
(But note that tags such as<br />still work!)
</pre>
```

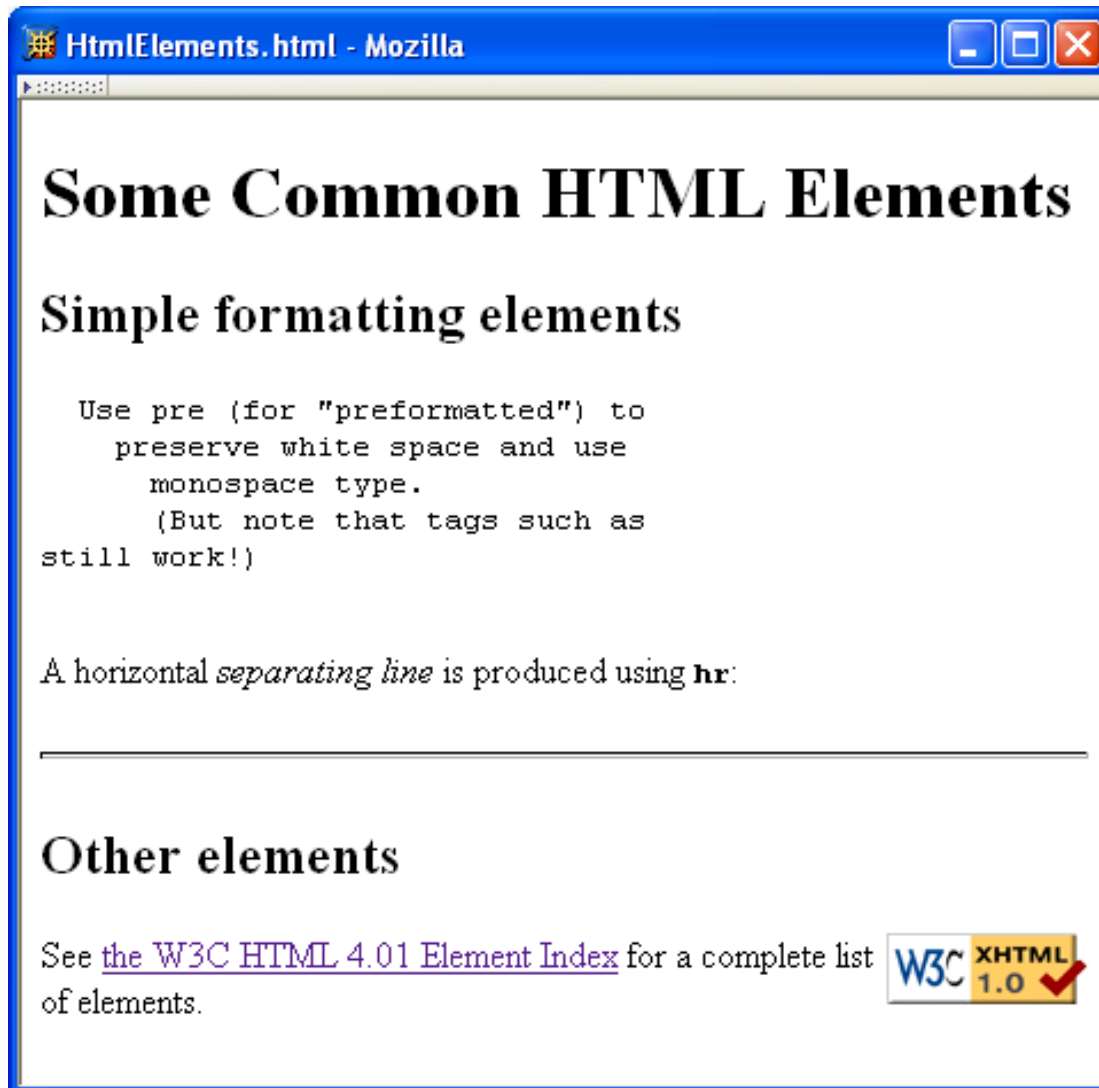
- ◆ Note that any embedded markup (such as *<br />* ) is still treated as markup!



# Common HTML Elements

- ◆ *br* element represents line break
- ◆ *br* is example of an **empty element**, i.e., element that is not allowed to have content
- ◆ XML allows two syntactic representations of empty elements
  - **Empty tag** syntax `<br />` is recommended for browser compatibility
  - XML parsers also recognize syntax `<br></br>` (start tag followed immediately by end tag), but many browsers do not understand this for empty elements

# Common HTML Elements



HtmlElements.html - Mozilla

## Some Common HTML Elements

### Simple formatting elements


Use `pre` (for "preformatted") to preserve white space and use monospace type.  
(But note that tags such as `pre` still work!)

A horizontal *separating line* is produced using `hr`:

---

### Other elements

See [the W3C HTML 4.01 Element Index](#) for a complete list of elements.



# Common HTML Elements

- ◆ Text can be formatted in various ways:
  - Apply style sheet technology (next chapter) to a *span* element (a styleless wrapper):

```
<span style="font-style:italic">separating line</span>
```

- Use a phrase element that specifies semantics of text (not style directly):

```
<strong>hr</strong>
```

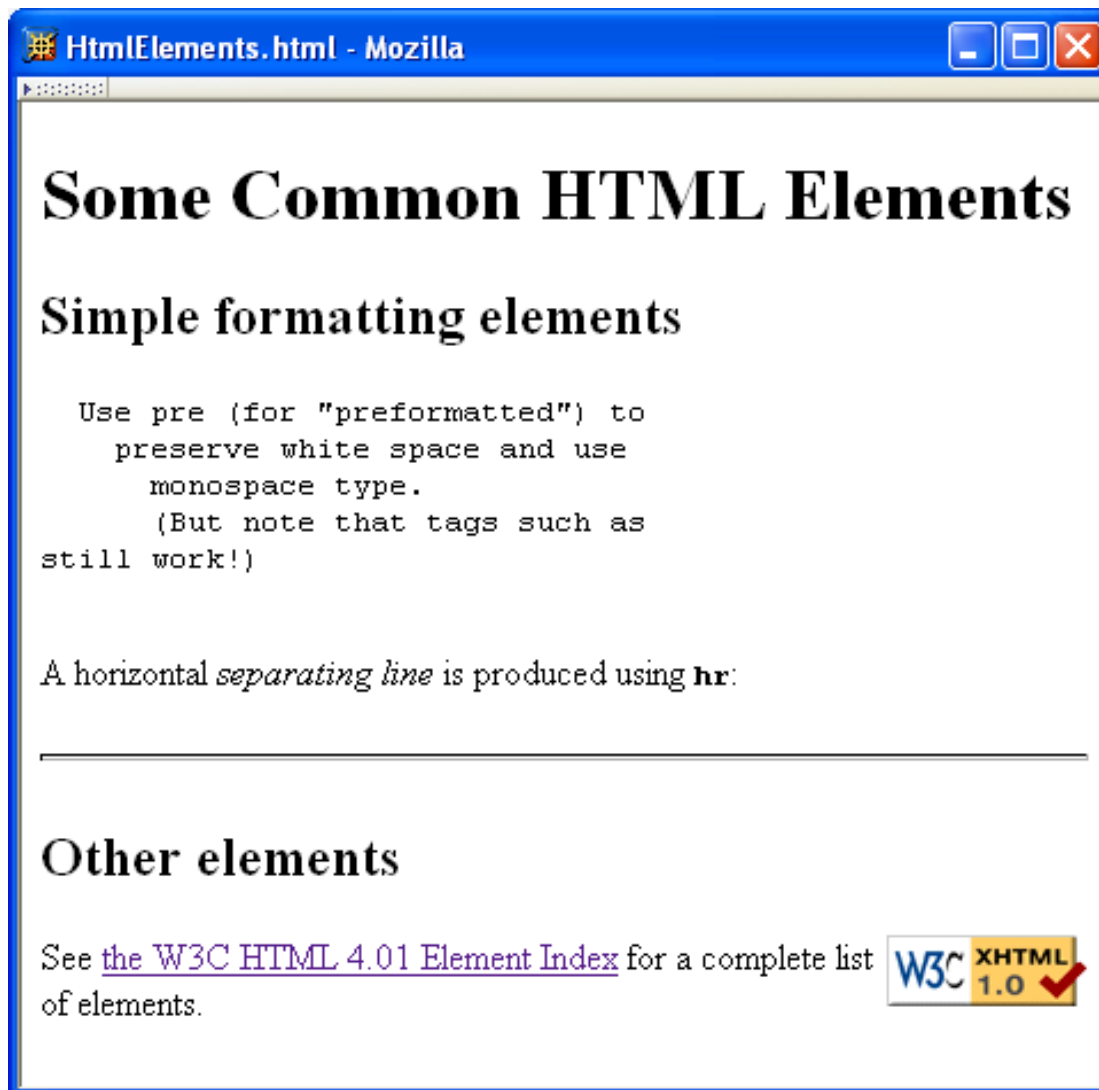
- Use a font style element
  - Not recommended, but frequently used

# Common HTML Elements

TABLE 2.3: HTML font style elements.

Element	Font used by content
<b>b</b>	Bold-face
<i>i</i>	Italic
<code>tt</code>	“Teletype” (fixed-width font)
<big>big</big>	Increased font size
<small>small</small>	Decreased font size

# Common HTML Elements



HtmlElements.html - Mozilla

## Some Common HTML Elements

### Simple formatting elements


Use `pre` (for "preformatted") to preserve white space and use monospace type.  
(But note that tags such as `pre` still work!)

A horizontal *separating line* is produced using `hr`:

---

### Other elements

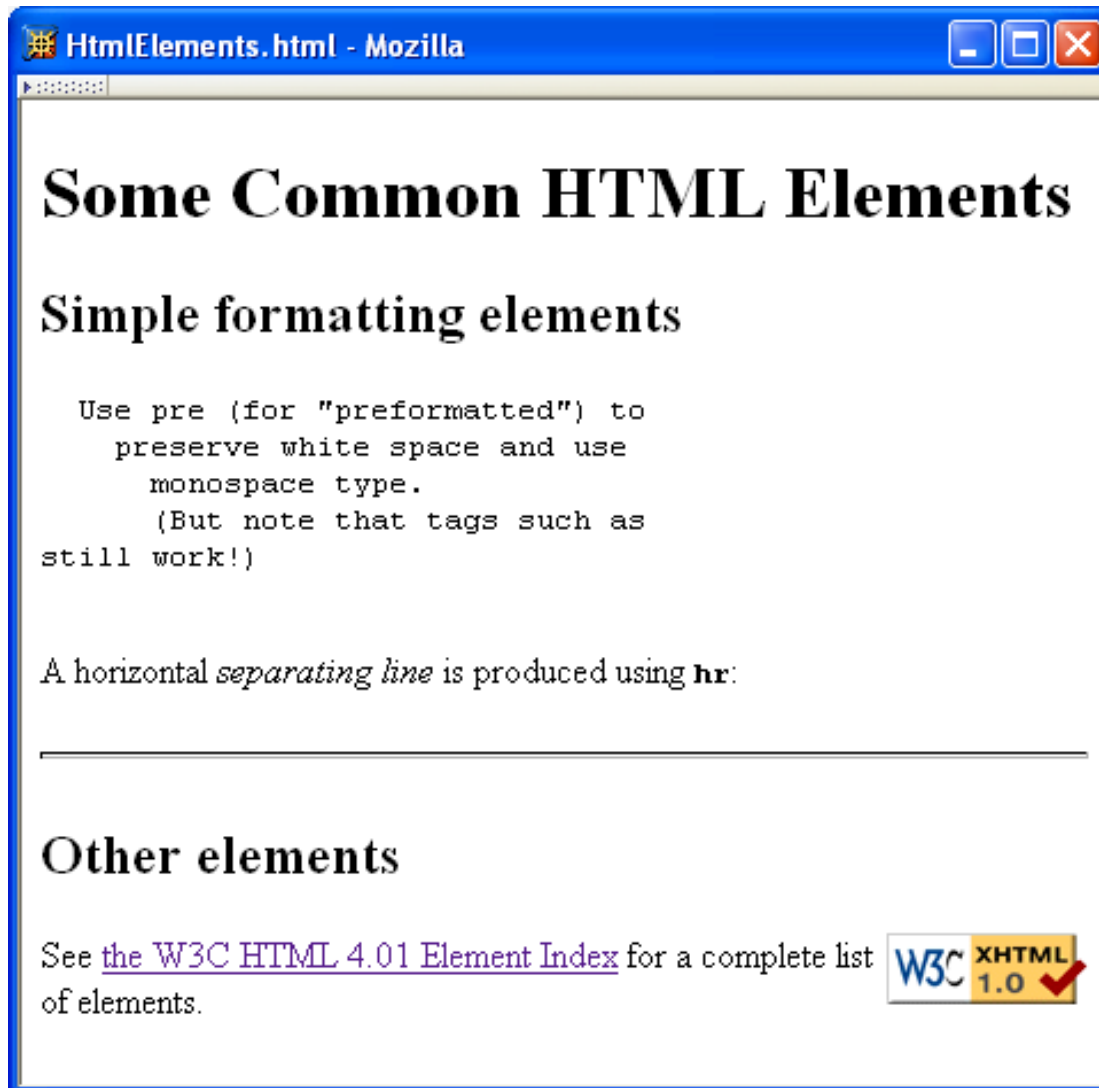
See [the W3C HTML 4.01 Element Index](#) for a complete list of elements.



# Common HTML Elements

- ◆ Horizontal rule is produced using *hr*
- ◆ Also an empty element
- ◆ Style can be modified using style sheet technology

# Common HTML Elements

A screenshot of a Mozilla browser window titled "HtmlElements.html - Mozilla". The browser displays a page with the following content:

**Some Common HTML Elements**

**Simple formatting elements**


Use `pre` (for "preformatted") to preserve white space and use monospace type.  
(But note that tags such as `still work!`)

A horizontal *separating line* is produced using `hr`:

---

**Other elements**

See [the W3C HTML 4.01 Element Index](#) for a complete list of elements.



# Common HTML Elements

- ◆ Images can be embedded using *img* element

```

```

- ◆ Attributes:

- *src*: URL of image file (required). Browser generates a GET request to this URL.
- *alt*: text description of image (required)
- *height* / *width*: dimensions of area that image will occupy (recommended)

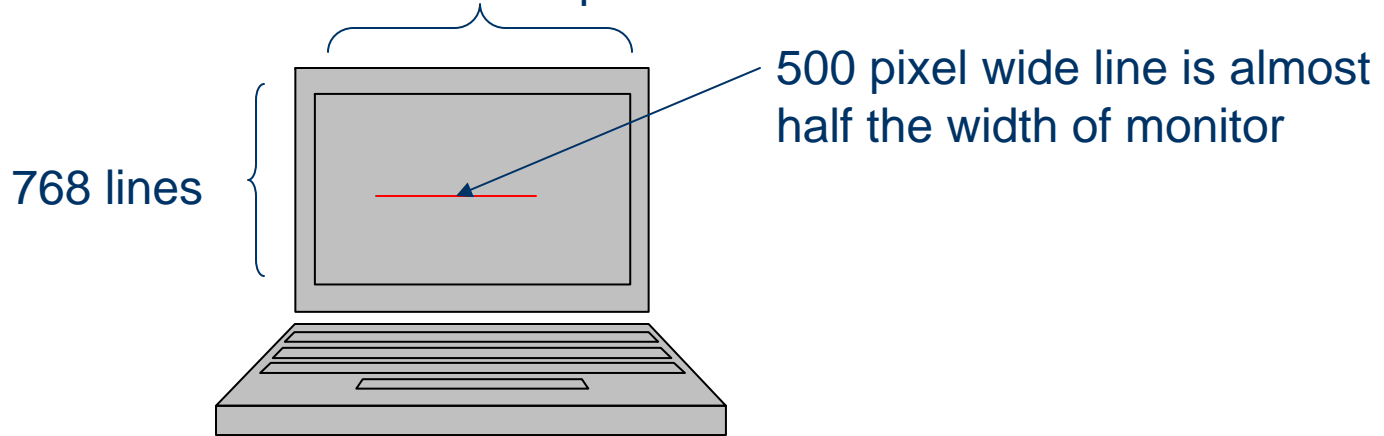


# Common HTML Elements

- ◆ If height and width not specified for image, then browser may need to rearrange the client area after downloading the image (**poor user interface** for Web page)
- ◆ If height and width specified are not the same as the original dimensions of image, browser will resize the image
- ◆ Default units for height and width are “picture elements” (pixels)
  - Can specify percentage of client area using string such as “50%”

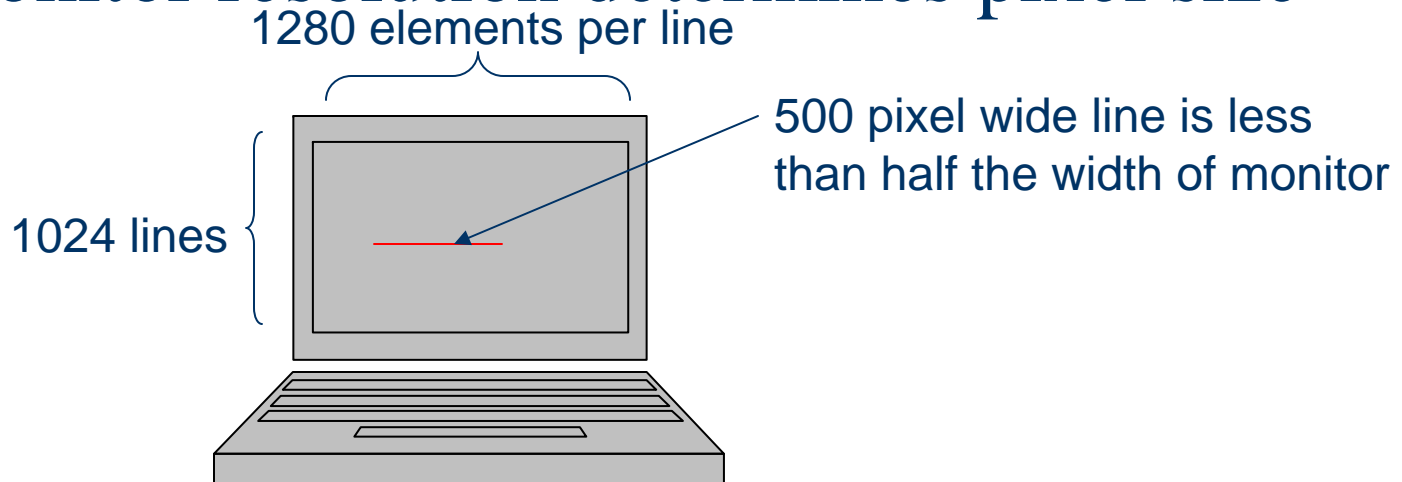
# Common HTML Elements

- ◆ Monitor resolution determines pixel size

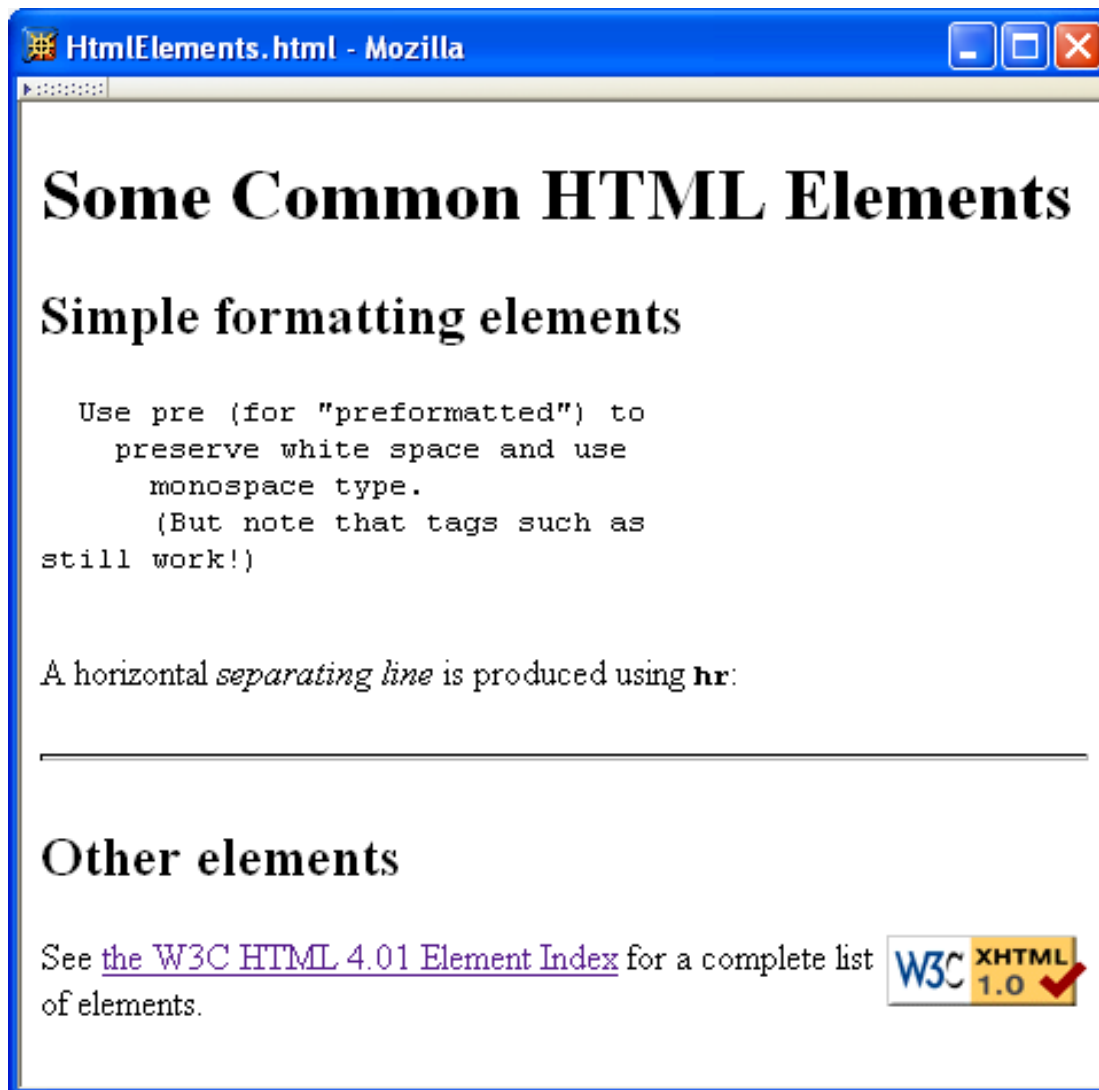


# Common HTML Elements

- ◆ Monitor resolution determines pixel size



# Common HTML Elements



The screenshot shows a Mozilla browser window titled "HtmlElements.html - Mozilla". The page content is as follows:

## Some Common HTML Elements

### Simple formatting elements


Use `pre` (for "preformatted") to preserve white space and use monospace type.  
(But note that tags such as `hr` still work!)

A horizontal *separating line* is produced using `hr`:

---

### Other elements

See [the W3C HTML 4.01 Element Index](#) for a complete list of elements.



# Common HTML Elements

- ◆ Hyperlinks are produced by the anchor element *a*

See

```
<a href="http://www.w3.org/TR/html4/index/elements.html">the  
  W3C HTML 4.01 Element Index</a>  
for a complete list of elements.
```

- ◆ Clicking on a hyperlink causes browser to issue GET request to URL specified in *href* attribute and render response in client area
- ◆ Content of anchor element is text of hyperlink (avoid leading/trailing space in content)

# Common HTML Elements

- ◆ Anchors can be used as source (previous example) or destination

```
<a id="section1" name="section1"></a>
```

- ◆ The fragment portion of a URL is used to reference a destination anchor

```
<a href="http://www.example.org/PageWithAnchor.html#section1">...
```

- ◆ Browser scrolls so destination anchor is at (or near) top of client area

# Common HTML Elements

`<!-- Notice that img must nest within a "block" element,  
such as p -->`

## ◆ Not allowed to use -- within comment

✗ `<!-- This is NOT  
-- a good comment.  
-->`

✗ `<!-- Can't end with more than two dashes! --->`

# Nesting Elements

- ◆ If one element is nested within another element, then the content of the inner element is also content of the outer element

```
<tt><strong>hr</strong></tt>
```

- ◆ XHTML requires that elements be properly nested

```
✗ <tt><strong>hr</tt></strong>
```



# Nesting Elements

- ◆ Most HTML elements are either block or inline
  - Block: browser automatically generates line breaks before and after the element content
    - Ex: *p*, *div*
  - Inline: element content is added to the “flow”
    - Ex: *span*, *tt*, *strong*, *a*

# Nesting Elements

- ◆ Syntactic rules of thumb:
  - Children of *body* must be blocks
  - Blocks can contain inline elements
  - Inline elements *cannot* contain blocks
- ◆ Specific rules for each version of (X)HTML are defined using SGML or XML (covered later)

# Relative URL's

- ◆ Consider an `<img>` start tag containing attribute specification

```
src="valid-xhtml10.png"
```

- ◆ This is an example of a relative URL: it is interpreted relative to the URL of the document that contains the `img` tag

- If document URL is

<http://localhost:8080/MultiFile.html>

then relative URL above represents absolute URL

<http://localhost:8080/valid-xhtml10.png>

# Relative URL's

TABLE 2.4: Absolute URL's corresponding to relative URL's when the base URL is `http://www.example.org/a/b/c.html`.

Relative URL	Absolute URL
<code>d/e.html</code>	<code>http://www.example.org/a/b/d/e.html</code>
<code>../f.html</code>	<code>http://www.example.org/a/f.html</code>
<code>../../g.html</code>	<code>http://www.example.org/g.html</code>
<code>../h/i.html</code>	<code>http://www.example.org/a/h/i.html</code>
<code>/j.html</code>	<code>http://www.example.org/j.html</code>
<code>/k/l.html</code>	<code>http://www.example.org/k/l.html</code>

# Relative URL's

◆ Query and fragment portions of a relative URL are appended to the resulting absolute URL

- Example: If document URL is

<http://localhost:8080/PageAnch.html>

and it contains the anchor element

```
<a href="#section1">...
```

then the corresponding absolute URL is

<http://localhost:8080/PageAnch.html#section1>

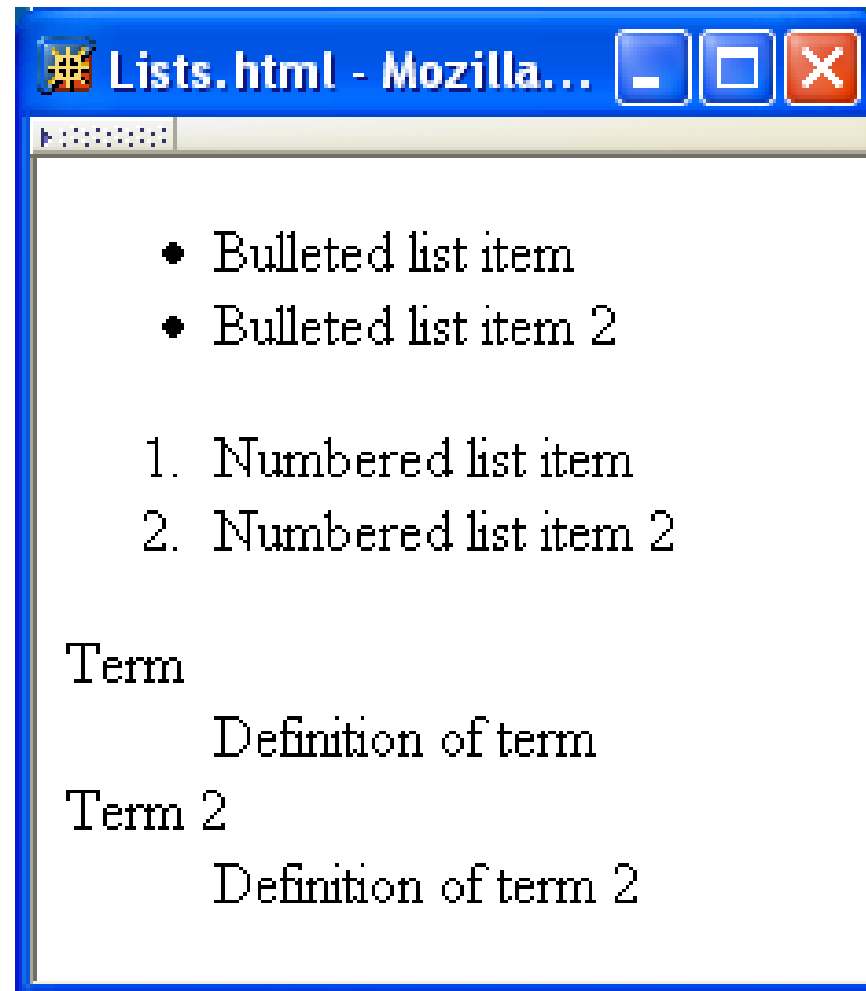
# Relative URL's

## ◆ Advantages:

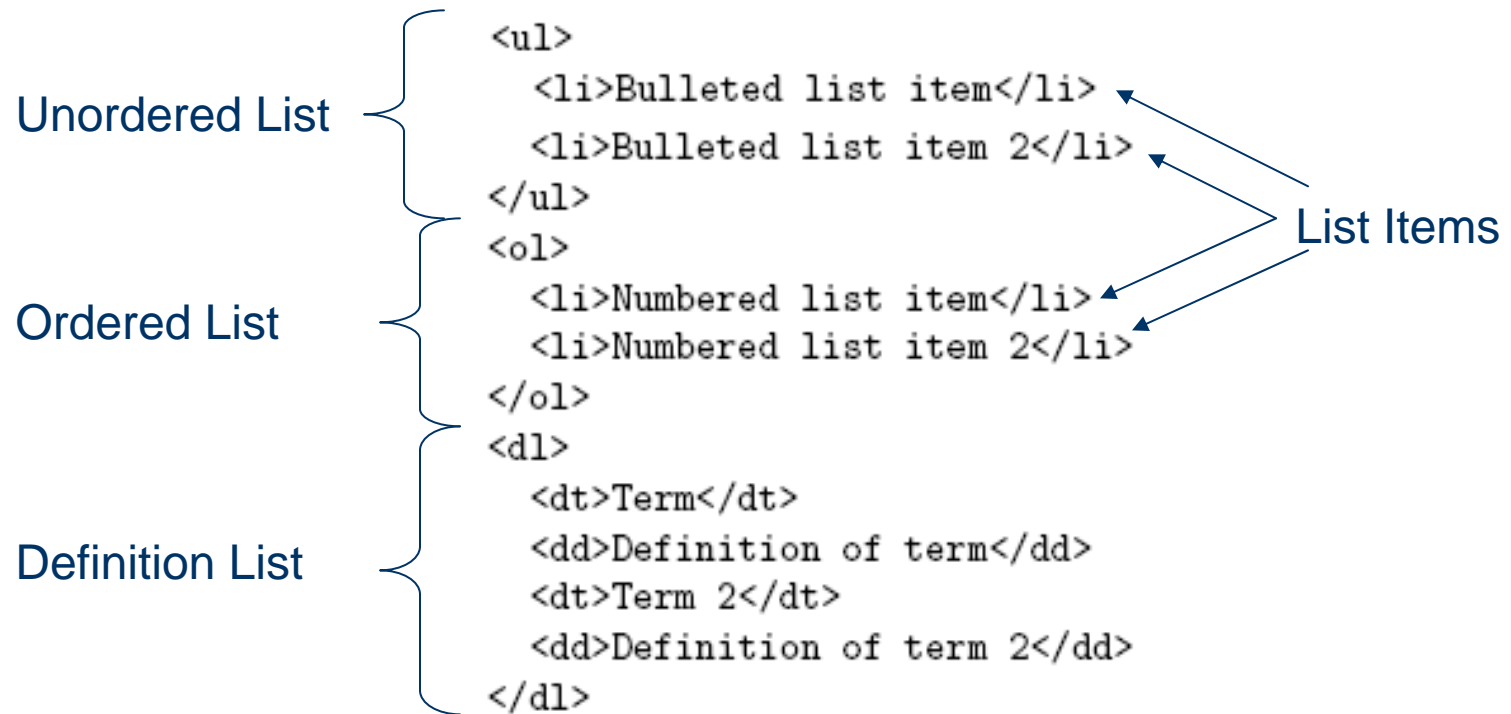
- Shorter than absolute URL's
- Primary: can change the URL of a document (e.g., move document to a different directory or rename the server host) without needing to change URL's within the document

◆ Should use relative URL's whenever possible

# Lists



# Lists





# Lists



```
<ul>
  <li>Bulleterd list item
    <ul>
      <li>Nesterd list item</li>
      <li>Nesterd list item 2</li>
    </ul>
  </li>
  <li>Bulleterd list item 2</li>
</ul>
```

# Tables

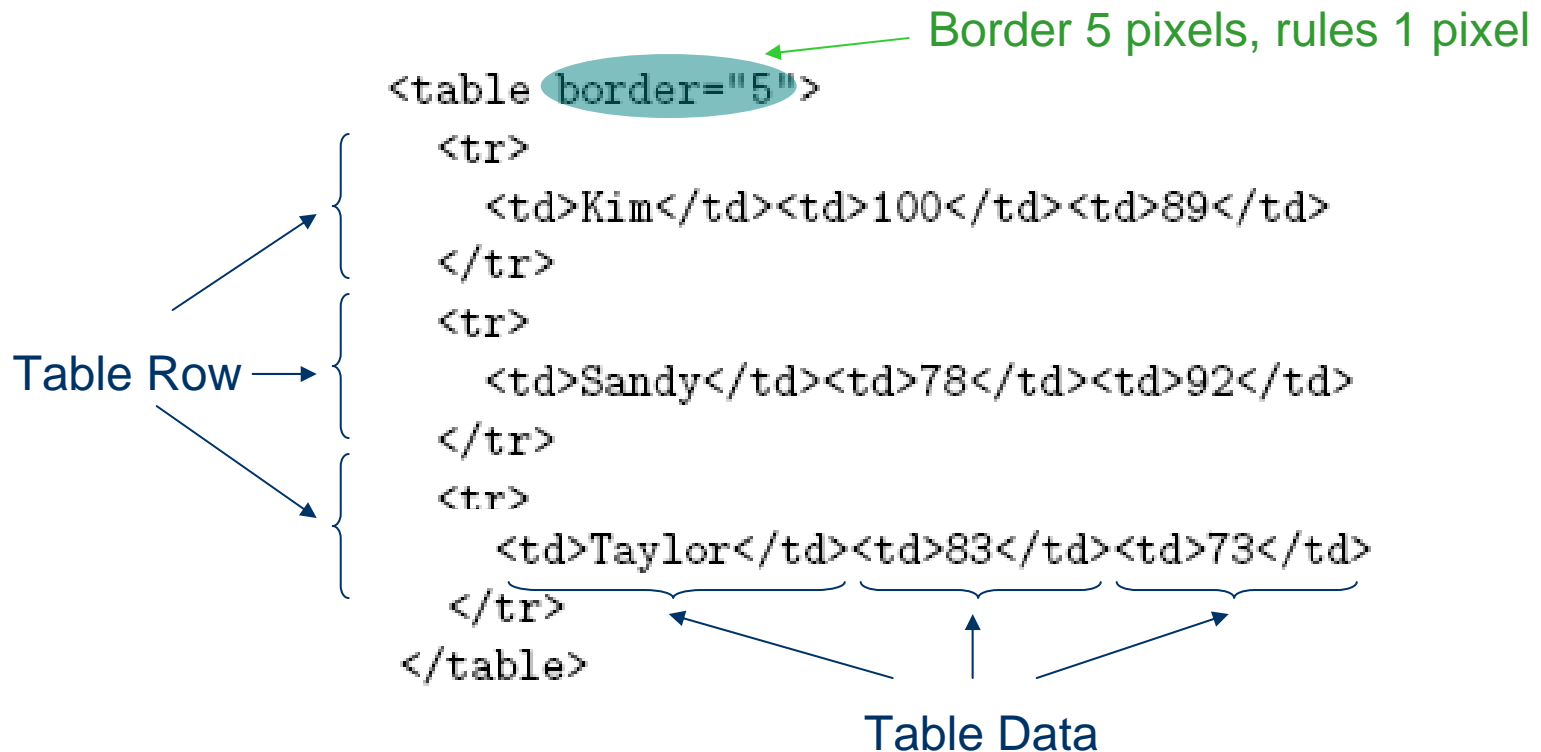
The image shows a web browser window titled "GradeTable.html...". Inside the window is a table with the following data:

Kim	100	89
Sandy	78	92
Taylor	83	73

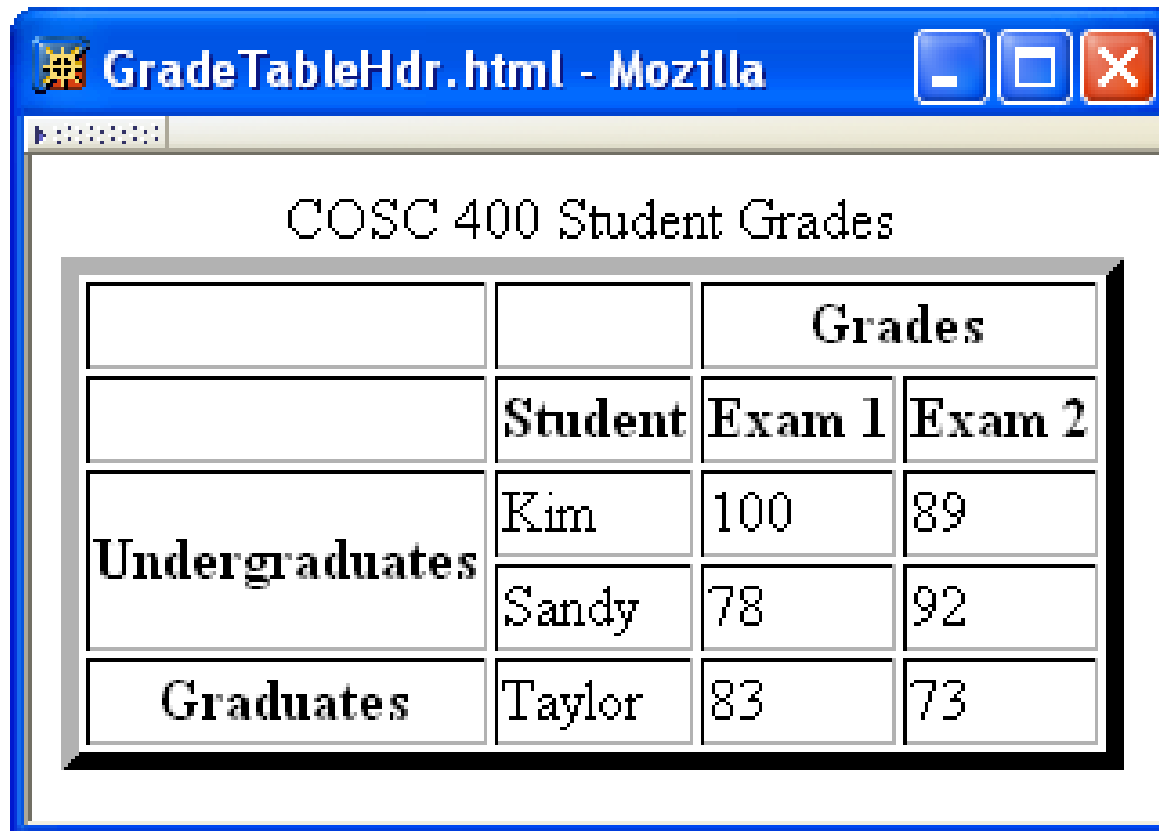
Annotations with arrows:

- "Rules" (top left) points to the top and left borders of the first row.
- "Rules" (bottom center) points to the bottom and right borders of the last row.
- "Borders" (bottom left) points to the bottom and left borders of the entire table.

# Tables



# Tables



The screenshot shows a Mozilla browser window titled "GradeTableHdr.html - Mozilla". The browser's address bar is empty. The main content area displays the text "COSC 400 Student Grades" centered above a table. The table has a header row with "Grades" in the rightmost column. Below the header, there are three rows of data. The first row has "Student", "Exam 1", and "Exam 2" as column headers. The second row shows "Undergraduates" in the first column, "Kim" in the second, "100" in the third, and "89" in the fourth. The third row shows "Sandy" in the second column, "78" in the third, and "92" in the fourth. The fourth row shows "Graduates" in the first column, "Taylor" in the second, "83" in the third, and "73" in the fourth.

		Grades	
	Student	Exam 1	Exam 2
Undergraduates	Kim	100	89
	Sandy	78	92
Graduates	Taylor	83	73

# Tables

```
<table border="5">
  <caption>
    COSC 400 Student Grades
  </caption>
  <tr>
    <td>&nbsp;</td><td>&nbsp;</td><th colspan="2">Grades</th>
  </tr>
  <tr>
    <td>&nbsp;</td><th>Student</th><th>Exam 1</th><th>Exam 2</th>
  </tr>
  <tr>
    <th rowspan="2">Undergraduates</th><td>Kim</td><td>100</td><td>89</td>
  </tr>
  <tr>
    <td>Sandy</td><td>78</td><td>92</td>
  </tr>
  <tr>
    <th>Graduates</th><td>Taylor</td><td>83</td><td>73</td>
  </tr>
</table>
```

Table Header

# Tables



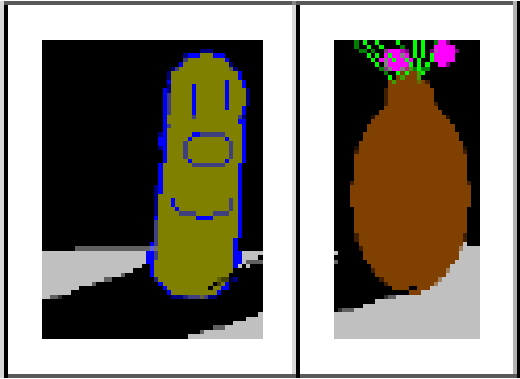
The screenshot shows a Mozilla browser window titled "TableSpacePad.html - Mozilla". The browser displays a table with three columns and two rows. The first row contains the headers "cellspacing", "cellpadding", and "Example". The second row contains the values "10", "10", and two side-by-side images of a yellow cylinder and a brown vase. The table is styled with a border of 1, a cellspacing of 10, and a cellpadding of 10.

cellspacing	cellpadding	Example
10	10	 

```
<table border="1" cellspacing="10" cellpadding="10">
```

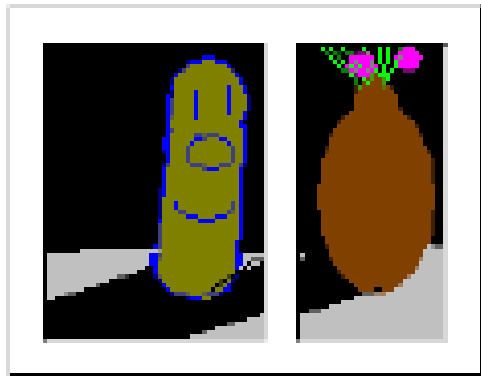
# Tables

cellspacing cellpadding

0	10	
---	----	---

# Tables

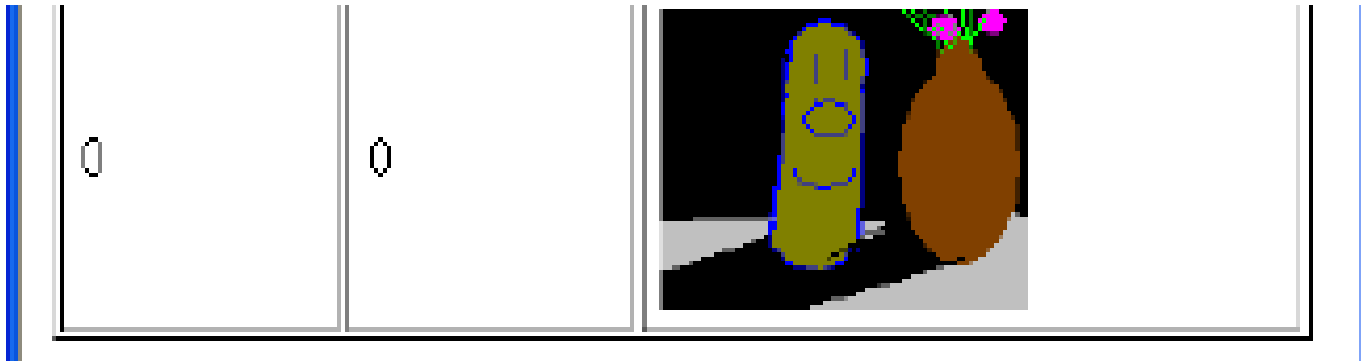
cellspacing cellpadding

10	0	
----	---	---



# Tables

cellspacing cellpadding



# Frames

The screenshot shows a Mozilla browser window titled "Applet (Java 2 Platform SE v1.4.2) - Mozilla". The page content is organized into a frame-like structure. On the left, there is a navigation pane with sections for "All Classes", "Packages" (listing [java.applet](#) and [java.awt](#)), and "Classes" (listing [Applet](#)). The main content area has a top navigation bar with links for "Overview", "Package", "Class" (selected), "Use", "Tree", "Deprecated", "Index", and "Help". Below this, there are links for "PREV CLASS", "NEXT CLASS", "FRAMES", "NO FRAMES", "SUMMARY:", "NESTED", "FIELD", "CONSTR", "METHOD", and "DETAIL: FIELD", "CONSTR", "METHOD". The main content displays the class hierarchy for `java.applet.Applet`, showing it inherits from `java.lang.Object`, `java.awt.Component`, `java.awt.Container`, and `java.awt.Panel`. Below the hierarchy, it lists "All Implemented Interfaces: [Accessible](#), [ImageObserver](#), [MenuContainer](#), [Serializable](#)".

Screen shots are reproduced by permission of Sun Microsystems Inc. All rights reserved.

# Frames

```
<!DOCTYPE html
  PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Java 2 Platform SE v1.4.2</title>
  </head>
  <frameset cols="20%,80%">
    <frameset rows="1*,2*">
      <frame src="overview-frame.html"
        id="upperLeftFrame" name="upperLeftFrame"></frame>
      <frame src="allclasses-frame.html"
        id="lowerLeftFrame" name="lowerLeftFrame"></frame>
    </frameset>
    <frame src="overview-summary.html"
      id="rightFrame" name="rightFrame"></frame>
  </frameset>
</html>
```

1/3,2/3 split

# Frames

- ◆ Hyperlink in one frame can load document in another:

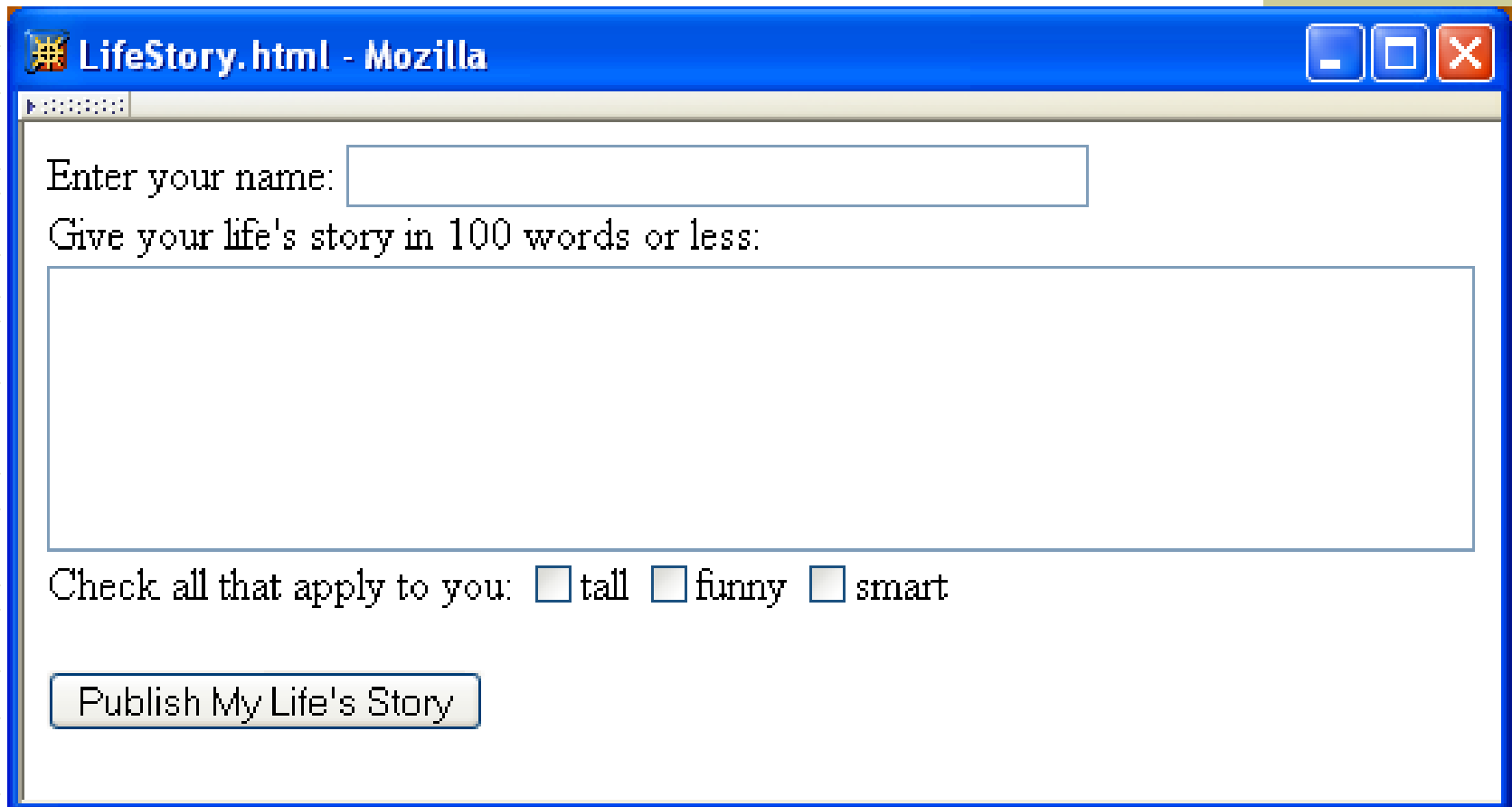
```
<a href="java/applet/package-frame.html" target="lowerLeftFrame">
```

- ◆ Value of target attribute specification is id/name of a frame

# Frames

- ◆ User interface issues:
  - What happens when the page is printed?
  - What happens when the Back button is clicked?
  - How should assistive technology “read” the page?
  - How should the information be displayed on a small display?
- ◆ Recommendation: avoid frames except for applications aimed at “power users”

# Forms



The screenshot shows a Mozilla browser window with the title bar 'LifeStory.html - Mozilla'. The address bar is empty. The main content area contains a form with the following elements:

- A text input field with the label 'Enter your name:'.
- A text area with the label 'Give your life's story in 100 words or less:'.
- A row of three checkboxes with labels 'tall', 'funny', and 'smart', preceded by the text 'Check all that apply to you:'.
- A button labeled 'Publish My Life's Story'.

# Forms

Each form is content of a *form* element

```
<form action="http://www.example.org" method="get">
  <div>
    <label>
      Enter your name: <input type="text" name="username" size="40" />
    </label>
    <br />
    <label>
      Give your life's story in 100 words or less:
      <br />
      <textarea name="lifestory" rows="5" cols="60"></textarea>
    </label>
    <br />
  </div>
</form>
```

# Forms

*action* specifies URL where form data is sent in an HTTP request

```
<form action="http://www.example.org" method="get">
  <div>
    <label>
      Enter your name: <input type="text" name="username" size="40" />
    </label>
    <br />
    <label>
      Give your life's story in 100 words or less:
      <br />
      <textarea name="lifestory" rows="5" cols="60"></textarea>
    </label>
    <br />
  </div>
</form>
```



# Forms

HTTP request method (lower case)

```
<form action="http://www.example.org" method="get">
  <div>
    <label>
      Enter your name: <input type="text" name="username" size="40" />
    </label>
    <br />
    <label>
      Give your life's story in 100 words or less:
      <br />
      <textarea name="lifestory" rows="5" cols="60"></textarea>
    </label>
    <br />
  </div>
</form>
```

# Forms

- ◆ The XHTML grammar requires any child of the *form* element to be a block
- ◆ Many form elements are actually *inline*, so including a block element on top such as a *div* or a *table* is a simple way to be compliant with the grammar

# Forms

```
<form action="http://www.example.org" method="get">
```

`<div>` *div* is the block element analog of *span* (no-style block element)

```
<label>
```

```
  Enter your name: <input type="text" name="username" size="40" />
```

```
</label>
```

```
<br />
```

```
<label>
```

```
  Give your life's story in 100 words or less:
```

```
  <br />
```

```
  <textarea name="lifestory" rows="5" cols="60"></textarea>
```

```
</label>
```

```
<br />
```

# Forms

```
<form action="http://www.example.org" method="get">
```

**<div>** Form control elements must be content of a block element

```
<label>
```

```
  Enter your name: <input type="text" name="username" size="40" />
```

```
</label>
```

```
<br />
```

```
<label>
```

```
  Give your life's story in 100 words or less:
```

```
  <br />
```

```
  <textarea name="lifestory" rows="5" cols="60"></textarea>
```

```
</label>
```

```
<br />
```

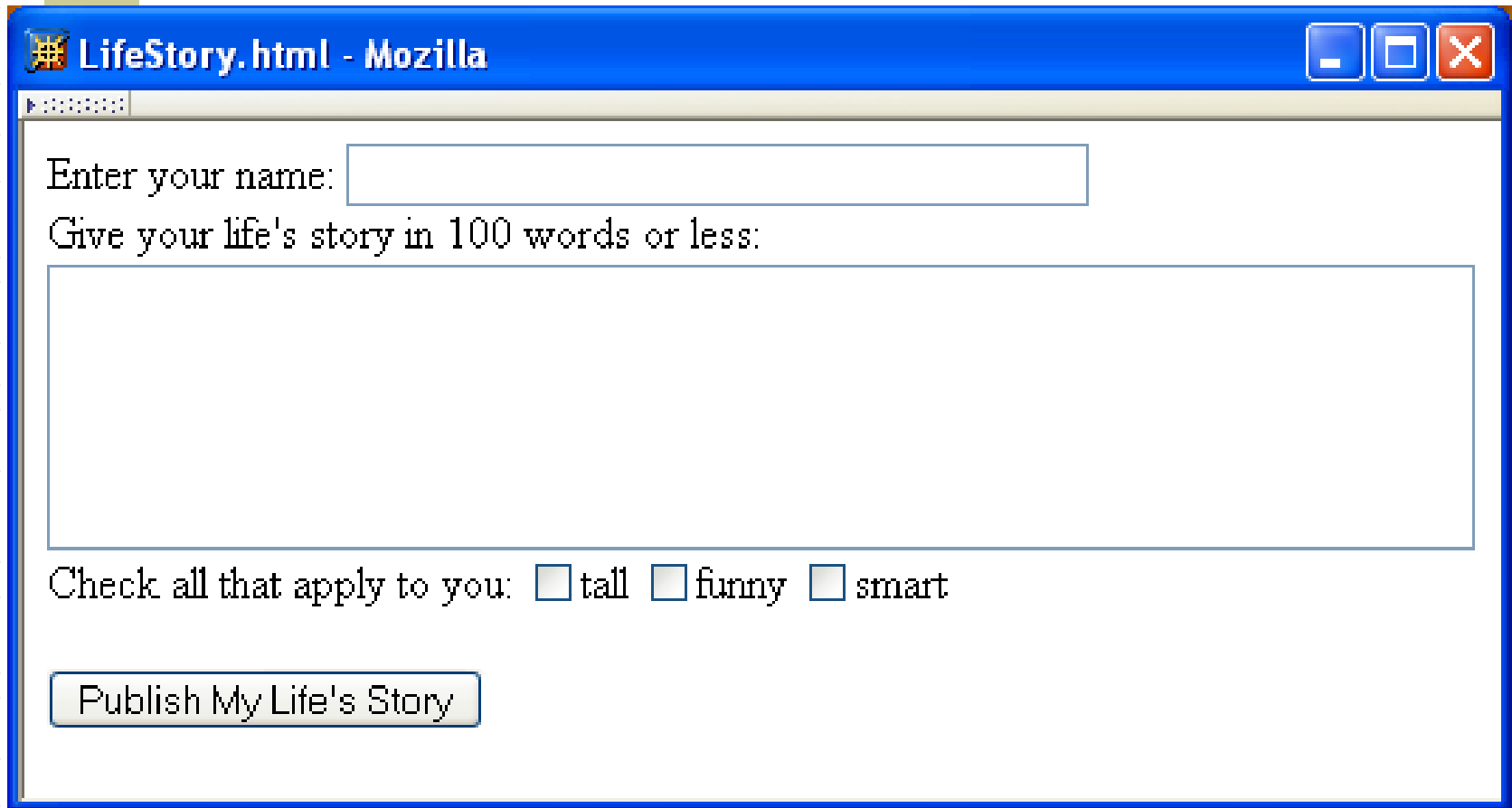
# Forms

```
<form action="http://www.example.org" method="get">
  <div>
    <label>
      Text field control (form user-interface element)
      Enter your name: <input type="text" name="username" size="40" />
    </label>
    <br />
    <label>
      Give your life's story in 100 words or less:
      <br />
      <textarea name="lifestory" rows="5" cols="60"></textarea>
    </label>
    <br />
  </div>
</form>
```

# Forms

```
<form action="http://www.example.org" method="get">
  <div>
    <label>
      Text field used for one-line inputs
      Enter your name: <input type="text" name="username" size="40" />
    </label>
    <br />
    <label>
      Give your life's story in 100 words or less:
      <br />
      <textarea name="lifestory" rows="5" cols="60"></textarea>
    </label>
    <br />
  </div>
</form>
```

# Forms



The screenshot shows a Mozilla browser window with the title bar 'LifeStory.html - Mozilla'. The address bar is empty. The main content area contains a form with the following elements:

- A text input field with the label 'Enter your name:'.
- A text area with the label 'Give your life's story in 100 words or less:'.
- A row of three checkboxes with labels 'tall', 'funny', and 'smart', preceded by the text 'Check all that apply to you:'.
- A button labeled 'Publish My Life's Story'.

# Forms

```
<form action="http://www.example.org" method="get">
  <div>
    <label> Name associated with this control's data in HTTP request
      Enter your name: <input type="text" name="username" size="40" />
    </label>
    <br />
    <label>
      Give your life's story in 100 words or less:
      <br />
      <textarea name="lifestory" rows="5" cols="60"></textarea>
    </label>
    <br />
  </div>
</form>
```



# Forms

```
<form action="http://www.example.org" method="get">
```

```
<div>
```

```
<label>
```

Width (number of characters) of text field

```
Enter your name: <input type="text" name="username" size="40" />
```

```
</label>
```

```
<br />
```

```
<label>
```

```
Give your life's story in 100 words or less:
```

```
<br />
```

```
<textarea name="lifestory" rows="5" cols="60"></textarea>
```

```
</label>
```

```
<br />
```

# Forms

```
<form action="http://www.example.org" method="get">
```

```
<div>
```

```
<label>
```

```
Enter your name: <input type="text" name="username" size="40" />
```

```
</label>
```

```
<br />
```

```
<label>
```

```
Give your life's story in 100 words or less:
```

```
<br />
```

```
<textarea name="lifestory" rows="5" cols="60"></textarea>
```

```
</label>
```

```
<br />
```

*input* is an empty element

# Forms

```
<form action="http://www.example.org" method="get">
  <div>
    <label> Use label to associate text with a control
      Enter your name: <input type="text" name="username" size="40" />
    </label>
    <br />
    <label>
      Give your life's story in 100 words or less:
      <br />
      <textarea name="lifestory" rows="5" cols="60"></textarea>
    </label>
    <br />
  </div>
</form>
```

Only one control inside a label element!

# Forms

```
<form action="http://www.example.org" method="get">
  <div>
    <label>
      Enter your name: <input type="text" name="username" size="40" />
    </label>
    <br /> Form controls are inline elements
    <label>
      Give your life's story in 100 words or less:
      <br />
      <textarea name="lifestory" rows="5" cols="60"></textarea>
    </label>
    <br />
```

# Forms

```
<form action="http://www.example.org" method="get">
  <div>
    <label>
      Enter your name: <input type="text" name="username" size="40" />
    </label>
    <br />
    <label>
      Give your life's story in 100 words or less:
      <br /> textarea control used for multi-line input
      <textarea name="lifestory" rows="5" cols="60"></textarea>
    </label>
    <br />
```

# Forms

```
<form action="http://www.example.org" method="get">
  <div>
    <label>
      Enter your name: <input type="text" name="username" size="40" />
    </label>
    <br />
    <label>
      Give your life's story in 100 words or less:
      <br />
      <textarea name="lifestory" rows="5" cols="60" />
    </label>
    <br />
  </div>
</form>
```

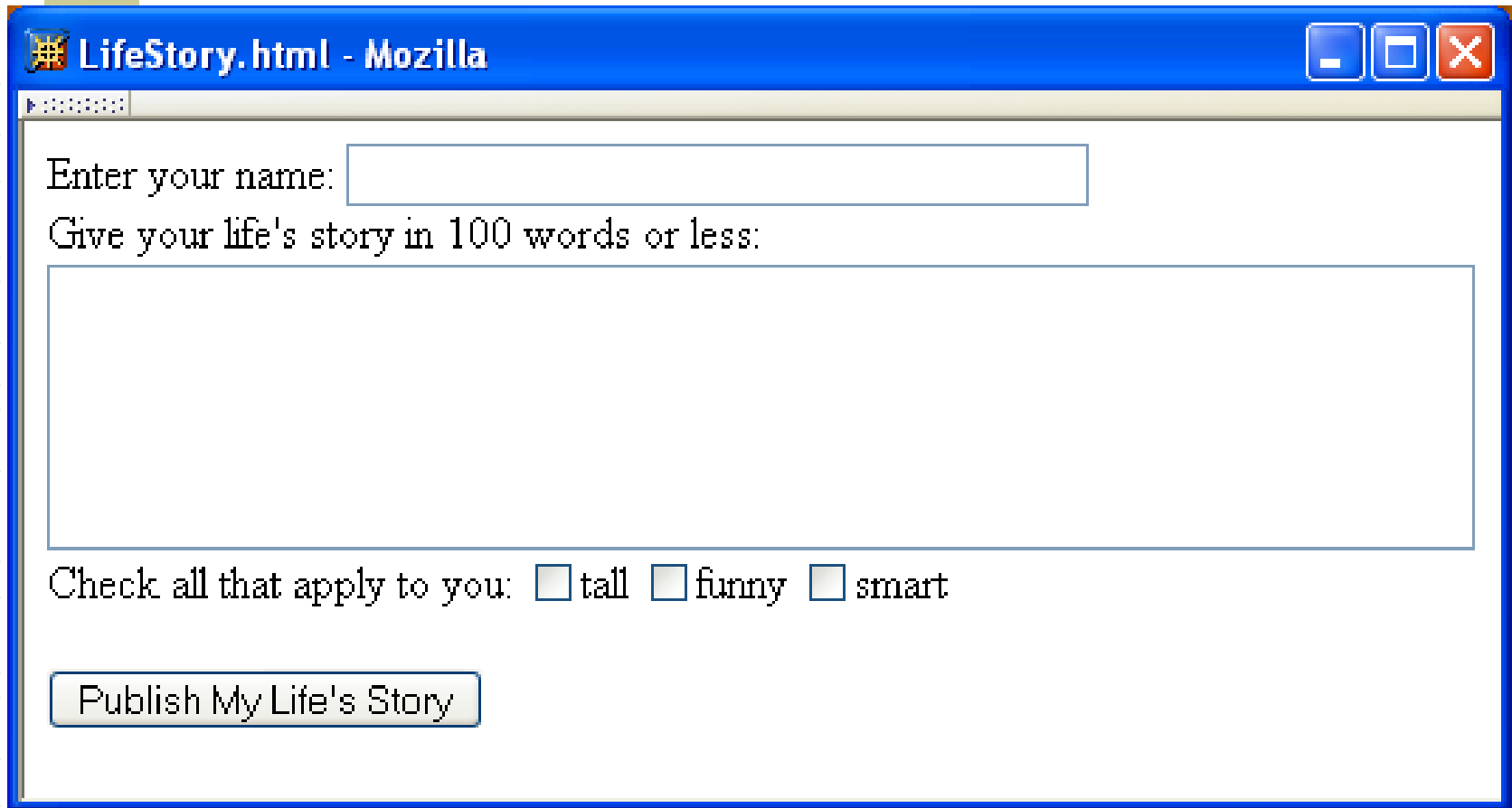
Height and width in characters

# Forms

```
<form action="http://www.example.org" method="get">
  <div>
    <label>
      Enter your name: <input type="text" name="username" size="40" />
    </label>
    <br />
    <label>
      Give your life's story in 100 words or less:
      <br />
      <textarea name="lifestory" rows="5" cols="60"></textarea>
    </label>
    <br />
```

*textarea* is not an empty element; any content is displayed

# Forms



The screenshot shows a Mozilla browser window with the title bar 'LifeStory.html - Mozilla'. The address bar is empty. The main content area contains a form with the following elements:

- A text input field with the label 'Enter your name:'.
- A text area with the label 'Give your life's story in 100 words or less:'.
- A row of three checkboxes with labels 'tall', 'funny', and 'smart', preceded by the text 'Check all that apply to you:'.
- A button labeled 'Publish My Life's Story'.



# Forms

Check all that apply to you:

```
<label>                Checkbox control
  <input type="checkbox" name="boxgroup1" value="tall" />tall
</label>
```

```
<label>
  <input type="checkbox" name="boxgroup1" value="funny" />funny
</label>
```

```
<label>
  <input type="checkbox" name="boxgroup1" value="smart" />smart
</label>
```

```
<br /><br />
```

```
<input type="submit" name="doit" value="Publish My Life's Story" />
```

```
</div>
```

```
</form>
```

# Forms

Check all that apply to you:

Value sent in HTTP request if box is checked

```
<label>
  <input type="checkbox" name="boxgroup1" value="tall" />tall
</label>
<label>
  <input type="checkbox" name="boxgroup1" value="funny" />funny
</label>
<label>
  <input type="checkbox" name="boxgroup1" value="smart" />smart
</label>
<br /><br />
<input type="submit" name="doit" value="Publish My Life's Story" />
</div>
</form>
```

# Forms

Controls can share a common name

Check all that apply to you:

```
<label>
  <input type="checkbox" name="boxgroup1" value="tall" />tall
</label>
<label>
  <input type="checkbox" name="boxgroup1" value="funny" />funny
</label>
<label>
  <input type="checkbox" name="boxgroup1" value="smart" />smart
</label>
<br /><br />
<input type="submit" name="doit" value="Publish My Life's Story" />
</div>
</form>
```

# Forms

Check all that apply to you:

```
<label>  
  <input type="checkbox" name="boxgroup1" value="tall" />tall  
</label>
```

```
<label>  
  <input type="checkbox" name="boxgroup1" value="funny" />funny  
</label>
```

```
<label>  
  <input type="checkbox" name="boxgroup1" value="smart" />smart  
</label>
```

```
<br /><br />
```

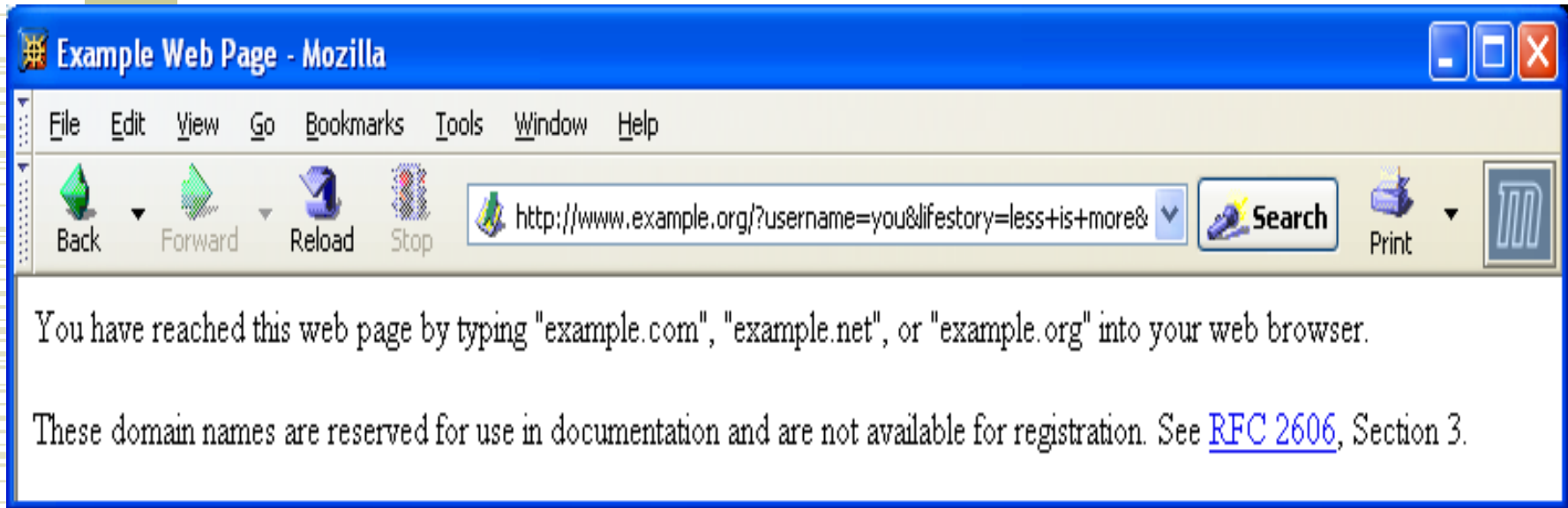
```
<input type="submit" name="doit" value="Publish My Life's Story" />
```

```
</div>
```

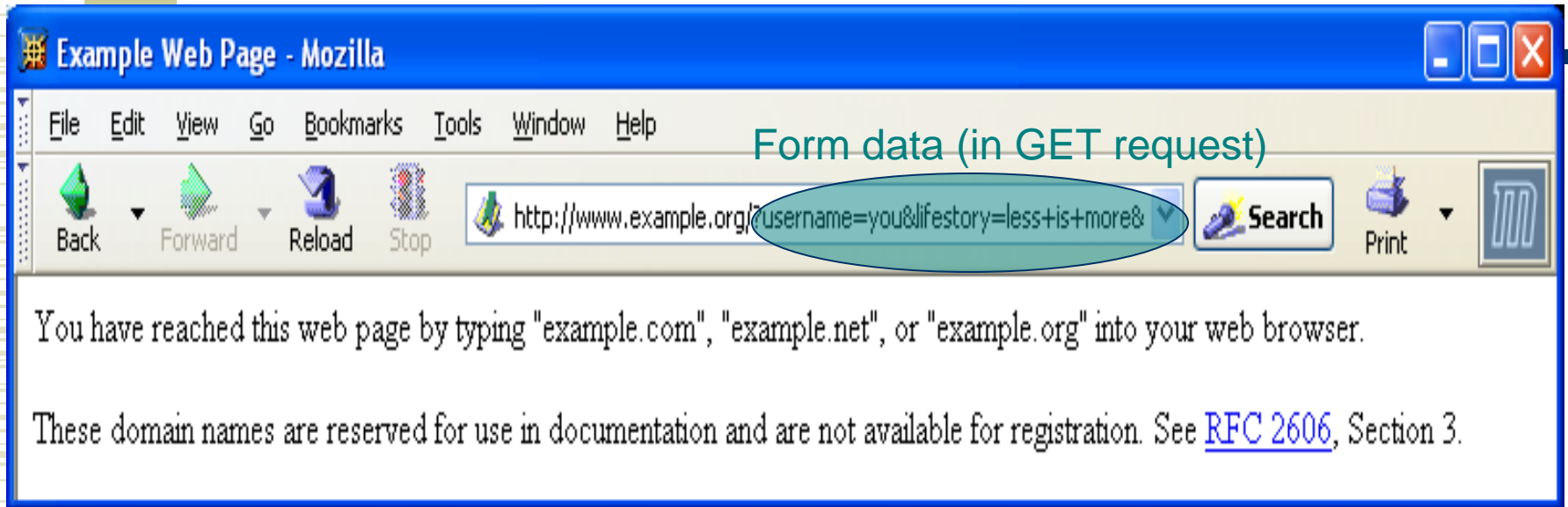
Submit button: form data sent to action URL if button is clicked

```
</form>
```

# Forms



# Forms



# Forms

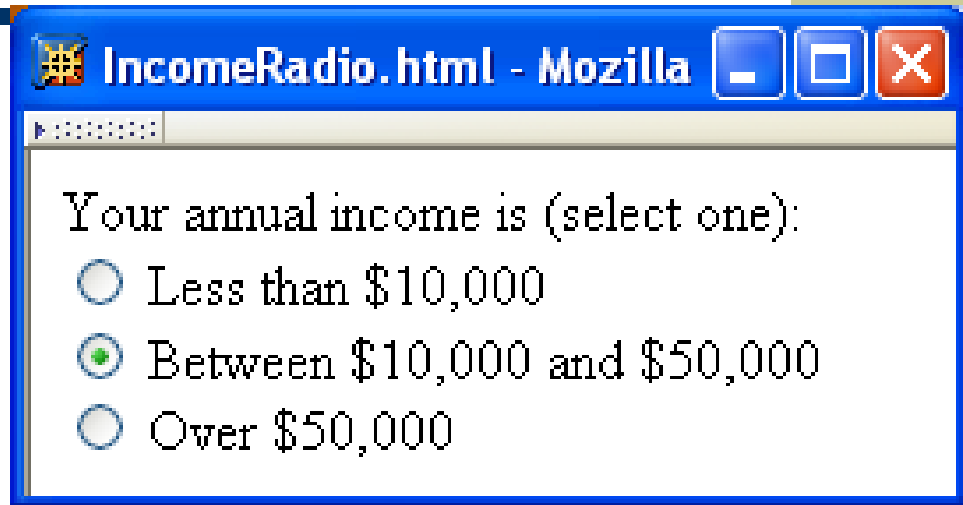
Check all that apply to you:

```
<label>
  <input type="checkbox" name="boxgroup1" value="tall" />tall
</label>
<label>
  <input type="checkbox" name="boxgroup1" value="funny" />funny
</label>
<label>
  <input type="checkbox" name="boxgroup1" value="smart" />smart
</label>
<br /><br />
<input type="submit" name="doit" value="Publish My Life's Story" />
</div>
</form>
```

Displayed on button and sent to server if button clicked

# Forms

Radio buttons: at most one can be selected at a time.



The screenshot shows a Mozilla browser window with the title "IncomeRadio.html - Mozilla". The address bar is empty. The main content area contains the text "Your annual income is (select one):" followed by three radio button options:

- Less than \$10,000
- Between \$10,000 and \$50,000
- Over \$50,000



# Forms

```
Your annual income is (select one):<br />
<label>                                Radio button control
  <input type="radio" name="radgroup1" value="0-10" />
    Less than $10,000
</label><br />
<label>
  <input type="radio" name="radgroup1" value="10-50"
    checked="checked" />
    Between $10,000 and $50,000
</label><br />
<label>
  <input type="radio" name="radgroup1" value=">50" />
    Over $50,000
</label>
```

# Forms

Your annual income is (select one):<br />

```
<label>
```

```
  <input type="radio" name="radgroup1" value="0-10" />
```

```
    Less than $10,000
```

```
</label><br />
```

```
<label>
```

```
  <input type="radio" name="radgroup1" value="10-50"
    checked="checked" />
```

```
    Between $10,000 and $50,000
```

```
</label><br />
```

```
<label>
```

```
  <input type="radio" name="radgroup1" value=">50" />
```

```
    Over $50,000
```

```
</label>
```

All radio buttons with the same name form a *button set*

# Forms

```
Your annual income is (select one):<br />
<label>
  <input type="radio" name="radgroup1" value="0-10" />
  Less than $10,000
</label><br />
<label>
  <input type="radio" name="radgroup1" value="10-50"
    checked="checked" />
  Between $10,000 and $50,000
</label><br />
<label>
  <input type="radio" name="radgroup1" value=">50" />
  Over $50,000
</label>
```

Only one button of a set can be selected at a time

# Forms

Your annual income is (select one):<br />

```
<label>
```

```
  <input type="radio" name="radgroup1" value="0-10" />
```

```
    Less than $10,000
```

```
</label><br />
```

```
<label>
```

```
  <input type="radio" name="radgroup1" value="10-50"
```

```
    checked="checked" /> This button is initially selected
```

```
    Between $10,000 and $50,000 (checked attribute also applies
```

```
</label><br />
```

```
to check boxes)
```

```
<label>
```

```
  <input type="radio" name="radgroup1" value=">50" />
```

```
    Over $50,000
```

```
</label>
```

# Forms

Your annual income is (select one):<br />

```
<label>
```

```
  <input type="radio" name="radgroup1" value="0-10" />
```

```
    Less than $10,000
```

```
</label><br />
```

```
<label>
```

```
  <input type="radio" name="radgroup1" value="10-50"
```

```
    checked="checked" /> Boolean attribute: default false,
```

```
    Between $10,000 and $50,000 set true by specifying name as  
                                value
```

```
</label><br />
```

```
  <input type="radio" name="radgroup1" value=">50" />
```

```
    Over $50,000
```

```
</label>
```

# Forms

Your annual income is (select one):<br />

```
<label>
```

```
  <input type="radio" name="radgroup1" value="0-10" />
```

```
    Less than $10,000
```

```
</label><br />
```

```
<label>
```

```
  <input type="radio" name="radgroup1" value="10-50"
        checked="checked" />
```

```
    Between $10,000 and $50,000
```

```
</label><br />
```

```
<label>
```

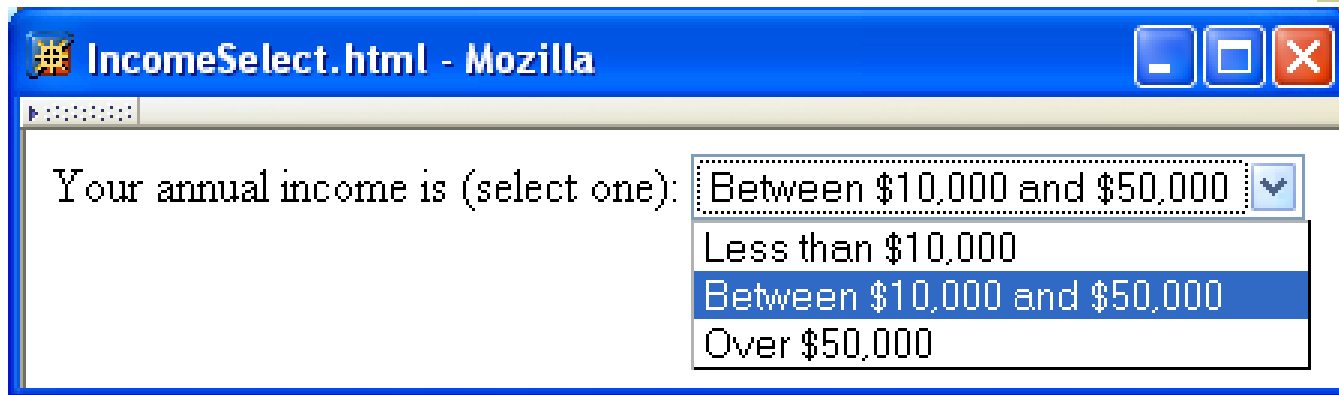
```
  <input type="radio" name="radgroup1" value="&gt;50" />
```

```
    Over $50,000
```

```
</label>
```

Represents string: >50

# Forms



IncomeSelect.html - Mozilla

Your annual income is (select one):

- Between \$10,000 and \$50,000
- Less than \$10,000
- Between \$10,000 and \$50,000
- Over \$50,000

} Menu

# Forms

Your annual income is (select one):

```
<select name="income"> Menu control; name given once
  <option value="0-10">Less than $10,000</option>
  <option value="10-50" selected="selected">
    Between $10,000 and $50,000
  </option>
  <option value=">50">Over $50,000</option>
</select>
```



# Forms

Your annual income is (select one):

```
<select name="income">Each menu item has its own value
  <option value="0-10">Less than $10,000</option>
  <option value="10-50" selected="selected">
    Between $10,000 and $50,000
  </option>
  <option value=">50">Over $50,000</option>
</select>
```

# Forms

Your annual income is (select one):

```
<select name="income">
```

```
  <option value="0-10">Less than $10,000</option>
```

```
  <option value="10-50" selected="selected">
```

```
    Between $10,000 and $50,000
```

Item initially displayed in menu control

```
  <option value=">50">Over $50,000</option>
```

```
</select>
```


# Forms

- ◆ Other form controls:
  - Fieldset (grouping)
  - Password
  - Clickable image
  - Non-submit buttons
  - Hidden (embed data)
  - File upload
  - Hierarchical menus

MoreControls.html - Mozilla

Example of a fieldset


input type=password:

input type=image: 


input type=button:

input type=hidden:

input type=file:

button type=button: 

Hierarchical menu

select with optgroup:  

- Make a selection
- Group1**
- 1.1
- 1.2
- Group2**
- 2.1
- 2.2

# Forms

TABLE 2.5: HTML 4.01/XHTML 1.0 non-deprecated form controls.

Element	type Attribute	Control
input	text	Text input
input	password	Password input
input	checkbox	Checkbox
input	radio	Radio button
input	submit	Submit button
input	image	Graphical submit button
input	reset	Reset button (form clear)
input	button	Push button (for use with scripts)
input	hidden	Non-displayed control (stores server-supplied information)
input	file	File select
button	submit	Submit button with content (not an empty element)
button	reset	Cancel button with content (not an empty element)
button	button	Button with content but no predefined action
select	N/A	Menu
option	N/A	Menu item
optgroup	N/A	Heading in a hierarchical menu
textarea	N/A	Multi-line text input
label	N/A	Associate label with control(s)
fieldset	N/A	Groups controls
legend	N/A	Add caption to a fieldset

# XML DTD

- ◆ Recall that XML is used to define the syntax of XHTML
- ◆ Set of XML files that define a language are known as the document type definition (DTD)
- ◆ DTD primarily consists of declarations:
  - Element type: name and content of elements
  - Attribute list: attributes of an element
  - Entity: define meaning of, *e.g.*, *&gt;*

# XML DTD

## ◆ Example from

<http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd>

```
<!ELEMENT html (head, body)>
```

```
<!ATTLIST html
```

```
  %i18n;
```

```
  id          ID          #IMPLIED
```

```
  xmlns      %URI;      #FIXED 'http://www.w3.org/1999/xhtml' >
```

```
<!ENTITY % i18n
```

```
  "lang      %LanguageCode; #IMPLIED
```

```
  xml:lang  %LanguageCode; #IMPLIED
```

```
  dir       (ltr|rtl)     #IMPLIED" >
```

# XML Element Type Declaration

```
<!ELEMENT html (head, body)>
```

Element type name

# XML Element Type Declaration

```
<!ELEMENT html (head, body)>
```

Element type *content specification* (or *content model*)



# XML Element Type Declaration

```
<!ELEMENT html (head, body)>
```

Element type *content specification* (or *content model*)

TABLE 2.6: Basic XML content specifications.

Specification Type	Syntax	Content Allowed
Empty	EMPTY	None
Arbitrary	ANY	Any content (no restrictions)
Sequence	(elt1, elt2, ...)	Sequence of elements that must appear in order specified
Choice	(elt1   elt2   ...)	Exactly one of the specified elements must appear
Character data	(#PCDATA)	Arbitrary character data, but no elements
Mixed	(#PCDATA   elt1   elt2  ... )*	Any mixture of character data and the specified elements in any order

# XML Element Type Declaration

```
<!ELEMENT html (head, body)>
```

Element type *content specification* (or *content model*)

TABLE 2.6: Basic XML content specifications.

Specification Type	Syntax	Content Allowed
Empty	EMPTY	None
Arbitrary	ANY	Any content (no restrictions)
Sequence	(elt1, elt2, ...)	Sequence of elements that must appear in order specified
Choice	(elt1   elt2   ...)	Exactly one of the specified elements must appear
Character data	(#PCDATA)	Arbitrary character data, but no elements
Mixed	(#PCDATA   elt1   elt2   ... )*	Any mixture of character data and the specified elements in any order

# XML Element Type Declaration

<!ELEMENT br EMPTY>

Element type *content specification* (or *content model*)

TABLE 2.6: Basic XML content specifications.

Specification Type	Syntax	Content Allowed
Empty	EMPTY	None
Arbitrary	ANY	Any content (no restrictions)
Sequence	(elt1, elt2, ...)	Sequence of elements that must appear in order specified
Choice	(elt1   elt2   ...)	Exactly one of the specified elements must appear
Character data	(#PCDATA)	Arbitrary character data, but no elements
Mixed	(#PCDATA   elt1   elt2  ... )*	Any mixture of character data and the specified elements in any order

# XML Element Type Declaration

`<!ELEMENT br EMPTY>`

Element type *content specification* (or *content model*)

TABLE 2.6: Basic XML content specifications.

Specification Type	Syntax	Content Allowed
Empty	EMPTY	None
Arbitrary	ANY	Any content (no restrictions)
Sequence	(elt1, elt2, ...)	Sequence of elements that must appear in order specified
Choice	(elt1   elt2   ...)	Exactly one of the specified elements must appear
Character data	(#PCDATA)	Arbitrary character data, but no elements
Mixed	(#PCDATA   elt1   elt2   ... )*	Any mixture of character data and the specified elements in any order

# XML Element Type Declaration

```
<!ELEMENT select (optgroup|option)+>
```

Element type *content specification (or content model)*

TABLE 2.6: Basic XML content specifications.

Specification Type	Syntax	Content Allowed
Empty	EMPTY	None
Arbitrary	ANY	Any content (no restrictions)
Sequence	(elt1, elt2, ...)	Sequence of elements that must appear in order specified
Choice	(elt1   elt2   ...)	Exactly one of the specified elements must appear
Character data	(#PCDATA)	Arbitrary character data, but no elements
Mixed	(#PCDATA   elt1   elt2  ... )*	Any mixture of character data and the specified elements in any order

# XML Element Type Declaration

```
<!ELEMENT select (optgroup|option)+>
```

Element type *content specification (or content model)*

TABLE 2.6: Basic XML content specifications.

Specification Type	Syntax	Content Allowed
Empty	EMPTY	None
Arbitrary	ANY	Any content (no restrictions)
Sequence	(elt1, elt2, ...)	Sequence of elements that must appear in order specified
Choice	(elt1   elt2   ...)	Exactly one of the specified elements must appear
Character data	(#PCDATA)	Arbitrary character data, but no elements
Mixed	(#PCDATA   elt1   elt2  ... )*	Any mixture of character data and the specified elements in any order

# XML Element Type Declaration

**<!ELEMENT textarea (#PCDATA)>**

Element type *content specification* (or *content model*)

TABLE 2.6: Basic XML content specifications.

Specification Type	Syntax	Content Allowed
Empty	EMPTY	None
Arbitrary	ANY	Any content (no restrictions)
Sequence	(elt1, elt2, ...)	Sequence of elements that must appear in order specified
Choice	(elt1   elt2   ...)	Exactly one of the specified elements must appear
Character data	(#PCDATA)	Arbitrary character data, but no elements
Mixed	(#PCDATA   elt1   elt2   ... )*	Any mixture of character data and the specified elements in any order

# XML Element Type Declaration

**<!ELEMENT textarea (#PCDATA)>**

Element type *content specification* (or *content model*)

TABLE 2.6: Basic XML content specifications.

Specification Type	Syntax	Content Allowed
Empty	EMPTY	None
Arbitrary	ANY	Any content (no restrictions)
Sequence	(elt1, elt2, ...)	Sequence of elements that must appear in order specified
Choice	(elt1   elt2   ...)	Exactly one of the specified elements must appear
Character data	(#PCDATA)	Arbitrary character data, but no elements
Mixed	(#PCDATA   elt1   elt2  ... )*	Any mixture of character data and the specified elements in any order



# XML Element Type Declaration

```
<!ELEMENT select (optgroup|option)+>
```

Element type *content specification* (or *content model*)

# XML Element Type Declaration

```
<!ELEMENT select (optgroup|option)+>
```

Element type *content specification* (or *content model*)

TABLE 2.7: XML content specification iterator characters.

Character	Meaning
?	Sequence/choice is optional (appears zero or one times)
*	Sequence/choice may be repeated an arbitrary number of times, including none
+	Sequence/choice may appear one or more times

# XML Element Type Declaration

```
<!ELEMENT select (optgroup|option)+>
```

Element type *content specification* (or *content model*)

TABLE 2.7: XML content specification iterator characters.

Character	Meaning
?	Sequence/choice is optional (appears zero or one times)
*	Sequence/choice may be repeated an arbitrary number of times, including none
+	Sequence/choice may appear one or more times

# XML Element Type

```
<!ELEMENT table  
  (caption?, (col*|colgroup*), thead?, tfoot?, (tbody+|tr+))>
```

- ◆ Child elements of *table* are:

# XML Element Type

```
<!ELEMENT table  
  (caption?, (col*|colgroup*), thead?, tfoot?, (tbody+|tr+))>
```

- ◆ Child elements of *table* are:
  - Optional *caption*

# XML Element Type

```
<!ELEMENT table  
  (caption?, (col*|colgroup*), thead?, tfoot?, (tbody+|tr+))>
```

- ◆ Child elements of *table* are:
  - Optional *caption* followed by

# XML Element Type

```
<!ELEMENT table  
  (caption?, (col*|colgroup*), thead?, tfoot?, (tbody+|tr+))>
```

- ◆ Child elements of *table* are:
  - Optional *caption* followed by
  - Any number of *col* elements

# XML Element Type

```
<!ELEMENT table  
  (caption?, (col*|colgroup*), thead?, tfoot?, (tbody+|tr+))>
```

- ◆ Child elements of *table* are:
  - Optional *caption* followed by
  - Any number of *col* elements or



# XML Element Type

```
<!ELEMENT table  
  (caption?, (col*|colgroup*), thead?, tfoot?, (tbody+|tr+))>
```

- ◆ Child elements of *table* are:
  - Optional *caption* followed by
  - Any number of *col* elements or any number of *colgroup* elements

# XML Element Type

```
<!ELEMENT table  
  (caption?, (col*|colgroup*), thead?, tfoot?, (tbody+|tr+))>
```

- ◆ Child elements of *table* are:
  - Optional *caption* followed by
  - Any number of *col* elements or any number of *colgroup* elements then

# XML Element Type

```
<!ELEMENT table  
  (caption?, (col*|colgroup*), thead?, tfoot?, (tbody+|tr+))>
```

- ◆ Child elements of *table* are:
  - Optional *caption* followed by
  - Any number of *col* elements or any number of *colgroup* elements then
  - Optional header

# XML Element Type

```
<!ELEMENT table  
    (caption?, (col*|colgroup*), thead?, tfoot?, (tbody+|tr+))>
```

- ◆ Child elements of *table* are:
  - Optional *caption* followed by
  - Any number of *col* elements or any number of *colgroup* elements then
  - Optional header followed by

# XML Element Type

```
<!ELEMENT table  
  (caption?, (col*|colgroup*), thead?, tfoot?, (tbody+|tr+))>
```

- ◆ Child elements of *table* are:
  - Optional *caption* followed by
  - Any number of *col* elements or any number of *colgroup* elements then
  - Optional header followed by optional footer

# XML Element Type

```
<!ELEMENT table  
  (caption?, (col*|colgroup*), thead?, tfoot?, (tbody+|tr+))>
```

- ◆ Child elements of *table* are:
  - Optional *caption* followed by
  - Any number of *col* elements or any number of *colgroup* elements then
  - Optional header followed by optional footer then

# XML Element Type

```
<!ELEMENT table  
  (caption?, (col*|colgroup*), thead?, tfoot?, tbody+|tr+)>
```

## ◆ Child elements of *table* are:

- Optional *caption* followed by
- Any number of *col* elements or any number of *colgroup* elements then
- Optional header followed by optional footer then
- One or more *tbody* elements

# XML Element Type

```
<!ELEMENT table  
  (caption?, (col*|colgroup*), thead?, tfoot?, (tbody+tr+))>
```

- ◆ Child elements of *table* are:
  - Optional *caption* followed by
  - Any number of *col* elements or any number of *colgroup* elements then
  - Optional header followed by optional footer then
  - One or more *tbody* elements or



# XML Element Type

```
<!ELEMENT table  
  (caption?, (col*|colgroup*), thead?, tfoot?, (tbody+(tr+)))>
```

- ◆ Child elements of *table* are:
  - Optional *caption* followed by
  - Any number of *col* elements or any number of *colgroup* elements then
  - Optional header followed by optional footer then
  - One or more *tbody* elements or one or more *tr* elements

# XML Attribute List Declaration

Element type name

```
<!ATTLIST html
  lang          NMTOKEN      #IMPLIED
  xml:lang      NMTOKEN      #IMPLIED
  dir           (ltr|rtl)    #IMPLIED
  id            ID           #IMPLIED
  xmlns         CDATA        #FIXED 'http://www.w3.org/1999/xhtml'>
```

# XML Attribute List Declaration

```
<!ATTLIST html
```

lang	NMTOKEN	#IMPLIED
xml:lang	NMTOKEN	#IMPLIED
dir	(ltr rtl)	#IMPLIED
id	ID	#IMPLIED
xmlns	CDATA	#FIXED 'http://www.w3.org/1999/xhtml'>

Recognized  
attribute names

# XML Attribute List Declaration

```
<!ATTLIST html
  lang          NMTOKEN          #IMPLIED
  xml:lang      NMTOKEN          #IMPLIED
  dir           (ltr|rtl)        #IMPLIED
  id            ID               #IMPLIED
  xmlns         CDATA            #FIXED 'http://www.w3.org/1999/xhtml'>
```

Attribute types  
(data types allowed as attribute values)

# XML Attribute List Declaration

ASCII characters: letter, digit, or . - \_ :

```
<!ATTLIST html
  lang          NMTOKEN #IMPLIED
  xml:lang      NMTOKEN #IMPLIED
  dir           (ltr|rtl) #IMPLIED
  id            ID       #IMPLIED
  xmlns         CDATA    #FIXED 'http://www.w3.org/1999/xhtml'>
```

# XML Attribute List Declaration

Attribute value must be ltr or rtl

```
<!ATTLIST html
  lang          NMTOKEN          #IMPLIED
  xml:lang      NMTOKEN          #IMPLIED
  dir           (ltr|rtl)        #IMPLIED
  id            ID                #IMPLIED
  xmlns         CDATA            #FIXED 'http://www.w3.org/1999/xhtml'>
```

# XML Attribute List Declaration

```
<!ATTLIST html
  lang          NMTOKEN      #IMPLIED
  xml:lang      NMTOKEN      #IMPLIED
  dir           (ltr|rtl)    #IMPLIED
  id            ID           #IMPLIED
  xmlns         CDATA        #FIXED 'http://www.w3.org/1999/xhtml'>
```

Like NMTOKEN but must begin with letter or \_ :  
Attribute value must be unique

# XML Attribute List Declaration

```
<!ATTLIST html
  lang          NMTOKEN      #IMPLIED
  xml:lang      NMTOKEN      #IMPLIED
  dir           (ltr|rtl)    #IMPLIED
  id            ID           #IMPLIED
  xmlns        CDATA        #FIXED 'http://www.w3.org/1999/xhtml'>
```

Any character except XML special characters < and & or the quote character enclosing the attribute value



# XML Attribute List Declaration

TABLE 2.8: Key attribute types used in XHTML 1.0 Strict DTD.

Attribute type	Syntax	Usage
Name token	NMTOKEN	Name (word)
Enumerated	( string1   string2   ... )	List of all possible attribute values
Identifier	ID	Type for id attribute
Identifier reference	IDREF	Reference to an id attribute value
Identifier reference list	IDREFS	List of references to id attribute values
Character data	CDATA	Arbitrary character data (except < and &)

# XML Attribute List Declaration

```
<!ATTLIST html
```

```
  lang          NMTOKEN      #IMPLIED
```

```
  xml:lang      NMTOKEN      #IMPLIED
```

```
  dir           (ltr|rtl)     #IMPLIED
```

```
  id            ID            #IMPLIED
```

```
  xmlns         CDATA        #FIXED 'http://www.w3.org/1999/xhtml'>
```

Attribute default declarations

# XML Attribute List Declaration

TABLE 2.9: XML attribute default-value declarations.

Default type	Syntax
No default value provided by DTD, attribute optional	#IMPLIED
Default provided by DTD, may not be changed	#FIXED followed by any valid value (quoted)
Default provided by DTD, may be overridden by user	Any valid value (quoted)
No default value provided by DTD, attribute required	#REQUIRED

# XML Entity Declaration

- ◆ Entity declaration is essentially a macro
- ◆ Two types of entity:
  - General: referenced from HTML document using &

```
<!ENTITY gt "&#62;">
```

Entity name

# XML Entity Declaration

- ◆ Entity declaration is essentially a macro
- ◆ Two types of entity:
  - General: referenced from HTML document using &

```
<!ENTITY gt
```

```
"&#62;">
```

Replacement text;  
recursively replaced if it is a reference

# XML Entity Declaration

- ◆ Entity declaration is essentially a macro
- ◆ Two types of entity:

- General: referenced from HTML document using `&`  
`<!ENTITY gt ">">`

- Parameter: reference from DTD using `%`

```
<!ENTITY % LanguageCode "NMTOKEN">
```

```
<!ATTLIST html
```

```
  lang          NMTOKEN      #IMPLIED
```

```
  xml:lang      %LanguageCode; #IMPLIED
```

# XML Entity Declaration

- ◆ Entity declaration is essentially a macro
- ◆ Two types of entity:
  - General: referenced from HTML document using `&` `<!ENTITY gt "&#62;">`
  - Parameter: reference from DTD using `%`

```
<!ENTITY % LanguageCode "NMTOKEN">
```

```
<!ATTLIST html
```

```
  lang          NMTOKEN      #IMPLIED
```

```
  xml:lang     %LanguageCode; #IMPLIED
```

# DTD Files

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

*System Identifier*: URL for primary DTD document

- ◆ DTD document contains element type, attribute list, and entity declarations
- ◆ May also contain declaration of external entities: identifiers for secondary DTD documents



# DTD Files

## External entity name

```
<!ENTITY % HTMLlat1 PUBLIC  
    "-//W3C//ENTITIES Latin 1 for XHTML//EN"  
    "xhtml-lat1.ent">  
%HTMLlat1;
```

# DTD Files

```
<!ENTITY % HTMLlat1 PUBLIC  
    "-//W3C//ENTITIES Latin 1 for XHTML//EN"  
    "xhtml-lat1.ent">  
%HTMLlat1;
```

System identifier (relative URL)

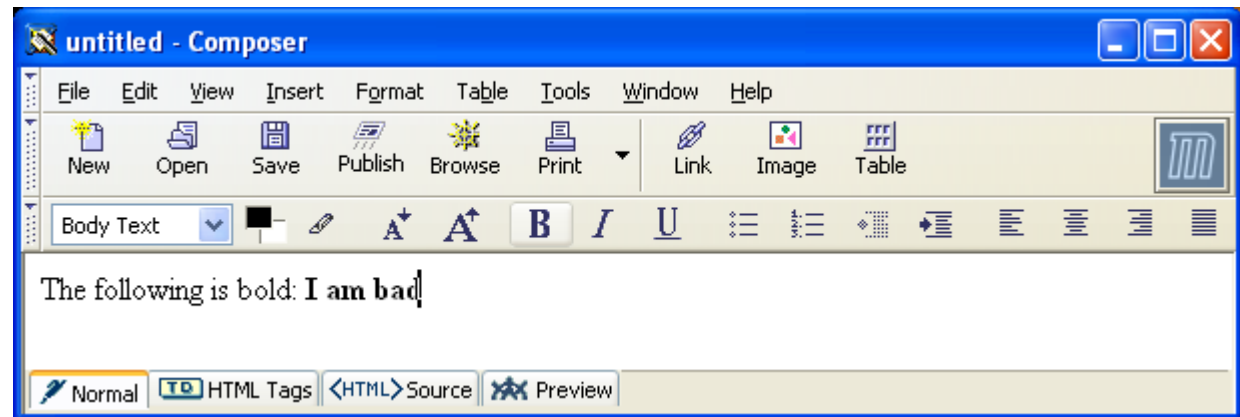
# DTD Files

```
<!ENTITY % HTMLlat1 PUBLIC  
    "-//W3C//ENTITIES Latin 1 for XHTML//EN"  
    "xhtml-lat1.ent">  
%HTMLlat1;
```

Entity reference; imports content (entity declarations, called *entity set*) of external entity at this point in the primary DTD

# HTML Creation Tools

- ◆ Mozilla Composer



- ◆ Microsoft FrontPage

- ◆ Macromedia Dreamweaver

- ◆ Etc.