

2.1 Origins and Evolution of HTML

- HTML was defined with SGML
- Original intent of HTML: General layout of documents that could be displayed by a wide variety of computers
- Recent versions:
 - HTML 4.0 – 1997
 - Introduced many new features and deprecated many older features
 - HTML 4.01 - 1999 - A cleanup of 4.0
 - XHTML 1.0 - 2000
 - Just 4.01 defined using XML, instead of SGML
 - XHTML 1.1 – 2001
 - Modularized 1.0, and drops frames
- Although the value of the consistent and coherent syntax rules of XHTML were widely recognized and accepted, its draconian error handling was not, so XHTML documents are served as HTML
- W3C worked on XHTML 2.0; WHAT worked on a new version of HTML

2.1 Origins and Evolution of HTML

(continued)

- In 2009, XHTML 2.0 was stopped; W3C took over development of HTML5
- Reasons to use XHTML syntax rules:
 1. HTML has lax syntax rules, leading to sloppy and sometimes to ambiguous documents
 - XHTML syntax is much more strict, leading to clean and clear documents in a standard form
 2. HTML processors do not even enforce the few syntax rule that do exist in HTML
 3. The syntactic correctness of XHTML documents can be validated
- In this book, HTML5 is used, but with XHTML syntax rules

2.2 Basic Syntax

- Elements are defined by tags (markers)
 - Tag format:
 - Opening tag: <name>
 - Closing tag: </name>
- The opening tag and its closing tag together specify a container for the *content* they enclose

2.2 Basic Syntax (continued)

- Not all tags have content
 - If a tag has no content, its form is `<name />`
- The container and its content together are called an *element*
- If a tag has attributes, they appear between its name and the right bracket of the opening tag
- Comment form: `<!-- ... -->`
- Browsers ignore comments, unrecognized tags, line breaks, multiple spaces, and tabs
- Tags are suggestions to the browser, even if they are recognized by the browser
- In XHTML, element and attribute names must be in all lowercase letters
- In HTML, they can be any combination of uppercase and lowercase

2.3 Standard XHTML Document Structure

- Every XHTML document must begin with:

```
<!DOCTYPE html>
```

- `<html>`, `<head>`, `<title>`, and `<body>` are required in every document (in XHTML, not HTML)
- The whole document must have `<html>` as its root
- `html` must have the `lang` attribute:

```
<html lang = "en">
```

 (this one is for English)
- A document consists of a head and a body
- The `<title>` tag is used to give the document a title, which is normally displayed in the browser's window title bar (at the top of the display)
- The `meta` tag is used to provide the character set used

```
<meta charset = "utf-8" />
```

2.4 Basic Text Markup

- Text is normally placed in paragraph elements

- *Paragraph Elements*

- The `<p>` tag breaks the current line and inserts a blank line - the new line gets the beginning of the content of the paragraph
- The closing tag is required in XHTML, not in HTML

```
<!DOCTYPE html>
<!-- greet.html
      A trivial document
-->
<html lang = "en">
  <head>
    <title> Our first document </title>
    <meta charset = "utf-8" />
  </head>
  <body>
    <p>
      Greetings from your Webmaster!
    </p>
  </body>
</html>
```

2.4 Basic Text Markup (continued)

- *Line breaks*

- The effect of the `
` tag is the same as that of `<p>`, except for the blank line (in HTML, it could be just `
`)
- No closing tag!

- *Example of paragraphs and line breaks*

On the plains of hesitation `<p>` bleach the bones of countless millions `</p>` `
` who, at the dawn of victory `
` sat down to wait, and waiting, died.

- *Typical display of this text:*

On the plains of hesitation

bleach the bones of countless millions
who, at the dawn of victory
sat down to wait, and waiting, died.

- *Preserving whitespace*

- The text content of a `<pre>` element is displayed as it is entered

2.4 Basic Text Markup (continued)

- Headings

- Six sizes, 1 - 6, specified with `<h1>` to `<h6>`
- 1, 2, and 3 use font sizes that are larger than the default font size
- 4 uses the default size
- 5 and 6 use smaller font sizes

```
<!DOCTYPE html>
<!-- headings.html
  An example to illustrate headings
-->
<html lang = "en">
  <head>
    <title> Headings </title>
    <meta charset = "utf-8" />
  </head>
  <body>
    <h1> Aidan's Airplanes (h1) </h1>
    <h2> The best in used airplanes (h2) </h2>
    <h3> "We've got them by the hangarful" (h3)
    </h3>
    <h4> We're the guys to see for a good used
        airplane (h4) </h4>
    <h5> We offer great prices on great planes
        (h5) </h5>
    <h6> No returns, no guarantees, no refunds,
        all sales are final (h6) </h6>
  </body>
</html>
```

2.4 Basic Text Markup (continued)

Aidan's Airplanes (h1)

The best in used airplanes (h2)

"We've got them by the hangarful" (h3)

We're the guys to see for a good used airplane (h4)

We offer great prices on great planes (h5)

No returns, no guarantees, no refunds, all sales are final! (h6)

- Blockquotes

- Content of `<blockquote>`

- To set a block of text off from the normal flow and appearance of text

- Browsers often indent, and sometimes italicize

- Font Styles and Sizes (can be nested)

- Emphasis - `` (often set in italics)
- Strong - `` (often set in boldface)
- Monospace - `<code>` (often set in Courier)

2.4 Basic Text Markup (continued)

- ``, ``, and `<code>` are not affected if they appear in the content of a `<blockquote>`, unless there is a conflict (e.g., `` (italics))

- *Superscripts and subscripts*

- Subscripts with `<sub>`
- Superscripts with `<sup>`

Example: `x₂³`

Display: x_2^3

- Inline versus block elements

- Block elements CANNOT be nested in inline elements (in XHTML)

2.4 Basic Text Markup (continued)

- *Character Entities*

Char.	Entity	Meaning
&	<code>&amp;</code>	Ampersand
<	<code>&lt;</code>	Less than
>	<code>&gt;</code>	Greater than
"	<code>&quot;</code>	Double quote
'	<code>&apos;</code>	Single quote
$\frac{1}{4}$	<code>&frac14;</code>	One quarter
$\frac{1}{2}$	<code>&frac12;</code>	One half
$\frac{3}{4}$	<code>&frac34;</code>	Three quarters
°	<code>&deg;</code>	Degree
(space)	<code>&nbsp;</code>	Non-breaking space
©	<code>&copy;</code>	Copyright
€	<code>&euro;</code>	Euro

- Horizontal rules

- `<hr />` draws a line across the display, after a line break

- The `meta` element (mainly for search engines)
Used to provide additional information about a document, with attributes, not content

2.5 Images

- Formats:

- GIF (Graphic Interchange Format)
 - 8-bit color (256 different colors)
 - JPEG (Joint Photographic Experts Group)
 - 24-bit color (16 million different colors)
 - Both use compression, but JPEG compression is better
 - Portable Network Graphics (PNG)
 - Relatively new
 - Should eventually replace both gif and jpeg
 - Files are bigger than jpeg – no lost data!
 - Images are inserted into a document with the `` tag with the `src` attribute
 - The `alt` attribute is required by XHTML
 - Purposes:
 1. Non-graphical browsers
 2. Browsers with images turned off
- ```
<img src = "comets.jpg"
 alt = "Picture of comets" />
```
- The `<img>` tag has 30 different attributes, including width and height (in pixels)

## 2.5 Images (continued)

```
<!DOCTYPE html>
<!-- image.html
 An example to illustrate an image
-->
<html lang = "en">
 <head>
 <title> Images </title>
 <meta charset = "utf-8" />
 </head>
 <body>
 <h1> Aidan's Airplanes </h1>
 <h2> The best in used airplanes </h2>
 <h3> "We've got them by the hangarful"
 </h3>
 <h2> Special of the month </h2>
 <p>
 1960 Cessna 210

 577 hours since major engine overhaul

 1022 hours since prop overhaul

 <img src = "c210new.jpg"
 alt = "Picture of a Cessna 210"/>

 Buy this fine airplane today at a
 remarkably low price

 Call 999-555-1111 today!
 </p>
 </body>
</html>
```

## 2.5 Images (continued)

### Aidan's Airplanes

The best in used airplanes

"We've got them by the hangarful"

#### Special of the month

1960 Cessna 210  
577 hours since major engine overhaul  
1022 hours since prop overhaul



Buy this fine airplane today at a remarkably low price  
Call 999-555-1111 today!

## 2.5 Images (continued)

### - HTML Validation

#### - Replace DOCTYPE with:

```
<!DOCTYPE html PUBLIC "-//W3C XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11-strict.dtd"
```

#### - Replace xmlns value in the html element with:

```
"http://www.w3.org/1999/xhtml"
```

#### - Comment out the meta element

#### - Download the validation tool from:

<http://totalvalidator.com>

#### - Use it on the document

## 2.6 Hypertext Links

### - A link is specified with the href (*hypertext reference*) attribute of <a> (the anchor tag)

### - The content of <a> is the visual link in the document

### - If the target is a whole document (not the one in which the link appears), the target need not be specified in the target document as being the target

## 2.6 Hypertext Links (continued)

```
<!DOCTYPE html>
<!-- link.html
 An example to illustrate a link
-->
<html lang = "en">
 <head>
 <title> Links </title>
 <meta charset = "utf-8" />
 </head>
 <body>
 <h1> Aidan's Airplanes </h1>
 <h2> The best in used airplanes </h2>
 <h3> "We've got them by the hangarful"
 </h3>
 <h2> Special of the month </h2>
 <p>
 1960 Cessna 210

 Information on the Cessna 210
 </p>
 </body>
</html>
```

## 2.6 Hypertext Links (continued)

### Aidan's Airplanes

The best in used airplanes

"We've got them by the hangarful"

Special of the month

1960 Cessna 210

[Information on the Cessna 210](#)

#### 1960 Cessna 210 Information

577 hours since major engine overhaul  
622 hours since prop overhaul



Buy this fine airplane today at a remarkably low price  
Call 999-555-1111 today!

## 2.6 Hypertext Links (continued)

- If the target is not at the beginning of the document, the target spot must be marked
- Target labels can be defined in many different tags with the `id` attribute, as in

```
<h1 id = "baskets"> Baskets </h1>
```

- The link to an `id` must be preceded by a pound sign (#); If the `id` is in the same document, this target could be

```

 What about baskets?
```

- If the target is in a different document, the document reference must be included

```
 Baskets
```

- Links can have images:

```

 <img src = "smallplane.jpg"
 alt = "Small picture of an airplane" />
 Info on C210
```

## 2.7 Lists

### - Unordered lists

- The list is the content of the `<ul>` tag
- List elements are the content of the `<li>` tag

```
<h3> Some Common Single-Engine Aircraft </h3>

 Cessna Skyhawk
 Beechcraft Bonanza
 Piper Cherokee

```



### - Ordered lists

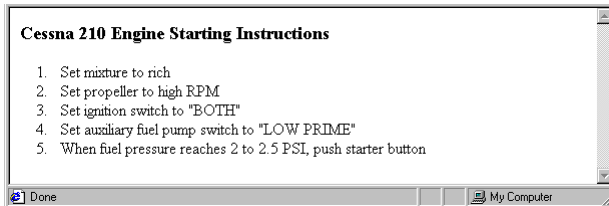
- The list is the content of the `<ol>` tag
- Each item in the display is preceded by a sequence value

## 2.7 Lists (continued)

```
<h3> Cessna 210 Engine Starting Instructions
</h3>

 Set mixture to rich
 Set propeller to high RPM
 Set ignition switch to "BOTH"
 Set auxiliary fuel pump switch to
 "LOW PRIME"
 When fuel pressure reaches 2 to 2.5
 PSI, push starter button

```



### - Nested lists

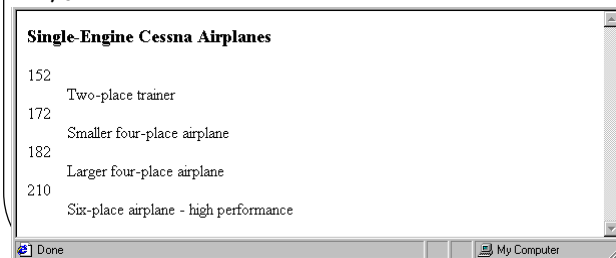
- Any type list can be nested inside any type list
- The nested list must be in a list item

## 2.7 Lists (continued)

### - Definition lists (for glossaries, etc.)

- List is the content of the <dl> tag
- Terms being defined are the content of the <dt> tag
- The definitions themselves are the content of the <dd> tag

```
<h3> Single-Engine Cessna Airplanes </h3>
<dl>
 <dt> 152 </dt>
 <dd> Two-place trainer </dd>
 <dt> 172 </dt>
 <dd> Smaller four-place airplane </dd>
 <dt> 182 </dt>
 <dd> Larger four-place airplane </dd>
 <dt> 210 </dt>
 <dd> Six-place airplane - high performance
</dd>
</dl>
```



## 2.8 Tables

- A table is a matrix of cells, each possibly having content
  - The cells can include almost any element
  - Some cells have row or column labels and some have data
- A table is specified as the content of a `<table>` tag
- In HTML5, tables do not have lines between the rows or between the columns
  - We can add those with Cascading Style Sheets, as will be discussed in Chapter 3
- Tables are given titles with the `<caption>` tag, which can immediately follow `<table>`

## 2.8 Tables (continued)

- Each row of a table is specified as the content of a `<tr>` tag
- The row headings are specified as the content of a `<th>` tag
- The contents of a data cell is specified as the content of a `<td>` tag

```
<table>
<caption> Fruit Juice Drinks </caption>
<tr>
 <th> </th>
 <th> Apple </th>
 <th> Orange </th>
 <th> Screwdriver </th>
</tr>
<tr>
 <th> Breakfast </th>
 <td> 0 </td>
 <td> 1 </td>
 <td> 0 </td>
</tr>
<tr>
 <th> Lunch </th>
 <td> 1 </td>
 <td> 0 </td>
 <td> 0 </td>
</tr>
</table>
```

## 2.8 Tables (continued)

Fruit Juice Drinks			
	Apple	Orange	Screwdriver
Breakfast	0	1	0
Lunch	1	0	0
Dinner	0	0	1

- A table can have two levels of column labels

- If so, the `colspan` attribute must be set in the `<th>` tag to specify that the label must span some number of columns

```
<tr>
 <th colspan = "3"> Fruit Juice Drinks </th>
</tr>
<tr>
 <th> Orange </th>
 <th> Apple </th>
 <th> Screwdriver </th>
</tr>
```

Fruit Juice Drinks			
	Apple	Orange	Screwdriver

## 2.8 Tables (continued)

- If the rows have labels and there is a spanning column label, the upper left corner must be made larger, using `rowspan`

```
<table>
 <tr>
 <td rowspan = "2"> </td>
 <th colspan = "3"> Fruit Juice Drinks
 </th>
 </tr>
 <tr>
 <th> Apple </th>
 <th> Orange </th>
 <th> Screwdriver </th>
 </tr>
 ...
</table>
```

Fruit Juice Drinks and Meals			
	Fruit Juice Drinks		
	Apple	Orange	Screwdriver
Breakfast	0	1	0
Lunch	1	0	0
Dinner	0	0	1

## 2.8 Tables (continued)

- **Table Sections**
  - Header, body, and footer, which are the elements: `thead`, `tbody`, and `tfoot`
- If a document has multiple `tbody` elements, they are separated by thicker horizontal lines
- **Uses of Tables**
  - In the past, tables were used to align elements in rows and columns – general layout
  - That use of tables is now frowned upon
  - Use Cascading Style Sheets to place elements in rows and columns – general layout
  - Use tables only when the information is naturally tabular

## 2.9 Forms

- A form is the usual way information is gotten from a browser user to a server
- HTML has tags to create a collection of objects that implement this information gathering
  - The objects are called *widgets* or *controls* or *components*
- When the *Submit* button of a form is clicked, the form's values are sent to the server for processing
- All of the widgets, or components of a form are defined in the content of a `<form>` tag
  - The only required attribute of `<form>` is `action`, which specifies the URL of the application that is to be called when the *Submit* button is clicked (XHTML requires it; HTML does not)

```
action =
 "http://www.cs.ucp.edu/cgi-bin/survey.pl"
```

- If the form has no action, the value of `action` is the empty string

## 2.9 Forms (continued)

- The `method` attribute of `<form>` specifies one of the two possible techniques of transferring the form data to the server, `get` and `post`
  - The default is `get`

- `get` and `post` are discussed in Chapter 10

### - Widgets

- Many are created with the `<input>` tag
- The `type` attribute of `<input>` specifies the kind of widget being created

#### 1. Text

- Creates a horizontal box for text input
- Default size is 20; it can be changed with the `size` attribute
- If more characters are entered than will fit, the box is scrolled (shifted) left

## 2.9 Forms (continued)

- If you don't want to allow the user to type more characters than will fit, set `maxlength`, which causes excess input to be ignored

```
<input type = "text" name = "Phone"
 size = "12" />
```

- Widgets should be placed in label elements

```
<label> Phone: <input type = "text"
 name = "phone" />
</label>
```

2. **Password** – just like text except asterisks are displayed, rather than the input characters

3. **Checkboxes** - to collect multiple choice input

- Every checkbox requires a `value` attribute, which is the widget's value in the form data when the checkbox is 'checked'
  - A checkbox that is not 'checked' contributes no value to the form data
- By default, no checkbox is initially 'checked'
- To initialize a checkbox to 'checked', the `checked` attribute must be set to `"checked"`

## 2.9 Forms (continued)

### - Widgets (continued)

```
Grocery Checklist
<form action = "">
 <p>
 <label> <input type = "checkbox" name = "groceries"
 value = "milk" checked = "checked" />
 Milk </label>
 <label> <input type = "checkbox" name = "groceries"
 value = "bread" />
 Bread </label>
 <label> <input type = "checkbox" name = "groceries"
 value = "eggs" />
 Eggs </label>
 </p>
</form>
```



- 4. Radio Buttons** - collections of checkboxes in which only one button can be 'checked' at a time
- Every button in a radio button group **MUST** have the same name

## 2.9 Forms (continued)

### - Widgets (continued)

#### 4. Radio Buttons (continued)

- If no button in a radio button group is 'pressed', the browser often 'presses' the first one

```
Age Category
<form action = "">
 <p>
 <label> <input type = "radio" name = "age"
 value = "under20" checked = "checked" />
 0-19 </label>
 <label> <input type = "radio" name = "age"
 value = "20-35" /> 20-35 </label>
 <label> <input type = "radio" name = "age"
 value = "36-50" /> 36-50 </label>
 <label> <input type = "radio" name = "age"
 value = "over50" /> Over 50 </label>
 </p>
</form>
```

## 2.9 Forms (continued)

### - Widgets (continued)



Age Category

☒ 0-19 ☐ 20-35 ☐ 36-50 ☐ Over 50

### 5. The <select> tag

- There are two kinds of menus, those that behave like checkboxes and those that behave like radio buttons (the default)
- Menus that behave like checkboxes are specified by including the `multiple` attribute, which must be set to "multiple"
- The `name` attribute of `<select>` is required
- The `size` attribute of `<select>` can be included to specify the number of menu items to be displayed (the default is 1)
- If `size` is set to `> 1` or if `multiple` is specified, the menu is displayed as a pop-up menu

## 2.9 Forms (continued)

### - Widgets (continued)

### 5. <select> (continued)

- Each item of a menu is specified with an `<option>` tag, whose pure text content (no tags) is the value of the item
- An `<option>` tag can include the `selected` attribute, which when assigned "selected" specifies that the item is preselected

Grocery Menu - milk, bread, eggs, cheese

```
<form action = "">
 <p>
 With size = 1 (the default)
 <select name = "groceries">
 <option> milk </option>
 <option> bread </option>
 <option> eggs </option>
 <option> cheese </option>
 </select>
 </p>
</form>
```

## 2.9 Forms (continued)

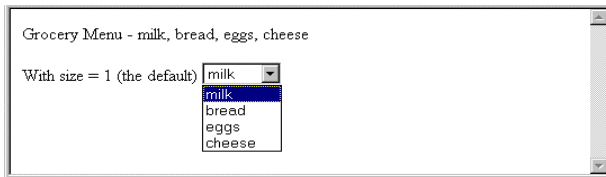
### - Widgets (continued)



Grocery Menu - milk, bread, eggs, cheese

With size = 1 (the default)

### - After clicking the menu:

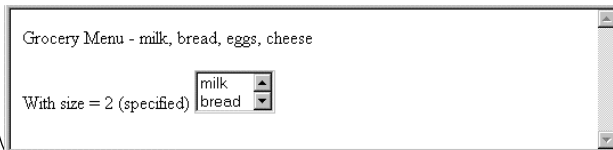


Grocery Menu - milk, bread, eggs, cheese

With size = 1 (the default)

- milk
- bread
- eggs
- cheese

### - After changing size to 2:



Grocery Menu - milk, bread, eggs, cheese

With size = 2 (specified)

- milk
- bread

## 2.9 Forms (continued)

### - Widgets (continued)

#### 6. Text areas - created with `<textarea>`

- Usually include the `rows` and `cols` attributes to specify the size of the text area
- Default text can be included as the content of `<textarea>`
- Scrolling is implicit if the area is overfilled

Please provide your employment aspirations

```
<form action = "">
 <p>
 <textarea name = "aspirations" rows = "3"
 cols = "40">
 (Be brief and concise)
 </textarea>
 </p>
</form>
```



Please provide your employment aspirations

(Be brief and concise)

## 2.9 Forms (continued)

### - Widgets (continued)

#### 7. Action buttons

- Both are created with `<input>`

```
<input type = "reset"
 value = "Reset Form" />
<input type = "submit"
 value = "Submit Form" />
```

- Submit has two actions:

1. Encode the data of the form
2. Request that the server execute the server-resident program specified as the value of the `action` attribute of `<form>`

- A Submit button is required in every form

→SHOW `popcorn.html` and display it

## 2.10 The `audio` Element

- Prior to HTML5, a plug-in was required to play sound while a document was being displayed
- Audio encoding algorithms are called *audio codecs* – e.g., MP3, Vorbis
- Coded audio data is stored in containers—e.g., Ogg, MP3, and Wav (file name extension indicates the container, not the audio code)
- Vorbis code is stored in Ogg containers
- MP3 code is stored in MP3 containers
- Wav code is stored in Wav containers

## 2.10 The audio Element (continued)

### - General syntax:

```
<audio attributes>
 <source src = "filename1" >
 ...
 <source src = "filenamen" >
```

Your browser does not support the audio element  
</audio>

- Browser chooses the first audio file it can play and skips the content; if none, it displays the content
- Different browsers have different audio capabilities
- The controls attribute, which is set to controls", creates a start/stop button, a clock, a progress slider, total time of the file, and a volume slider

## 2.10 The audio Element (continued)

```
<!DOCTYPE html>
<!-- audio.html
 Test the audio element
-->
<html lang = "en" >
 <head>
 <title> Test audio element </title>
 <meta charset = "utf-8" />
 </head>
 <body>
 This is a test of the audio element
 <audio controls = "controls" >
 <source src = "nineoneone.ogg" />
 <source src = "nineoneone.wav" />
 <source src = "nineoneone.mp3" />
 Your browser does not support the audio
 element
 </audio>
 </body>
</html>
```

## 2.11 The video Element

- Prior to HTML5, there was no standard way to play video clips while a document was being displayed
- Video codecs are stored in containers
- Video codecs:
  - H.264 (MPEG-4 AVC) – can be stored in MPEG-4
  - Theora – can be stored in any container
  - VP8—can be stored in any container
- Different browsers support different codecs
- The width and height attributes set the screen size
- The autoplay attribute, set to "autoplay", specifies that the video should play as soon as it is ready
- The preload attribute, set to "preload", specifies that the video should be loaded as soon as the document is loaded
- The controls attribute, set to "controls", is like that of the audio element

## 2.11 The video Element (continued)

```
<!DOCTYPE html>
<!-- testvideo.html
 test the video element
-->
<html lang = "en">
 <head>
 <meta charset = "UTF-8" />
 <title> test video element </title>
 </head>
 <body>
 This is a test of the video element.....
 <video width = "600" height = "500"
 autoplay = "autoplay"
 controls = "controls"
 preload = "preload">
 <source src = "NorskTippingKebab.mp4" />
 <source src = "NorskTippingKebab.ogv" />
 <source src = "NorskTippingKebab.webm" />
 Your browser does not support the video
 element
 </video>
 </body>
</html>
```

## 2.12 Organization Elements

### - Header Elements

- `hgroup` – a container for header information

```
<hgroup>
 <header>
 <h1> The Podunk Press </h1>
 <h2> "All the news we can fit" </h2>
 </header>
 -- table of contents -
</hgroup>
```

### - Footer Elements

- `footer` – a container for footer information

```
<footer>
 © The Podunk Press, 2012

 Editor in Chief: Squeak Martin
</footer>
```

- The `section` Element – a container for sections
- The `article` Element – a container for self-contained part of a document (from another source)
- The `aside` Element – a container for tangential info
- The `nav` Element – navigation sections (list of links)

## 2.13 The `time` Element

- For putting a time stamp on a document

- Two parts, text and machine-readable (`datetime`)
- `datetime` attribute (optional) – the machine-readable part

- Date part: 4-digit year, a dash, 2-digit month, a dash, 2-digit day of the month ( "2012-08-29" )

- Time (optional) format: `T09:00`

- Text part is given as the content of `time`

```
<time datetime = "2012-08-29T09:00 ">
 August 8, 2012 9:00 am
</time>
```

- The two parts need not specify the same date

- Deficiencies:

1. Dates prior to the Christian era are not possible
2. No approximations

## **2.14 Syntactic Differences between HTML & XHTML**

- Case sensitivity
- Closing tags
- Quoted attribute values
- Explicit attribute values
- `id` and `name` attributes
- Element nesting