ID1354
Internet Applications

HTML

Leif Lindbäck, Nima Dokoohaki

leifl@kth.se, nimad@kth.se

SCS/ICT/KTH

HTML

INTRODUCTION TO HTML

Introduction to HTML

- HTML defines parts of documents (headers, paragraphs, images, etc). **Note that it does not define how these parts are rendered in a browser.**

- HTML is maintained by W3C, which is the main international standards organization for the World Wide Web.

Current HTML Versions

- HTML 4.0/4.01 – 1997/1999
  Introduced many new features and deprecated many older features.
- XHTML 1.0/1.1 – 2000/2001
  Same content as HTML, but much more strict, leading to clean and clear documents in a standard form.
- HTML 5 – 2012
  Unites the many different technologies in use on the WWW today, such as HTML, XHTML, standards introduced by browser manufacturers, common practices etc. Also introduces new features, including many APIs.

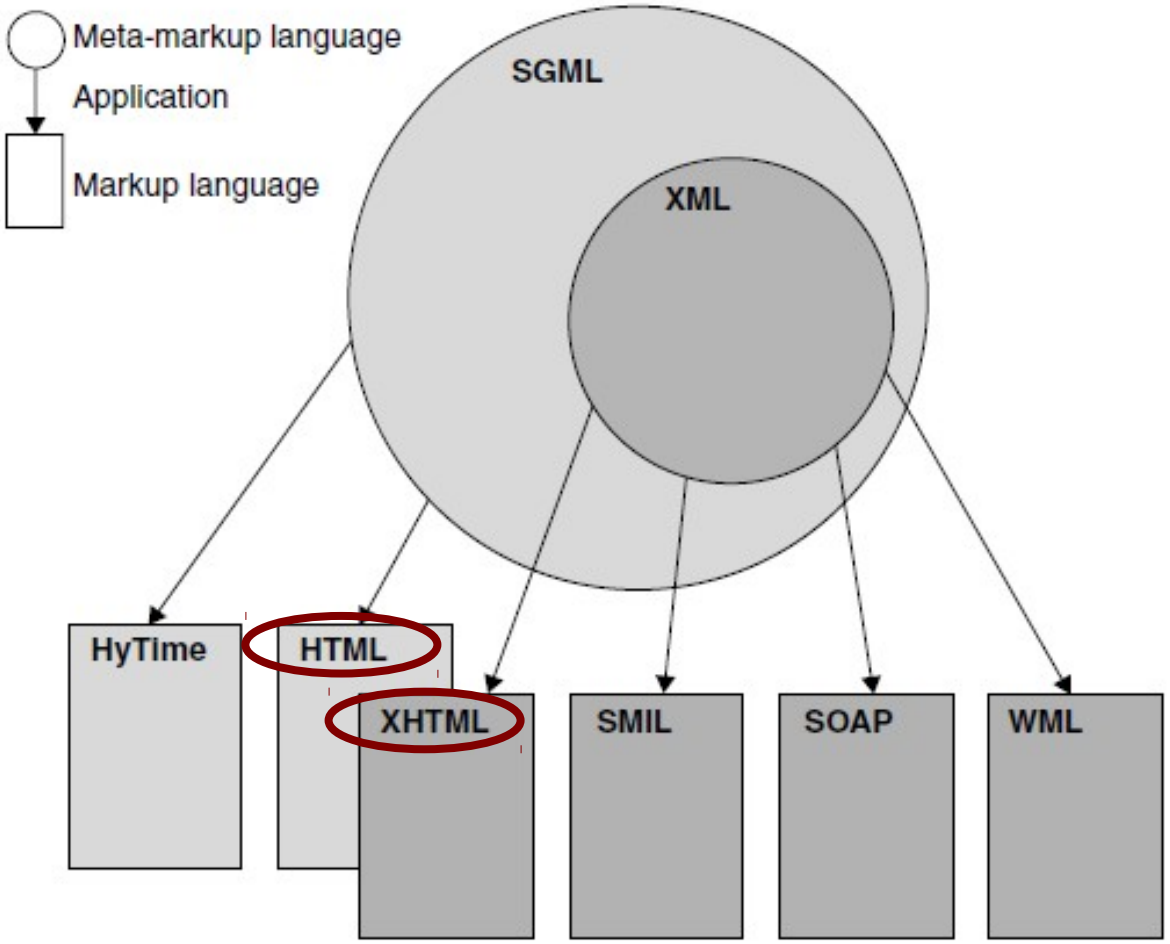
Which HTML Version Shall We Use?

- When developing new documents, use HTML5 with XHTML syntax. This is the case in the course text book.
- Reasons to use XHTML syntax rules:
  1. 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 rules that do exist in HTML.
  3. The syntactic correctness of XHTML documents can be validated.
- Not all HTML5 features are implemented in all browsers. Implementations status can be checked at **`http://caniuse.com`**

The Purpose of HTML

- The purpose of HTML is to define parts of a document.
- Layout should not be coded in HTML, but in CSS.

(X)HTML are SGML/XML applications



Taken from Shklar, Rosen: *Web Application Architecture*.

Standard Generalized Markup Language, SGML

- SGML is a standard SGML for defining generalized markup languages for documents or data structures.
- The markup shall describe the document's structure and attributes, rather than the processing to be performed on it.
- HTML was a SGML application before HTML5

Extensible Markup Language, XML

- XML is a subset of SGML, and also introduces additional restrictions on syntax compared to SGML. The purpose is to make it easier to parse.
- XHTML is a XML application. HTML5 is not, but may be written in XML syntax.

XML, and HTML, Terminology

- A **tag** defines an **element**. The HTML below, which states that the heading should be *My Homepage*, has the **opening tag** **<h1>**, the **closing tag** **</h1>** and the whole line is an element.

```
<h1>My Homepage</h1>
```

- The text between the opening and closing tag, **My Homepage** in the example above, is the elements **content**.

XML, and HTML, Terminology (Cont'd)

- There are **empty elements**, for example, the element **<img/>** defines an image.
- Tags may have **attributes**. The **src** attribute below defines the file containing the image.

```
<img src="the-image.gif"/>
```

- A **nested element** is located between the start and end tags of another element, as ** some text ** in the html below.

```
<div>
   <p>some text</p>
</div>
```

Basic XML Syntax Rules

- The document contains only Unicode characters.
- The special characters (e.g. **<** or **&**) are used only for markup.
- Tags are correctly nested, with none missing and none overlapping.
- Tags are case-sensitive, the start and end tags must match exactly. Tag names cannot contain any of the characters **!"#%&'()*+,/;<=>?@[\]^`{|}~**, nor a space character, and cannot start with **-**, **.**, or a numeric digit.
- A single root element contains all the other elements

HTML

BASIC HTML SYNTAX

Basic HTML Syntax

- XHTML documents must have XML syntax. HTML documents do not have that requirement.
- As mentioned above, it is a good habit to always use xml syntax since it leads to clean and clear documents.
- Comments in HTML have the form
  `<!-- a comment -->`
- Browsers ignore comments, unrecognized tags and other errors, line breaks, multiple spaces, and tabs.

Basic HTML Syntax (Cont'd)

- 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

HTML Errors

- HTML syntax errors or tags are silently ignored.
- This might cause much extra work.
- Always validate your HTML files! Use for example the W3C HTML validator, **`http://validator.w3.org/`**

Standard HTML5 Document Structure

- Every HTML5 and XHTML document must begin with **`<!DOCTYPE html>`**
- **`<html>`**, **`<head>`**, **`<title>`**, and **`<body>`** are required in every document.
- The whole document must have **`<html>`** as its root.

Standard HTML5 Document Structure (Cont'd)

- A document consists of a head, **<head>**, and a body, **<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 required to provide the character set used:

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

Standard HTML5 Document Structure (Cont'd)

- A minimum HTML5 Document:

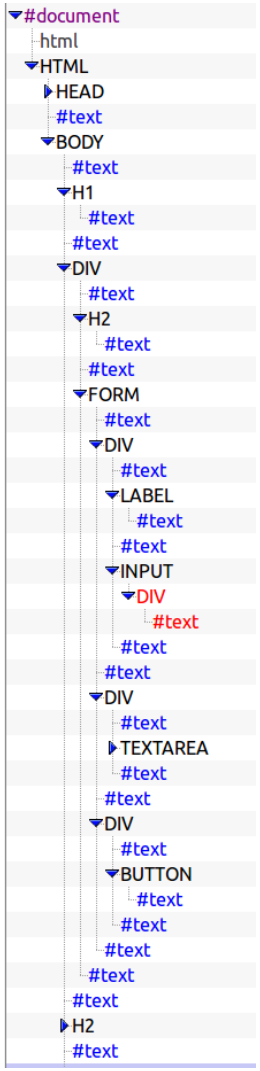
```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Document Title</title>
    </head>
    <body>
        Content of the document.....
    </body>
</html>
```

Question 1

DOCUMENT OBJECT MODEL, DOM

HTML

The Document Object Model, DOM



- The browser creates a tree-like structure, called Document Object Model (DOM), which represents the elements in the HTML document.
- The picture to the left is a part of the DOM tree for the course's chat program.
- The DOM provides a JavaScript API, defined by the W3C.

HTML

BASIC TEXT MARKUP

Basic Text Markup

Paragraph Elements

- 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.

Example

```
<!DOCTYPE html>

<html>
  <head>
    <title> Document With
       Paragraph </title>
    <meta charset = "utf-8" />
  </head>
  <body>
    <p>
      Greetings from your
Webmaster!
    </p>
  </body>
</html>
```

Basic Text Markup (Cont'd)

- Line breaks

- The effect of the **
** tag is the same as that of <p>, except for the blank line.

- This is an empty tag.

Basic Text Markup (Cont'd)

- Example of paragraphs and line breaks

```
On the plains of hesitation <p> bleach the
bones of countless millions </p> <br/>
who, at the dawn of victory <br /> 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.
```

Basic Text Markup (Cont'd)

- To keep text indentation, use the **`<pre>`** element. It preserves whitespaces and displays text as it is entered.

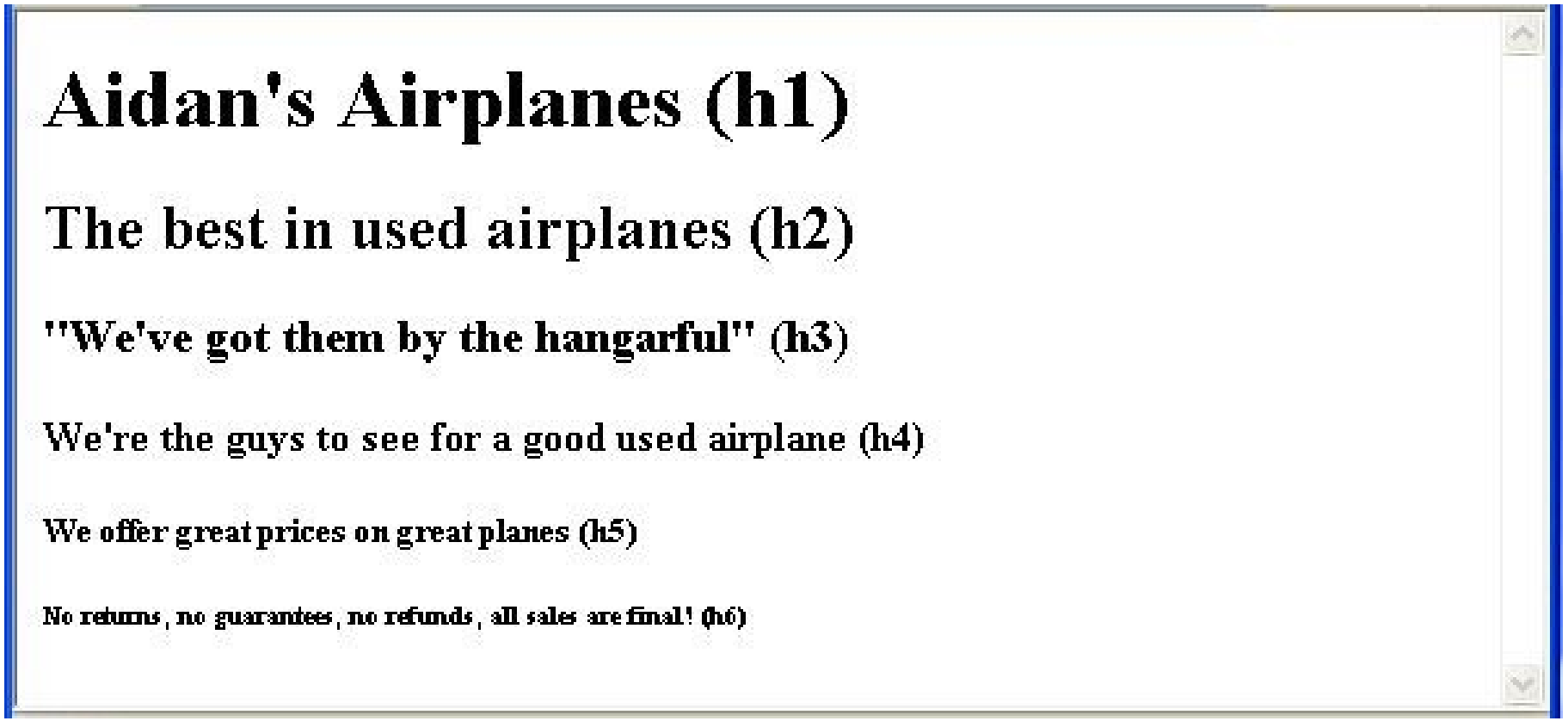
Basic Text Markup (Cont'd)

Headings

Six sizes, 1-6, specified with **<h1>** to **<h6>**

```
<!DOCTYPE html>
<html>
  <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>
```

Example for headings



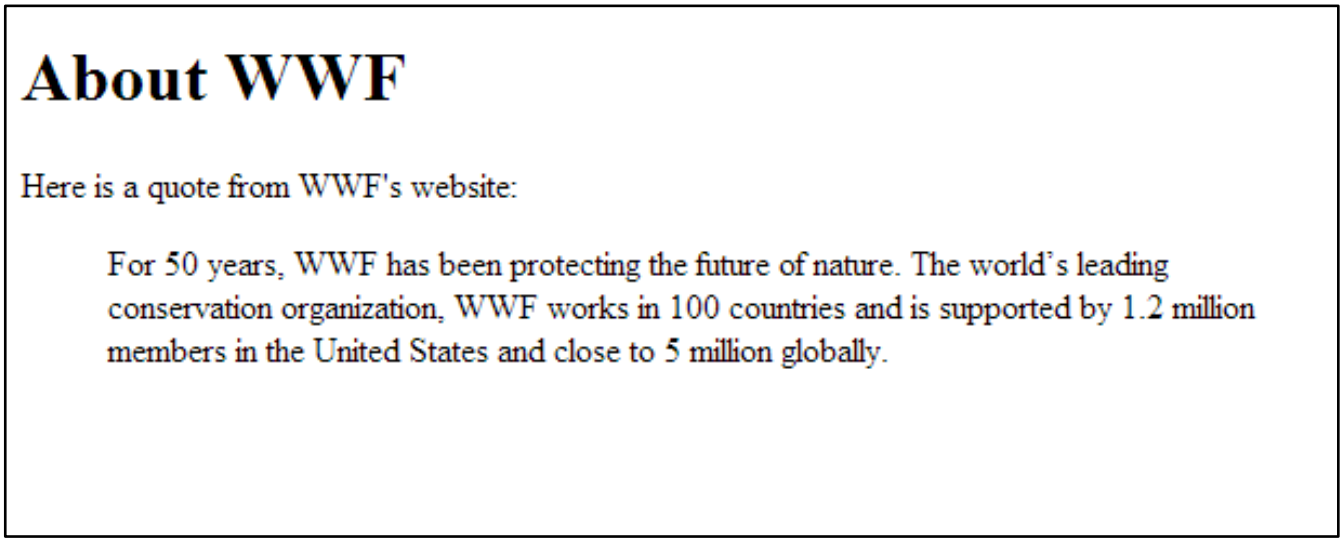
Basic Text Markup (Cont'd)

- Blockquotes
  - Content of **<blockquote>**
  - To set a block of text off from the normal flow and appearance of text

- Font Styles and Sizes
  - Emphasis - ****
  - Strong - ****
  - Code - **<code>**
  - Do not use **<b>** (bold) or **<i>** (italic), that is layout, which shall be coded in CSS.

BlockQuote example

```
<!DOCTYPE html>
<html>
<body>

<h1>About WWF</h1>
<p>Here is a quote from WWF's website:</p>

<blockquote>
   For 50 years, WWF has been protecting the future of nature. The world's
   leading conservation organization, WWF works in 100 countries and is supported
   by 1.2 million members in the United States and close to 5 million globally.
</blockquote>

</body>
</html>
```



Basic Text Markup (Cont'd)

Superscripts and subscripts

- Subscripts: **<sub>**
- Superscripts: **<sup>**

HTML:

```
x<sub>2</sub><sup>3</sup>
```

Display: x_2^3

Basic Text Markup (Cont'd)

Character Entities

Char.	Entity	Meaning
&	&	Ampersand
<	<	Less than
>	>	Greater than
"	"	Double quote
'	'	Single quote
¼	¼	One quarter
½	½	One half
¾	¾	Three quarters
°	°	Degree
(space)		Non-breaking space

HTML:
ampersand: &
less than: <

Display:
ampersand: &
less than: <

A non-breaking space requests the browser not to break a line at that space.

Basic Text Markup (Cont'd)

Horizontal rules

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

HTML

IMAGES

Images

Allowed 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
- **PNG** (Portable Network Graphics)
  - 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 **`<img/>`** tag.
- The **`src`** attribute specifies the image file.
- The **`alt`** attribute defines a text that is displayed if the image can not be shown.

```
<!DOCTYPE html>
<html>
  <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 <br/>
      577 hours since major engine overhaul  <br/>
      1022 hours since prop overhaul  <br/><br/>
      <img src = "c210new.jpg“
           alt = "Picture of a Cessna210"/>  <br/>
      Buy this fine airplane today at a remarkably low price <br/>
      Call 999-555-1111 today!
    </p>
  </body>
</html>
```

Image Example

Image Example (Cont'd)

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!

Question 2

HYPER TEXT LINKS

Hypertext Links

- Hypertext is text which contains links to other texts.

- A link is specified with the **`href`** (*h*ypertext *ref*erence) attribute of **`<a>`** (the anchor tag)

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

Link Example

```
<!DOCTYPE html>
<html>
  <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 <br/>
      <a href = "C210data.html">
        Information on the Cessna 210 </a>
    </p>
  </body>
</html>
```

Link Example (Cont'd)

Link Targets and Images (Cont'd)

- If the target is not at the beginning of the document, the target must be defined.

- Many tags allow a target to be defined 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 as the link, the link can be written as

```
<a href = "#baskets">
   What about baskets?
</a>
```

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

```
<a
  href="myAd.html#baskets”>
  Baskets
</a>
```

- Links can have images:

```
<a href="c210data.html">
   <img src="plane.jpg“/>
   Info on C210
</a>
```

References

- A relative reference identifies the target document relative to the current document's location:

```
<a href=”dir1/dir2/file.html> my link</a>
```

- Here, the target document is **`file.html`** in the directory **`dir2`**, in the directory **`dir1`**, in the current document's directory.

References (Cont'd)

A relative reference can also identify the target document relative to the web server's root directory:

```
<a href="/dir1/dir2/file.html>
    my link
</a>
```

- Here, the target document is `file.html` in the directory `dir2`, in the directory `dir1`, in the web server's root directory.

References (Cont'd)

- An absolute reference specifies a complete uri and may indicate a document on another server:

```
<a href="http://myserver.se/dir1/dir2/file.html>
   my link
</a>
```

- Here, the target document is **`file.html`** in the directory **`dir2`**, in the directory **`dir1`**, in the root directory of the server **`myserver.se`**.

References (Cont'd)

It is best to use references relative to the current document's directory. Only then is it possible to move the web site without editing all links.

HTML

LISTS

Lists

Unordered lists

- The list is the content of the **<ul>** tag
- A List element is the content of a **<li>** tag

Ordered lists

- The list is the content of the **<ol>** tag
- A List element is the content of a **<li>** tag
- Each item in the display is preceded by a sequence value

```
<h3>
  Some Common Single-Engine Aircraft
</h3>
 <ul>
   <li> Cessna Skyhawk </li>
   <li> Beechcraft Bonanza </li>
   <li> Piper Cherokee </li>
</ul>
```

Lists (Cont'd)

Nested lists

- Any type of list can be nested inside any type of list.
- The nested list must be in a list item of the outer list.

Lists (Cont'd)

Definition list

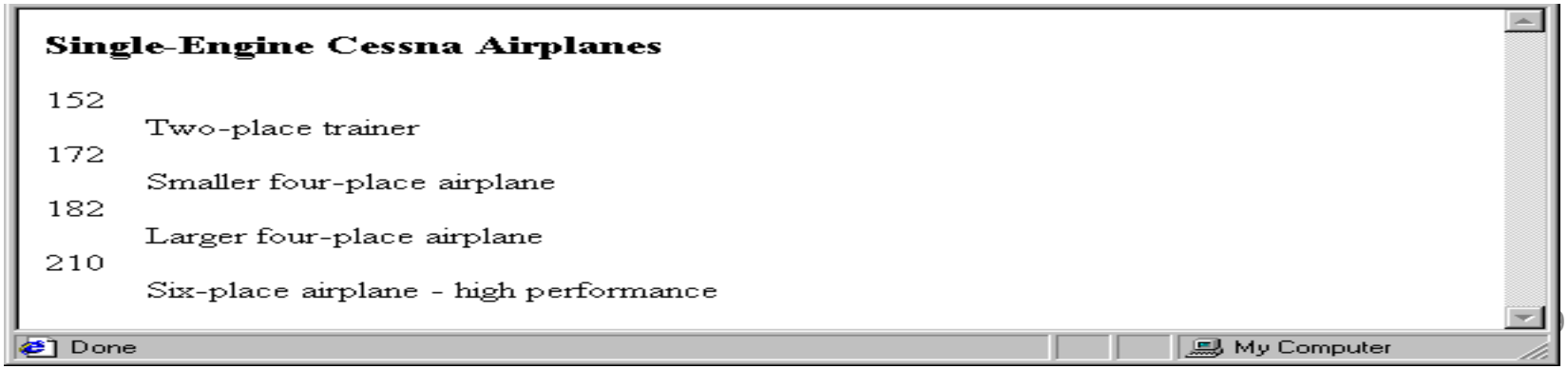
is a list of terms/names, with a description of each term/name.

- The list is the content of a **<dl>** element.

- A term is the content of a **<dt>** element.

- A description is the content of a **<dd>** element.

```
<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>
```



HTML

TABLES

Tables

- A table is a matrix of cells, each possibly having content.

- The cells can include almost any element.

- A table is specified as the content of a **<table>** tag.

- In HTML5, tables do not have borders between the rows or columns. CSS is used for this.

- Tables are given titles with the **<caption>** tag.

Tables (Cont'd)

- Each row of a table is specified as the content of a `<tr>` element.

- Row headings are specified as the content of `<th>` elements.

- Contents of a data cells are specified as the content of `<td>` elements.

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

Table Example

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

Fruit Juice

	Apple	Orange
Breakfast	0	1
Lunch	1	0

Tables (Cont'd)

- A table can have more levels of column labels

- The **`colspan`** attribute of the **`<th>`** tag specifies the number of spanned columns.

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

	Fruit Juice	
	Apple	Orange
Breakfast	0	1
Lunch	1	0

Tables (Cont'd)

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

	Fruit Juice	
	Apple	Orange
Breakfast	0	1
Lunch	1	0

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

Tables Sections

- The table sections are header, body, and footer, defined wiht the elements **`thead`**, **`tbody`**, and **`tfoot.`**
- Table rows in a **`<thead>`** element are rendered at the top of table, rows in a **`<tfoot>`** element are rendered at the bottom.

DO NOT Use Tables for Layout

- In the past, tables were used for page layout, that is deprecated.
- Use only CSS for layout
- Use tables only when the information is naturally tabular

Question 3

HTML

FORMS

Forms

- A form is the usual way information is gotten from a browser user to a server.

- HTML has tags to create a many different elements for this information gathering

- These elements are called widgets or controls or components. (e.g., radio buttons and checkboxes)

- When the Submit button of a form is clicked, the form's values are sent to the server for processing

Forms (Cont'd)

- All 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 called when the Submit button is clicked.

```
<form action="handle-input.php">
```

- The **method** attribute of **<form>** specifies which of the two possible HTTP methods to use for transferring form data to the server, **get** or **post.**
  - The default is **get** .

Components

- Many are created with the **`<input>`** tag
- The **`type`** attribute of **`<input>`** specifies which component to create.

Components (Cont'd)

1. Text

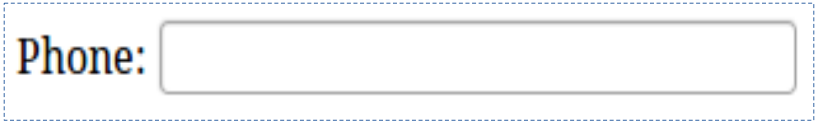
- A horizontal box for text input.
- The **`name`** attribute is used to reference elements in a JavaScript, or to reference form data after a form is submitted.
- Note: Only form elements with a name attribute will have their values passed when submitting.

```
<input type="text" name="phone“/>
```

Components (Cont'd)

- The **<label>** tag is used to define labels for components.
- The **id** attribute of the input element must match the **for** attribute of the **label** element.

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



Components (Cont'd)

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 component's value in the form data when the checkbox is ‘checked’.

- To initialize a checkbox to ‘checked’, the **`checked`** attribute must be set to **`checked`**.

Components (Cont'd)

4. **Radio Buttons**

- Collection 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.

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

Radio Buttons (Cont'd)

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

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

Components (Cont'd)

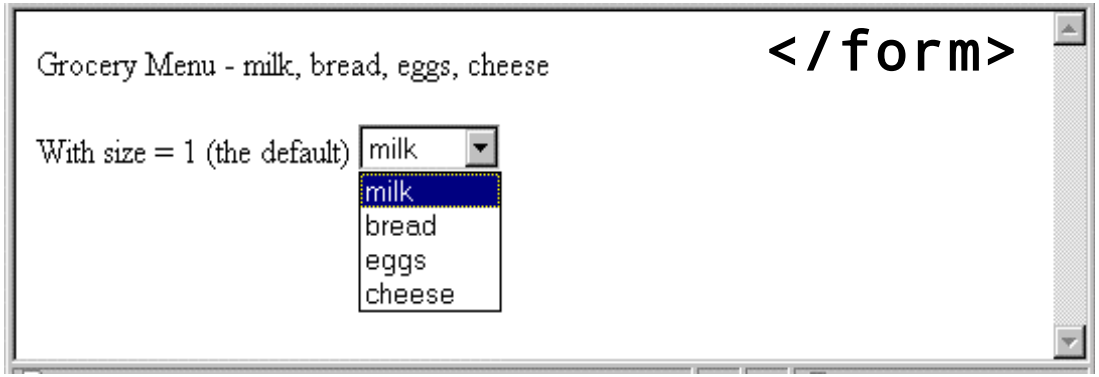
5. The `<select>` tag

- 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)

<select> (Cont'd)

- Each item of a menu is specified with an **<option>** tag, whose text content 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 = "">
    With size = 1 (the default)
    <select name = "groceries">
      <option> milk </option>
      <option> bread </option>
      <option> eggs </option>
      <option> cheese </option>
    </select>
</form>
```



`<select>` (Cont'd)

- After clicking the menu:

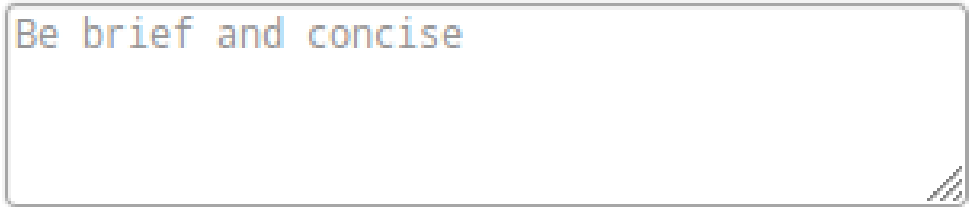
- After changing `size` to 2:

Components (Cont'd)

6. Text area - created with **<textarea>**

- Usually includes the **rows** and **cols** attributes to specify the size of the text area.
- The **placeholder** attribute specifies a text that disappears when the user types.

```
Please provide your
  employment aspirations

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



Components (Cont'd)

7. Reset and Submit buttons

- Both are created with **<input>**

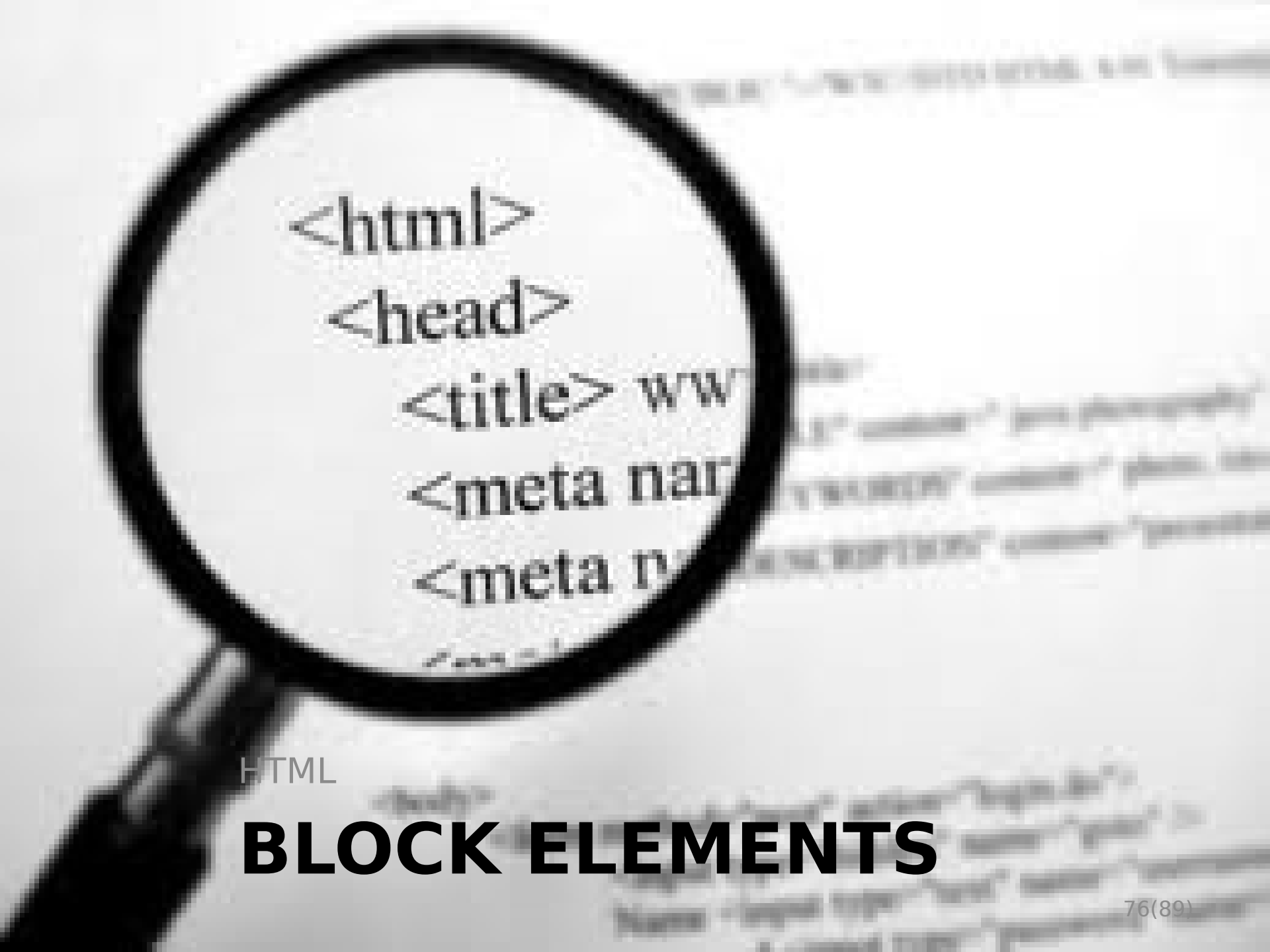
```
<input type = "reset"
       value = "Reset
               Form"
/>

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

- Submit performs two actions:

1. *Encodes the data of the form*
2. Sends HTTP request to the server *specified in the value of the* ***action*** *attribute of* ***<form>*** *tag.*

- A Submit button is required in every form.

- Reset empties all components in the form and does not send any HTTP request.



HTML

BLOCK ELEMENTS

Block and Inline Elements

- A block element takes up the full width available, and has a line break before and after it.
  Examples: **<h1>**, **<p>**, **<ul>**, **<table>**
- Page layout is managed by positioning block elements using CSS.
- A ** ** is a block element that has no other purpose than to define a block. It is often used for layout.
- The 'opposite' of block element is inline element, which does not force line breaks and occupies only the necessary width.
  Examples: **<td>**, **<a>**, **<img>**
- A **** is an inline element that has no other purpose than to define the element. It is also often used for layout.

Question 4

HTML5

HTML5

- All HTML code in this presentation is valid HTML5 (using XHTML syntax).
- However, HTML5 adds lots of new features, like new elements, new attributes, extended CSS support, video and audio players, graphics, local SQL database, full duplex communication, and more.
- Browser support increases steadily, but it is necessary to check that all required browsers support the HTML5 feature you wish to use. A good source for this is **`http://caniuse.com`**.

HTML5 Goals (Cont'd)

- HTML5 is not just new features. It also has different goals than previous versions.
- Deliver rich content without need for additional plugins.
- Features should be based on HTML, CSS, and JavaScript
- Designed to be cross-platform and work for example on PCs, tablets, smartphones, and smart TVs.
- Error handling should be easier than in previous versions.

HTML5 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

HTML5 Audio Element (Cont'd)

General syntax:

```
<audio controls>
  <source src="horse.mp3"
          type="audio/mpeg">
  <source src="horse.ogg"
          type="audio/ogg">
  Your browser does not support
  this audio format.
</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 creates a start/stop button, a clock, a progress slider, total time of the file, and a volume slider

HTML5 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 an MPEG-4 container
  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 specifies that the video should play as soon as it is ready
- The **`preload`** attribute specifies that the video should be loaded as soon as the document is loaded
-The **`controls`** attribute is like that of the **`audio`** element

HTML5 Video Element

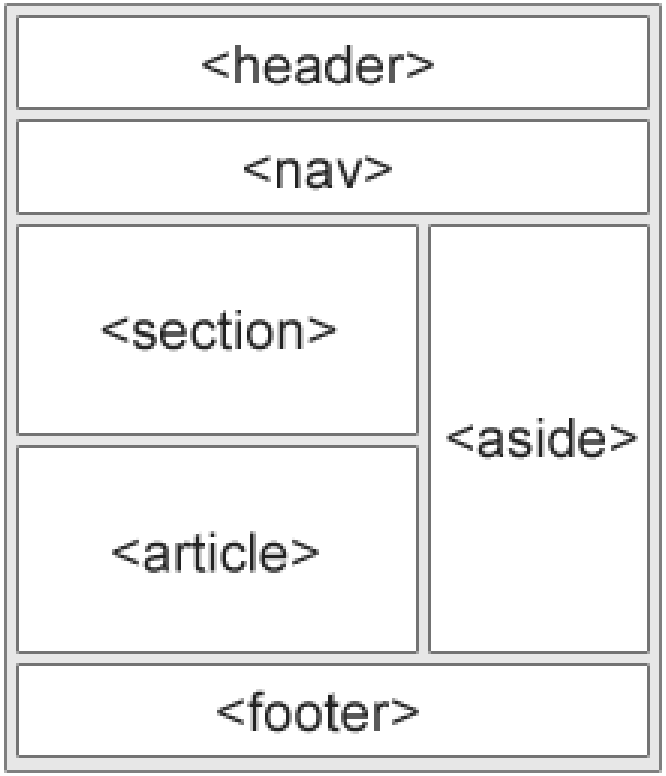
```
<video width="320" height="240"
       controls autoplay>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
  Your browser does not support the video
  tag.
</video>
```

HTML5 Semantic Elements

- A semantic element is an element that clearly describes its meaning, e.g., **<form>**, **<table>**, and **<img>**.
- A typical non-semantic element is ** **, which was previously often used for layout.
- In order to make the code clearer, HTML5 introduces many new semantic elements as alternatives to ** **. Some examples are **<header>**, **<nav>**, **<section>**, **<article>**, **<aside>**, and **<footer>**.

HTML5 Semantic Elements (Cont'd)

- These new semantic elements represent a page layout as depicted to the right.
- Note that these elements do not make the browser place the blocks, that must still be done with CSS. The elements are just names to make the CSS clearer than using `<div>` for all these blocks.



Picture from w3schools.com

Many More HTML Elements

- There are many more new elements and attributes in HTML5 (and also in HTML4).
- **`w3schools.com`** is a good starting point for exploring them.

A Final World of Great Importance

- It is worth to iterate once more:
  HTML should only be used to define parts of a document.
- Layout is coded in CSS, client side behaviour in JavaScript and server side behaviour in for example PHP.
- This separation is necessary to maintain high cohesion.