The Hypertext Markup Language (HTML)

Sivan Toledo
Tel-Aviv University

Text and Markup

Text files are more useful when the text is marked up

```
\title{The Snap-Back Pivoting Method}
\author{Dror Irony and Sivan Toledo}
...
This is \emph{logical markup}.
This is \bold{visual markup}.
```

Logical markup is usually more useful

Applications must understand syntax & meaning

More on Logical vs. Visual Markup

Some markup is clearly logical or visual (title, author, font control)

Usually best to use logical markup and to control the visual presentation using some style-sheet; this allows multiple visual outputs and easy cut & paste

Some markup is hard to classify: footnotes, tables, some mathematical markup

Ad-hoc, SGML, HTML

In the beginning, there were many ad-hoc markup languages (troff, TeX, LaTeX, many others)

SGML: a general method to define markup languages; the syntax is always the same; a single parser can parse any SGML language; very complex

HTML: a specific application of SGML for web pages (text with links and images)

From SGML to XML

HTML is not enough; we need more markup

But SGML is too complex, mainly because it was designed to allow people to easily compose documents in a text editor (examples later)

Insight: programs should construct the actual file format; people edit using a visual editor (think of Word, PowerPoint, Illustrator, formula editors, etc.)

XML

We can simplify parsers by requiring a more regular syntax

This is XML: a way to define markup languages with a specific syntax that is annoying to write by hand but easy to parse (and to generate in a program)

Applications of XML: SVG, MathML, XHTML, OpenOffice file formats, many ad-hoc file formats

More on HTML and XHTML

HTML evolved quickly in the 1990's; the evolution ended in late 1999 with HTML 4.01

XHTML started (1.0) as an XML syntax for HTML 4.01; evolved into a set of standards that allows a web-page file to include in-line objects encoded in other XML languages, such as SVG or MathML

XHTML is still evolving rather quickly

The SGML and XML Document Model

A document is a rooted and ordered tree of nodes
The document has a root
The children of a node are ordered
Nodes can have attributes (key-value pairs)
Attributes are not children of the node
Node types: element, text, comment, cdata (XML only)
The document has a character set (in XML it's always Unicode) and an encoding

Basics

```
<html>                                  open tag
  <head>                                tags are nested
  ...
  </head>                               close tag

  <body dir="ltr">                      tag with attribute
    <p>
      This <strong>course<strong> ...
    </p>
    <hr/>                               empty tag
  </body>
</html>
```

Consistency with XHTML

Lower case for tag names (`html` not `HTML`)

All tags must have an end tag

`<p>...</p>`	`<p>...</p>`	good
`<p>...`	`<p>...`	bad (ok in HTML)
`<hr></hr>`	`<hr/>`	good
`<hr>`	`<hr>`	bad (ok in HTML)

Values in quotes for all attributes

`<... ismap="ismap">`	`<... dir="ltr">`
`<... ismap>`	`<... dir=ltr>`

Core Attributes (for almost all tags)

```
lang="he"
dir="rtl"

id="paragraph821"
class="footnotes"
title="..."

style="font: Arial"

onclick, onblclick, onkeydown, onkeyup,
onkeypress, onmousedown, onmouseup,
onmouseover, onmouseout
```

Core Attributes Cannot be Used on...

Header tags (`html`, `head`, `meta`, `title`, `style`, etc.)

`applet, script, param`

`br, hr`

`frameset, frame, iframe`

In short, on tags that do not represent a container for textual content

HTML Comments

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

No nesting

Can span multiple lines

Can include tags

Must have a space after `<!--` and before `-->`

Comments and Character Data

```
<script> alert("</script>"); </script>
```

This is wrong! The HTML parser will end the script when it finds the end tag within the script
The general principle: text is parsed

To avoid parsing, use comments (in HTML) or character-data sections (in XML)

```
<!-- alert( "</script>" ); -->
<![CDATA ... ]]>
```

Comments/CDATA for Scripts

If a script embedded in HTML or XML might contain markup (not a good idea but very common with old-style scripts), wrap the script in a comment or cdata; make sure the end tag of the comment or cdata does not appear in the script

```
<script>
  // <!--
  alert( "</script>" );
  // -->
</script>
```

The Structure of the File

```
<html>
  <head>
  ...        information about the page; no text
  </head>

  <body ...>
  ...        the content of the page
  </body>
</html>
```

Before the HTML Tag

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

<html>
...
</html>
```

Leave the `html` tag without attributes (can use `lang` and `dir`)

The Head: Title

```
<title>
  The Home Page of Sivan Toledo
</title>
```

Shows up in the title bar of the browser and in bookmarks

Should always be included

Specify `lang` for non-English titles

The Head: Meta

```
<meta
  http-equiv="Content-Type"
  content="text/html;
           charset=iso-8859-1"
>
```

Specifying the charset is very important for non-ASCII pages; `utf-8` is a good choice (supports Unicode; ASCII encoded as is)

```
<meta name="keywords" content="...">
```

The Head: Links and Base URL

<link> tags allow you to specify a relationship to another document, such as a style sheet stored in a separate file

The <base> tag allows you to specify the base address of relative URL's in the document

The Head: Scripts and Style Sheets

```
<script type="text/javascript">
...
</script>

<noscript>
```
What to do if the browser does not support scripts
(show an error message, refresh to another page)
```
</noscript>

<style type="text/css">
...
</style>
```

Scripts are Executed Immediately

So if you want a header script to do something only after the document is fully loaded, register an onload handler and use it to perform the post-load actions

The body: an Overview

Frames can decompose the page into rectangles, for layout (controlling what is displayed where)

Frames (or the entire body if there are not frames) contain flowed material (but with CSS, also positioned elements)

Text: block-level elements and in-line elements

Tables, images, other embedded objects

Block-Level Text Elements

`h1, h2, h3, h4, h5, h6`	Headings
`p`	Paragraphs (`br` for line breaks)
`pre`	Preformatted (preserves line breaks)
`blockquote`	Long quotations
`address`	Address, contact information
`div`	no special meaning; useful grouping for setting attributes (lang, dir) or for CSS control

Inline Text Elements

`em, strong`	emphasis
`dfn, q`	definition, short quotation
`code, var`	computer code, variable
`kbd, samp`	input, output
`sup, sub`	superscript, subscript
`abbr`	abbreviation
`acronym`	acronym
`span`	no special meaning, useful for attributes and CSS control

Lists

`ol`	ordered list (contains li's)
`ul`	unordered list
`li`	list item
`dl`	list of definitions
`dt`	a term to be defined
`dd`	the definition of a term

Table Basics

`table`	
`caption`	the title of the table (if present, must immediately follow `<table>`)
	table if
`tr`	a row
`td`	a cell

Important cell attributes:

`colspan`	multiple columns; "0" is to the end
`rowspan`	multiple rows; "0" is to the end

Advanced Tables: Row Groups

Rows can be grouped into `<thead>`, `<tfoot>`, and one or more `<tbody>`'s

`<thead>` and `<tfoot>` must appear before the `<tbody>`'s

If all are missing, all the rows are implicitly grouped into one `<tbody>`

Row groups help the browser split long tables etc.

Advanced Tables: Column Groups

(less useful)

```
<colgroup span="3" />
```

or

```
<colgroup>
  <col id="first_name" />
  <col id="last_name"  />
  <col id="grade"      />
</colgroup>
```

Images

```
<img src="Eiffel.png"
     alt="A picture of the Eiffel ..."
/>
```

The <object> tag subsumes <img> in HTML 4; it is more general and more flexible

```
<object type="image/png"
        data="Eiffel.png"
/>
```

Alternates in <object>

Whatever is embedded in an <object> tag is considered an alternate rendering, in case the browser cannot or should not render the object

```
<object type="application/mpeg"
        data="Eiffel.mpg">
  <object type="image/png"
          data="Eiffel.png">
    A picture of the Eiffel Tower.
  </object>
</object>
```

Inline Data in <object>

```
<OBJECT type="image/gif"
        data=".."
/>
```

Does not work in IE 6 and lower (works in Mozilla, will work in IE7)

Will be good for JavaScript-generated objects, and for small objects (e.g. icons)

Links

```
<a href="ex2.pdf">Exercise 2</a> is due
next Tuesday.

<h2 id="section5">Syntax Error</h2>
...
Syntax errors were described in <a
href="#section5">Section 5</a>.
```

Image Maps

```
<OBJECT data="navbar1.gif"
        type="image/gif"
        usemap="#map1" />

<MAP name="map1">
  <AREA href="guide.html"
        alt="Access Guide"
        shape="rect"
        coords="0,0,118,28">
  ...
</MAP>
```

Can also respond to clicks on images with JS events

Forms: An Example

```
<FORM action="http://xyz.com/ adduser"
      method="post">
  <P>
    <LABEL for="first">
      First name:
    </LABEL>
    <INPUT type="text" id="first">
    <INPUT type="submit" value="Add">
    <INPUT type="reset">
  </P>
 </FORM>
```

<input> Types

button	push buttons
reset, submit	buttons with form semantics
checkbox	binary input
radio	one of several options (grouped by the name attribute)
text	single-line text input
password	hidden text
file	file selection
hidden	just to send values to the server
image	special submit button; not useful

Better Buttons with <button>

Can have HTML content

```
<BUTTON name="submit"
        value="submit"
        type="submit">
    <IMG src="send.gif"
         alt="send" />
    Send
</BUTTON>
```

Types: submit, reset, button

List Boxes and/or Combo Boxes

```
<SELECT multiple size="4" name="lbox">
  <OPTION selected> Fish  </OPTION>
  <OPTION selected> Chips </OPTION>
  <OPTION> Rice           </OPTION>
  <OPTION> Baked Potato   </OPTION>
  <OPTION> Salad          </OPTION>
  <OPTION> Pasta          </OPTION>
  <OPTION> Ice Cream      </OPTION>
 </SELECT>
```

In this case we allow multiple sections, two options are preselected; we ask the browser to show 4

Hierarchical Combo Boxes

```
<select name="operating-system">
  <option selected >None</option>
  <optgroup label="Linux">
    <option>Red Hat</option>
    <option>Debian </option>
  </optgroup>
  <optgroup label="Windows">
    <option>2000   </option>
    <option>XP     </option>
  </optgroup>
</select>
```

Multi-Line Text

```
<textarea name="thetext"
          rows="20"
          cols="80">
First line of initial text.
Second line of initial text.
</textarea>
```

Grouping Controls in Forms

Group related controls in a `<fieldset>` and give it a title with the `<legend>` element

(good for long forms, especially for non-visual browsers)

Form Navigation

Specify a `tabindex` attribute and an `accesskey` attribute to input elements

This allows users to move between fields in a logical order and to activate fields using alt-key combinations

Submitting a Form to the Server

Not too relevant in AJAX systems

(post vs. get, encoding the form, etc.)

Frames

<frameset>s allow you to split the window into several <frame>s (rectangles) that serve as containers for content stored in other files

```
  </head>
  <frameset cols="20%, 80%">
    <frameset rows="100, 200" />
      <frame src="logo.gif"   />
      <frame src="news.html"  />
    </frameset>
    <frame src="main.html"    />
  </frameset>
</html>
```

Before We Move On with Frames...

Frames are not very useful in web pages that use JavaScript and style sheets

<noframes>

```
...
</head>
<frameset cols="20%, 80%">
   ...
   <noframes>
   ... a rendering of the document without frames
   </noframes>
</frameset>
</html>
```

Internal Frames

Frames that are used as inline elements (like images, tables, etc)

```
<iframe src="foo.html"
        width="400" height="500"
        scrolling="auto">
  Your browser does not support frames.
  However, you may visit
  <A href="foo.html">the related
                    document.</A>
</iframe>
```

HTML: That's It!

For more details (and lists of attributes and what they mean), see a book or the HTML standard on the W3C web site

We didn't say anything about controlling the visual aspects: fonts, sizes, margins, backgrounds, colors, alignment, etc.

To control the visual aspects, use CSS
keep your HTML clean