HTML: Hypertext Markup Language

Lecture 8
HTML

- Hypertext Markup Language

Key ideas:

1. Connect documents via (hyper)links
   - Visual point-and-click
   - Distributed, decentralized set of documents

2. Describe content of document, not style
   - Structure with semantics
   - Separation of concerns

Rephrasing these key ideas:

1. Hypertext
2. Markup
Markup: Describing Content

- **WYSIWYG**
  - A paragraph or bulleted list in MS Word
  - **Benefits:**
    - No surprises in final appearance
    - Quick and easy
    - Control: Author can use visual elements to stand in for structural elements

- **WYSIWYM**
  - A paragraph or list in LaTeX
  - **Benefits:**
    - More information in document (visual & semantic)
    - Lack of Control: Author doesn't know how to apply visual elements *properly* for structure
Abstraction vs Representation

To Do List

1. Study for midterm
2. Sleep
Chapter 9

Now that we have the ability to display a catalog containing all our wonderful products, it would be nice to be able to sell them. We will need to cover sessions, models, and adding a button to a view. So let's get started.

Iteration D1: Finding a Cart

...
Evolution of HTML

- HTML (Berners-Lee, early 90's)
- HTML 2.0 (W3C, ’95)
- HTML 3.2 (W3C, ’97)
- HTML 4.0 (W3C, ’97)
  - To form a more perfect union...
- HTML 4.01 (W3C, ’99)
  - To smooth out the edges... big dog for years
- The great schism
  - W3C: XHTML 1.0 ('00), 1.1 ('01), 2.0
  - Everyone else: HTML Forms, WHAT...
- Capitulation ('09): W3C abandons XHTML 2.0
- HTML5 (end of 2014?)
  - One ring to rule them all...
  - (will include XHTML5, but no one seems to care)
Page Validation

- Design-by-contract:
  - Strong ensures, weak requires
  - Be strict in output, permissive in input

- Browsers (taking HTML as input) are permissive
  - "Tag soup" still renders

- Web authors (writing HTML as output) should be as strict as possible
  - But permissive browsers hide errors!

- Solution: use a validator
  - See validator.w3.org
  - HTML5 validation is "experimental", but still useful
  - Checks for syntax problems only
Example

<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Something Short and Sweet</title>
    <meta charset="utf-8" />
  </head>
  <body>
    <p>
      Hello <a href="planet.html">World</a>!
    </p>
    <img src="pic.png" alt="a globe" />
  </body>
</html>
Example

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Something Short and Sweet</title>
    <meta charset="utf-8" />
</head>
<body>
    <p>Hello <a href="planet.html">World</a>!</p>
</body>
</html>
```
<!DOCTYPE html> <html lang="en"> <head> <title>Something Short and Sweet</title> <meta charset="utf-8" /> </head> <body> <p>Hello <a href="planet.html">World</a>! <br /><img src="pic.png" alt="a globe" /></p> </body> </html>
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Something Short and Sweet</title>
    <meta charset="utf-8" />
  </head>
  <body>
    <p>Hello <a href="planet.html">World</a>! <br />
      <img src="pic.png" alt="a globe" />
    </p>
  </body>
</html>
Document Type Declarations

- HTML 5
  ```html
  <!DOCTYPE html>
  </!DOCTYPE>
  ```

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

- XHTML 1.0 Strict
  ```html
  <!DOCTYPE html
  PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
  strict.dtd">
  ```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Something Short and Sweet</title>
    <meta charset="utf-8" />
  </head>
  <body>
    <p>Hello <a href="planet.html">World</a>!
        <br />
    <img src="pic.png" alt="a globe" />
  </p>
</body>
</html>
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Something Short and Sweet</title>
    <meta charset="utf-8"/>
  </head>
  <body>
    <p>Hello <a href="planet.html">World</a>!</p>
  </body>
</html>
Structure: Nesting of Elements

- `html`
  - `head`
    - `title`
    - `meta`
  - `body`
    - `p` (Hello)
    - `a`
    - `br`
    - `img`

- Something Short and Sweet

- World
Attributes: Name/Value Pairs

<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Something Short and Sweet</title>
    <meta charset="utf-8" />
  </head>
  <body>
    <p>Hello <a href="planet.html">World</a>!</p>
    <img src="pic.png" alt="a globe" />
  </body>
</html>
Structure of Example

- `html`
  - `lang: en`

- `head`
  - `meta`
    - `charset: utf-8`
  - `title`
    - Something Short and Sweet

- `body`
  - `p`
    - `Hello`
    - `a`
      - `href: planet.html`
      - ![br]`World`
    - `img`
      - `src: pic.png`
      - `alt: a globe`
HTML Entities

- Familiar problem: Encoding
  - Is `<br />` a tag or (literal) content?
  - Meta-characters (e.g. '<') need to be escaped

- HTML entities represent a literal
  - `&#dddd;`
  - Where `dddd` is the "unicode code point" (as a decimal number)
    - `&###hhhh;`
    - Where `hhhh` is the code point in hex
  - `&name;`
  - Where `name` is from a small set (`lt`, `gt`, `amp`...)

- Examples:
  - `&#60;` `&#x3C;` `&lt;`
  - `&#9829;` `&#x2665;` `&hearts;`
Kinds of Elements

1. Document structure elements
   - Root of tree is always `<html>`
   - Two children: `<head>`, `<body>`

2. Head elements
   - (Meta) information about document

3. Body elements
   1. Block
      - Content that stands alone
      - Starts new line of text
      - May contain other elements (block or inline)
   2. Inline
      - Intimately part of surrounding context
      - Does not interrupt "flow" of text
      - May contain other inline elements
Block vs Inline

- Flow
- Inline
- Blocks
- Body
- Paragraph
- Heading
- Horz Rule
Required Structure for HTML5

- **html**
  - **head**
    - **title**
    - **meta**
      - charset: utf-8
  - **body**

- element
  - attr name: attr value
  - text
Common Head Elements

- `<title>`: required, must be only text
  - May be displayed in window title bar
- `<script>`: client-side code to run
- `<link>`: other documents to use
  - Commonly used for style information
- `<meta>`: information about the information (document)
  - `<meta http-equiv=""..."" content=""..." />` becomes a header field in HTTP response!
    - `<meta http-equiv="Content-Type" content="..." />
    - `<meta http-equiv="Location" content="..." />
    - `<meta http-equiv="Last-modified" content="..." />
    - `<meta name="keywords" content="..." />`
Common Block Elements in Body

- Text
  - Paragraph `<p>`, horizontal rule `<hr>`
  - Headings `<h1>`, `<h2>`, ... `<h6>`
  - Preformatted `<pre>`¸ quotations `<blockquote>`

- Lists
  - Ordered `<ol>`¸ unordered `<ul>`¸ definition `<dl>`
  - Item in list `<li>` ( `<dt>` `<dd>` for definitions)

- Table `<table>`

- Form `<form>` (and some form elements)

- Sectioning (HTML 5)
  - Article `<article>`, section `<section>`
  - Header `<header>`, footer `<footer>`
  - Canvas `<canvas>`

- Generic container for flow content `<div>`
Common Inline Elements

- Anchor `<a>`
- Phrasing and text
  - Emphasis `<em>`, strong emphasis `<strong>`
  - Code snippet `<code>`
  - Inline quotation `<q>`
  - Inserted text `<ins>`, deleted text `<del>`
- Image `<img>`
- Form elements
- Generic container within flow content `<span>`
- Visual markup: deprecated
  - Bold `<b>`, italic `<i>`, underline `<u>`
  - Typewriter font `<tt>`
  - Font control `<font>`
And Don't Forget Comments

- Comments set off by `<!-- ... -->`
- Beware: they do not nest
Tables

- Row <tr>
- Cell of data <td>
- Header cell (for row or column) <th>
- Caption <caption>
- And some more exotic ones too
  - Header (repeat if splitting) <thead>
  - Body <tbody>
  - Footer (repeat if splitting) <tfoot>
Table Example

<table>
  <caption>Important Dates in CSE 3901</caption>
  <tr>
    <th scope="col">Quiz</th>
    <th scope="col">Day, time</th>
  </tr>
  <tr>
    <td>Midterm 1</td>
    <td>Friday, Sept 21, in class</td>
  </tr>
  <tr>
    <td>Midterm 2</td>
    <td>Monday, Oct 22, in class</td>
  </tr>
  <tr>
    <td>Final</td>
    <td>Wednesday Dec 12, 12:00–1:45</td>
  </tr>
</table>
Important Dates in CSE 3901

Quiz
Midterm 1 Friday, Sept 21, in class
Midterm 2 Monday, Oct 22, in class
Final Wednesday Dec 12, 12:00–1:45
Forms

- Mechanism for generating HTTP requests!
- User Input `<input type="x">`
  - Text fields `<input type="text">`...
  - Radio buttons `<input type="radio">`...
  - checkboxes `<input type="checkbox">`...
  - Submit button `<input type="submit">`...
- Select lists `<select>` and `<option>`
- Text areas `<textarea>`
- Details later
Summary

- Evolution of HTML: HTML 5
  - Tension between permissive and strict
  - Page validation
- An HTML document is a tree
  - Elements are nodes, text is leaves
  - Elements have attributes
- Head elements: meta information
- Body elements: content
  - Block elements
  - Inline elements
- Tables and Forms