# LBSC 690: Information Technology
# Lecture 04
# Separating content, structure, and style

William Webber
CIS, University of Maryland

Spring semester, 2012

# Content, structure, style

**Content** The (textual, graphical) content to be delivered to the user

**Structure** The way the content is laid out on the page, including framing material (navigation, banners, headers, footers)

**Style** The aesthetic presentation of the page (colours, fonts, spacing, borders)

# All in HTML

- Content, structure, and style can all be done in static html (a single, hand-written HTML file):
  - Built-in styled `<hX>`, `<b>` tags; inline `<font>`, `<color>` tags; `style` attributes; etc. for styling.
  - Use of `<p>`, `<div>`, and especially `<table>` tags for layout of page.
  - Place content directly into body of HTML file

# Example: Blandsville Public Library

```
<body>

<p>Blandsville library</p>

<ul>
  <li>Home</li>
  <li>Our history</li>
  <li>Online catalogue</li>
  <li>Upcoming events</li>
</ul>

<p>Welcome to Blandsville
public library!</p>

<p>We have a collection of
7 books, plus the weekly
Blandsville Gazette.</p>

<p>Last updated:
Feb 19, 2012.</p>
</body>
```

Blandsville library

- Home
- Our history
- Online catalogue
- Upcoming events

Welcome to Blandsville public library!

We have a collection of 7 books, plus the weekly Blandsville Gazette.

Last updated: Feb 19, 2012.

# Inline

```
<body bgcolor="AliceBlue">
<h1 style="color: red">Blandsville
library</h1>

<table><tr><td width="80%">
<p font="Arial, Helvetica">Welcome
to <b>Blandsville
public library!</b></p>

<p font="Arial, Helvetica">We
have a collection of
7 books, plus the weekly
Blandsville Gazette.</p></td>

<td width="20%">
<ul><li>Home</li><li>Our history</li>
  <li>Online catalogue</li>
  <li>Upcoming events</li></ul>
</td></tr>

<tr><td colspan="2">
<p font="Arial, Helvetica">Last
updated: Feb 19, 2012.</p>
</td></tr></table></body>
```

# Problems with inline: style

**Confusion of responsibilities:**

- With styling done inline, person who edits content also has to maintain styling.
  - If I add an new paragraph, have to remember to make the font “Arial, Helvetica”.
- We’d prefer to keep these responsibilities (editor, designer) separate—even if we’re performing them both ourselves.

**Difficulty of maintenance:**

- If we want to change the style (e.g. change font), we have to (remember to) change every file, possibly every <p> tag.
- Time-consuming, error-prone.

# CSS: separating content from style

CSS (Cascading Style Sheets)

- allow styling information to be given separately from content
- allow one style to be applied across entire site
- give hierarchical control over style (from general case to specific)
- give more general, flexible, consistent styling language
- different CSS for different media (desktop, mobile, printing)

# CSS

```
<head><link rel="stylesheet"
type="text/css"
href="style.css" /></head>
<body>
<h1>Blandsville
library</h1>

<table><tr><td width="80%">
<p>Welcome
to <b>Blandsville
public library!</b></p>

<p>We have a collection of
7 books, plus the weekly
Blandsville Gazette.</p></td>

<td width="20%">
<ul><li>Home</li><li>Our history</li>
  <li>Online catalogue</li>
  <li>Upcoming events</li></ul>
</td></tr>

<tr><td colspan="2">
<p>Last
updated: Feb 19, 2012.</p>
</td></tr></table></body>
```

style.css:

```
body { background: AliceBlue;
       font: Arial,Helvetica,Serif}
h1   { color: red }
```

# Blandsville library

Welcome to **Blandsville public library!**

We have a collection of 7 books, plus the weekly Blandsville Gazette.

- Home
- Our history
- Online catalogue
- Upcoming events

Last updated: Feb 19, 2012.

# External CSS

```
<head>
  <link rel="stylesheet" type="text/css" href="style.css" />
</head>
```

- External style sheet placed in separate file (here, `style.css`)
- Referenced by `<link>` tag in `<head>` tag of HTML page
- Multiple HTML pages can reference the one CSS file

# CSS syntax and semantics

```
body { background: AliceBlue;
       font: Arial,Helvetica,Serif}
h1   { color: red }
```

CSS entry has syntax of:

| selector | | property | | value | | property | | value | | |
|---|---|---|---|---|---|---|---|---|---|---|
| body | { | background | : | AliceBlue | ; | font | : | Arial | ; | ... } |

**Selector** target of rule; can be tag, class, or id (see later)

**Property** name of property, fixed by CSS standard

**Value** value to apply to property for selected item

Note in above we set font property for <body>, not for <p>. The property is inherited by tags contained in <p> unless overriden.

# Logical structure: CSS classes and ids

- Sometimes, different instances of a tag (e.g. different <p> tags) have different logical meaning, and we wish to present differently
- This can be done by assigning class or id to tag:

```
<p class="highlight">Welcome!</p>
<p id="timestamp">Last updated ...</p>
```

- ...then assigning CSS values to these selectors:

```
.highlight { font-style: italic }
p#timestamp { font-family: monospace; color: grey}
```

# CSS: class and id

```
<head><link rel="stylesheet"
type="text/css"
href="style2.css" /></head>
<body>
<h1>Blandsville
library</h1>

<table><tr><td width="80%">
<p class="highlight">Welcome
to <b>Blandsville
public library!</b></p>

<p>...</p>

<td width="20%">
<ul><li>Home</li><li>...</li></ul>
</td></tr>

<tr><td colspan="2">
<p id="timestamp">Last
updated: Feb 19, 2012.</p>
</td></tr></table></body>
</html>
```

style2.css:

```
body { background: AliceBlue;
       font: Arial,Helvetica,Serif}
h1   { color: red }
.highlight { font-style: italic }
p#timestamp { color: Grey;
    font-family: Monospace }
```

**Blandsville library**

*Welcome to **Blandsville public library!***

- Home
- ...

...

Last updated: Feb 19, 2012.

# CSS: divs and spans

(X)HTML defines two basic, naturally unstyled elements.

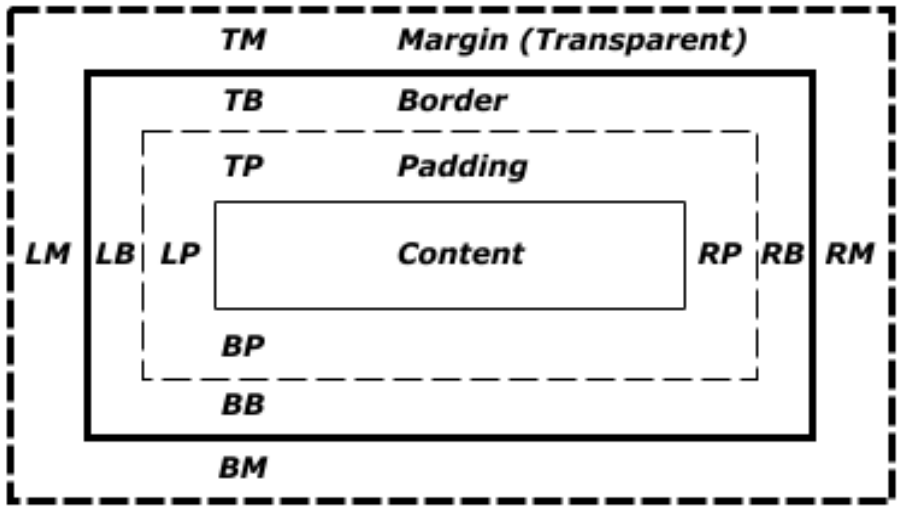
`<span>` an extent of flowing text; may begin, end in mid-line.

`<div>` a rectangular box of content.

- `<span>` can be used instead of `<b>`, `<em>`, etc..

# CSS: the box model

 and other box elements such as <p> have the following components:




- `margin` is outside the border; `padding` is inside
- all three have separately settable `top`, `right`, `bottom`, `left` parts

# CSS: the box model

```
<head><link rel="stylesheet"
type="text/css"
href="style3.css" /></head>
<body>
<h1>Blandsville
library</h1>

<table><tr><td width="80%">
<p class="highlight">Welcome
to <span class="highlight">Blandsville
public library!</span></p>

<p>...</p>

<td width="20%">
<div id="nav">
<ul><li>Home</li><li>Our history</li>
  <li>Online catalogue</li>
  <li>Upcoming events</li></ul></div>
</td></tr>

<tr><td colspan="2">
<p id="timestamp">Last
updated: Feb 19, 2012.</p>
</td></tr></table></body>
</html>
```

style3.css:

```
body { background: AliceBlue;
       font: Arial,Helvetica,Serif}
h1   { color: red }
.highlight { font-style: italic }
.highlight .highlight {
        font-weight: bold }
div#nav { border-style: solid;
          border-width : 1px;
          padding-right: 10px;
          margin-bottom: 5px
}
#timestamp { color: Grey;
    font-family: Monospace }
```

# Structure of the page

- Large site will have multiple pages
- Generally, some structure will be common to each page
- Repeating this structure inline verbatim is time-consuming, difficult to change, error-prone (just like repeating style)

# Repeating structure

- HTML does not natively provide a solution to this (unless very imperfectly with <iframe> tag).
- Some HTML editors (e.g. Dreamweaver) provide a templating facility, which regenerates verbatim HTML code from template, but must regenerate every file when template changes.
- General solution is a programmatic one.

As example, we will examine a *very simple* (please breathe out!) programmatic solution: **server-side includes**.

# Server-side includes (SSI)

- SSI provides very basic programming commands for enhancing an HTML page.
- SSI commands are written as directives embedded inside an HTML page.
- For directives to be processed, the HTML file generally needs to have the extension `.shtml`.
- SSI is not supported on all web servers, nor enabled on all installations (but it is on terpconnect)

# The `#include` directive

- We will look at only one directive:

```
<!--#include file="header.html" -->
```

- When the page is served, this directive will be replaced with the content of `header.html`
- This allows the one fragment of html to be shared by multiple files.

# Templates using SSI

## Listing 1: ssi.shtml

```
<head><link rel="stylesheet"
type="text/css"
href="style3.css" /></head>
<body>

<!--#include file="header.html" -->

<table><tr><td width="80%">
<p class="highlight">Welcome to
<span class="highlight">Blandsville
public library!</span></p>

<p>...</p>

<td width="20%">
<!--#include file="nav.html" -->
</td></tr>

<tr><td colspan="2">
<!--#include file="footer.html" -->
</td></tr></table></body>
```

## Listing 2: header.html

```
<h1>Blandsville library</h1>
```

## Listing 3: nav.html

```
<div id="nav">
<ul><li>Home</li><li>Our history</l
  <li>Online catalogue</li>
  <li>Upcoming events</li></ul></div
```

## Listing 4: footer.html

```
<p id="timestamp">Last
updated: Feb 19, 2012.</p>
```

# A templated web site

## Blandsville library

*Welcome to **Blandsville public library!***

...

- Home
- Our history
- Online catalogue
- Upcoming events

Last updated: Feb 19, 2012.

## Blandsville library

### History

Blandsville public library was established in 2007. ...

- Home
- Our history
- Online catalogue
- Upcoming events

Last updated: Feb 19, 2012.

## Blandsville library

### Catalogue

Title:

Search

- Home
- Our history
- Online catalogue
- Upcoming events

Last updated: Feb 19, 2012.

## Blandsville library

### Events

2012-04-01: Blandsville public library closes permanently.

- Home
- Our history
- Online catalogue
- Upcoming events

Last updated: Feb 19, 2012.

# CSS: separating layout and content

- So far, only used CSS for styling (fonts, colours) and boxes
- Layout is semi-independent of content
  - whether navigation is on left, right, or top, is a presentational issue, independent of what is in navigation
  - 2d layout for desktops may change to 1d for mobile devices
  - navigation, ads may disappear altogether for printing
  - may want to allow browser itself to reflow items based on screen width

# Layout using tables

Traditional layout via `<table>`:

```
<table><tr><td width="80%">
<h2>Events</h2>

<p>2012-04-01: Blandsville public
library closes permanently.</p>

<td width="20%">
<!-- #include file="nav.html" -->
</td></tr>

<tr><td colspan="2">
<!-- #include file="footer.html" -->
</td></tr></table></body>
```

- Entangles layout and content
- Confuses logical and layout sense of table
  - Logical table: rows are items; columns are values (like a spreadsheet)
  - This is an accessibility issue: what does screen reader for visually impaired do?

# Layout using CSS

```
<div id="header">
<!-- #include file="header.html" -->
</div>

<div id="content">
<h2>Events</h2>
<p>...</p>
</div>

<div id="navcont">
<!-- #include file="nav.html" -->
</div>

<div id="footer">
<!-- #include file="footer.html" -->
</div>
```

```
div#header { clear: both }
div#content { width: 60%;
              float: left }
div#navcont{ width: 40%;
             float: right }
div#footer{ clear: both }
```

**Blandsville library**

**Events**

...

- Home
- Our history
- Online catalogue
- Upcoming events

Last updated: Feb 19, 2012.

- NOTE: layout with CSS nice in theory, tricky in practice!

# Accessibility

Making pages accessible to vision impaired (blind; needing large type; color-blind)

- Provide text alternatives to images (e.g. `alt` tag for <`img`>)
- Avoid using tables for layout
- Consider what happens when page is linearized

Accessibility changes also help with:

- Alternate visual formats (mobile, print)
- Web search engines and other online tools

Again: separate content from presentation!