HTML (HyperText Markup Language)

- Current version: HTML5
  - 1991: HTML(1)
    * Designed by Tim-Berner’s Lee at CERN
    * Based on SGML (Standard Generalized Markup Language)
  - HTML 4.01 is most widely used
  - Standardization is both technical and political process

- Basic HTML
  - `<!DOCTYPE ...>`
    * "mostly useless, but required" header to trigger "standards mode" in common browsers in HTML5
    * HTML5
      - `<!DOCTYPE html>`
    * HTML 4.01
      - `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">`
      - `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">`
        - Includes deprecated tags for easier transition
      - The URLs point to DTD (Document Type Definition) file that describes allowed HTML tags and their structure. Remnants of SGML heritage
  - XHTML 1.1
    - `<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">`

- `document = text + tags`
  - Tags are enclosed in `< … >`

- HTML tags
  - Tags are enclosed in `< … >`
  - Tag names are case insensitive, but lowercase is recommended
  - Open tag `<x>` needs a matching closed tag `</x>`
    - Except “empty elements” such as `<br>`, `<hr>`, `<img>` …
    - `<p>` is NOT an empty element, but most browsers do not enforce closing it
  - Tags can have “attributes”
    - E.g., `<img src="…">`
• Both single or double quotes can be used to enclose the attribute value
* Tags represent the “structure” of the document not the “style” or “formatting”
• Use CSS (Cascading Style Sheet) for formatting
  – Most formatting tags like <font>, <center>, <u>, <s>, <tt> are deprecated in HTML5
    * Can use attribute style="font-family:courier" if necessary
    * Use <del>, <ins>, <kbd>, <code>, ..., instead
  – <b> and <i> tags still remain. Just too popular
    * Use <strong> or <em> if that is what we intend
• Q: Why do they try separate structure from style?

  – Text
    * Any text not enclosed in < ... >
      • Q: How do we include < or > in text? What about &?

    * Multiple white spaces and line breaks are merged into one white space
      • Q: How do we include multiple white spaces? Line breaks?

  – Comments appear in <!- ... -->

  – Overall structure

    <!DOCTYPE html>
    <html>
    <head>...</head>
    <body>...</body>
    </html>

Including non-text materials

  – Q: how can we embed a link?
    * <a href=url>...</a>

    * note: <link rel=relationship href=url> does not generate a clickable link
      • e.g., <link rel="stylesheet" type="text/css" href="style.css">
- Q: how can we embed a multimedia object?
  * Image: `<img ...> tag`
  * Audio: `<audio src="voice.mp3" type="audio/mpeg" controls> tag`
    - Controls attribute add control buttons like play, pause, volume, etc
  * Video: `<video src="video.mp4" type="audio/mp4" controls> tag`
  * Others: `<object data=url type=content-type> tag`

  `<object data="http://www.youtube.com/admp.swf?vids=W-5sB6WtFe4&eurl=/index&iurl=http%3A//img.youtube.com/vi/W-5sB6WtFe4/2.jpg&t=OEgsToPDskK05Y3DPYXD_7PQPapoSbvK" type="application/x-shockwave-flash"></object>`

  - `<embed src=url type=content-type>` in also very common (non-standard) due to browser compatibility issues
  * HTML4.01 and below does not have `<audio>` or `<video>` tags

- XHTML
  - Mostly the same thing, but much stricter formatting rules
    * tags and attributes MUST be lower case, not upper case.
    * ALL tags MUST have matching end tags (e.g., `<br></br>`)
    * always use quotes around attribute values
  - Not widely adopted because it is too strict

- What’s new in HTML5
  - New tags, such as `<audio>`, `<video>`, ...
  - Programmable API for Javascript
    * Canvas element for 2D drawing
    * Web Storage for local data storage
    * Offline Web Application for offline app support
    * Document Editing and Drag-and-Drop
    * ...
  - Detailed rules on how to interpret HTML document and handle errors
    * Ensures that different browsers produce the same results in case of error

## HTML Forms
- `<form>`, `<input>`: an intuitive interface to get user's input
  - a `<form>` consists of multiple `<input>`'s

- Google search box example at [http://oak.cs.ucla.edu/cs144/examples/form.html](http://oak.cs.ucla.edu/cs144/examples/form.html)

  ```html
  -- initial request -->
  client << form page --------- server
  -- request w input -->
  <<--- result ---------

  <form action="http://www.google.com/search" method="GET">
    <input name="q" type="text"><input type="submit">
  </form>
  ```
• <input>
  - many "type"s: text, textarea, checkbox, radio, password, file, hidden, submit...
  - name and value pair: q=user_input

• <form>
  - action: the destination of data (or the location of the server process)

• METHOD
  - most common:
    * GET: "retrieve" a resource (no side effect)
      • IMPORTANT: GET should not have any significant side effect at server
      • input values are encoded within URL
        e.g. http://www.google.com/search?q=yahoo
    * POST: "post" data through the specified URL
      • input values are encoded in the body of the request
      • show example packets generated from
        http://oak.cs.ucla.edu/cs144/examples/post.html
        e.g. POST /search HTTP/1.0
            Content-Type: application/x-www-form-urlencoded
            Content-Length: 7
           ...
            q=yahoo
  - less common:
    • HEAD: the same but the header only
    • PUT: "place" the data at the URL (~ replace the data)
    • DELETE: "delete" the resource at the URL
    • OPTIONS: requests for information on available options at the server
    • TRACE: the final recipient returns the whole request message in the response body
      o Q: When will it be useful?

• POST example: http://oak.cs.ucla.edu/cs144/examples/post.html
  - Show POST form example and the HTTP request generated from the form

Cascading Style Sheet (CSS)
• A set of rules for specifying document formatting and presentation
• Rule = selector + declaration block
• CSS can be specified directly inside <style> tag or in a separate page through <link> tag
  o <style type="text/css"> ... </style>
  o <link rel="stylesheet" href="example.css" type="text/css">
• Browsers uses its “default style” to format some tags
  ▪ HTML recommendation: http://www.w3.org/TR/CSS2/sample.html
• Example: http://oak.cs.ucla.edu/cs144/examples/css.html

```css
h1 { /* selector: tag, class or ID name */
    font-size: 30pt; /* declaration block: list of CSS properties and values */
}

h1, p { /* multiple selectors can be separated by commas. * means all tags */
    font-family: "Times New Roman";
}

.note {
    color: blue;
    background-color: yellow;
}

.code { /* Indicates class name */
    font-family: monospace;
    white-space: pre;
    background-color: lightgrey;
    border: 1px solid black;
}

#warning { /* Indicates ID name */
    color: red;
}
```

- Show the body of the page and explain how I want to format it
  - Explain CSS rules on how to interpret them
  - Format “notes” paragraph by adding “notes” class to it
  - Format code using `<div>`
  - Format warning text using `<span>`

• Use `<div>` or `<span>` tags to “mark” a portion of a page for CSS formatting
  o `<span>` for “inline” element (embedded in flowing text)
  o `<div>`: for a “block-level” element (starts a new line and creates a “block”)

• Cascading and Inheritance
  o CSS can be specified in three places. Higher precedence property “cascades”:
    ▪ User’s browser preference
    ▪ Part of Web page
    ▪ Browser default style
  o Unless a CSS property is set by one of the three places (including browser default), a child inherits its parent’s properties.

• Specificity: In case of conflict, most “specific” rule wins!
  o Class vs ID: In a web page, multiple elements with the same class can exist. ID should be unique inside a page
CSS Layout

- CSS can be used to specify page layout

Show CSS layout example at [http://oak.cs.ucla.edu/cs144/examples/css-layout.html](http://oak.cs.ucla.edu/cs144/examples/css-layout.html)

<table>
<thead>
<tr>
<th>header</th>
<th>80px</th>
</tr>
</thead>
<tbody>
<tr>
<td>menu</td>
<td>content</td>
</tr>
<tr>
<td>100px</td>
<td>flexible</td>
</tr>
</tbody>
</table>

- Menu and header stays on screen
- Content area is stretched to fill the screen and scrollable if overflows

- Relevant CSS concepts and properties
  - CSS box model and block-level vs inline elements
  - position: relative, absolute, and fixed

- CSS box model
  - Example: [http://oak.cs.ucla.edu/cs144/examples/css-box.html](http://oak.cs.ucla.edu/cs144/examples/css-box.html)
  - Block-level vs inline element (“display” property)
    - the example page shows the shape of the default “CSS box”
    - block: a separate rectangular “block” is created for the element
      - e.g., <div>, <ul>, <p>…
      - the block starts at a new line, stretches all the way horizontally and uses minimal necessary space vertically
    - inline: the element is displayed as part of normal flowing text
      - e.g., <span>, <a>, …
    - default element type can be overridden through “display” CSS property
      - e.g., display: block;

- CSS box model
  - The “space” around the box can be specified using CSS properties
Add the following properties to `.block-box` class and `.inline-box` class one line at a time and explain what happens:

```css
margin: 1em 2em 3em 4em
padding: 1em;
width: 60%;
height: 300px; /* height: 100% */
```

- Text area size is determined by width and height for block element. For inline elements, width and height values are ignored.
- For “padding:” or “margin:” specification, if any of the four values are missing, missing values are set to be “symmetric”.

- position property:
  - Using position property, and top, right, bottom, and left properties, we can “position” a CSS box of an element
    - Example: [http://oak.cs.ucla.edu/cs144/examples/css-position.html](http://oak.cs.ucla.edu/cs144/examples/css-position.html)
  - Four position values:
    - relative
      - positioned relative to is normal position
    - absolute
      - positioned relative to its nearest *positioned* ancestor
    - fixed
      - positioned relative to the window “viewport”
    - static
      - default value. Element is “unpositioned”
  - actual position is specified through “top”, “bottom”, “left”, “right” properties
    - Add the following to `.box` class and explain
      ```css
      position: relative; /* try fixed and absolute */
      left: 5em;
      top: 10em;
      ```

- Q: How can we specify the layout of [http://oak.cs.ucla.edu/cs144/examples/css-layout.html](http://oak.cs.ucla.edu/cs144/examples/css-layout.html)?

```
+-----------------------------+     +-----------------------------+
| header                     |     | content                     |
| +---------------------------+     | +---------------------------+ |
| menu                       | 80px| flexible                   |
| 100px                      |     | flexible                   |
```

- Note:
• White space due to default margins of body, ul, etc.
• Unless an element is positioned as absolute or fixed, percent height does not stretch.
• Use calc(100% - 100px) to “calculate” length. Space needed around the operator.
  o Overflow property: dealing with overflow text
    ▪ Visible (default): show overflow text
    ▪ Hidden: “clip” overflow text
    ▪ Scroll: always show scrollbar
    ▪ Auto: show scrollbar only if overflow

• Float property: “wrap” text around the box
  o E.g., float: left
  o Instead of position property, use “float: left;” to the .box class rule and show what happens.

• Overlapping elements and z-index
  o Z-index property specifies vertical location if elements overlap.
  o Higher z-index elements is placed on top of lower z-index elements.