CS144 Notes: Web Standards

Basic interaction

- **Example:** http://www.youtube.com
  - Q: what is going on behind the scene?
    - Q: What entities are involved in this interaction?
    - Q: What is the role of each entity?
      - Q: What runs on server? client? network?
        Who keeps track of what is being done?
    - Q: There are many Web servers on the Internet. How can the Web browser reach and communicate with the YouTube server?
    - Q: Many things are exchanged over Internet. Email, instant messaging, file transfer, etc. How does the server know that this "client" wants a "Web page"?
    - Q: Only bytes are transferred. How do they communicate rich, dynamic multimedia content?

- **TCP/IP, http, html**
  - TCP/IP (transmission control protocol and internet protocol)
    - internet routing and transportation protocol
  - http (hypertext transportation protocol)
    - communication protocol between web servers and web clients
  - html (hypertext markup language)
    - page formatting and linking standard

HTTP

- HTTP/1.1 most popular (HTTP/2 is the most recent)
- Request & response

```
        --- request -->
client                  server
        <-- response -
```

- Stateless: every request is handled independently from others
  - Q: what are pros/cons of stateless protocol?

- message = request/status line + header + body

  - http request

    * the bare minimum HTTP request -- can be issued through telnet

      ```
      GET / HTTP/1.0
      ```

    * e.g.

      ```
      GET /cs144/examples/form.html HTTP/1.1    <- request line
      Host: oak.cs.ucla.edu                     <- beginning of header
      User-Agent: Mozilla/5.0 ...
      Referrer: http://oak.cs.ucla.edu/cs144/
      Accept:text/xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
      Accept-Language: en-us,en;q=0.5
      Accept-Encoding: gzip,deflate
      Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
      Keep-Alive: 300
      Connection: keep-alive
      Cookie: __utmz=125574670.1174236576.14.14...  <-- end of header
      ```

  - request line: the actual request
    * more on the "GET" method later

  - header: additional information for the request
    * Host: the name of the web server
      - Q: why do we need the "Host:" field? Aren't we already contacting it?

    * User-Agent: information on the client software
    * Referrer: The page linking to the requested page.
        - Q: how can it be used? Can the server reconstruct the user's click path?

    * Accept ... : what media/content is acceptable to the client q=... specifies how much the type is "preferred"

    * Keep-Alive, Connection: in case we want to make multiple requests through one connection
        - Q: why do we want to make multiple requests per connection?

    * Cookie: more on this later

- http response
    * e.g.

```
HTTP/1.1 200 OK                        <-- status line
Date: Wed, 04 Apr 2007 03:20:33 GMT    <-- beginning of header
Server: Apache
Last-Modified: Wed, 04 Apr 2007 03:19:25 GMT
ETag: "15b63b-af-ebdb0940"
Content-Length: 175
Connection: close
Content-Type: text/html                <-- end of header
            <-- header and body separated by empty line with CRLF
<html>                                 <-- beginng of body
<head><title>Example page</title></head>
 ...
```

- Status line:
    * 2xx: Success - The action was successfully received, understood, and accepted
    * 3xx: Redirection - Further action must be taken in order to complete the request
    * 4xx: Client Error - The request contains bad syntax or cannot be fulfilled
    * 5xx: Server Error - The server failed to fulfill an apparently valid request

- ETag: a unique tag that is the same only if the body is the same
    * Q: when will it be useful?

- Content-Length: length of the body

- Content-Type: the type of the content html, flash, pdf, etc.

- **favicon (favorite icon. small icon next to the URL)**
  - default: /favicon.ico of the site
  - can be customized for every page by adding the following to

```
<head>
<link rel="shortcut icon" href="..." type="image/vnd.microsoft.icon">
```

- HTTP/2
  - HTTP standard approved on Feb 17, 2015
  - Backward-compatible extension to HTTP/1.1
    * Data compression of HTTP headers
    * Parallel loading of page elements over a single TCP connection
    * Server push technologies

HTML (HyperText Markup Language)

- Current version: HTML5
  - 1991: HTML(1)
    * Designed by Tim-Berner's Lee at CERNS
    * Based on SGML (Standard Generalized Markup Language)
  - 1995: HTML2.0, 1997: HTML3.2, 1998: HTML4.01, 2000: XHTML, 2014: HTML5
  - HTML 4.01 is most widely used
  - Standardization is both technical and political process

- Basic HTML
  - 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
, <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

        - <font>, , <u>, <s>, <tt> tags 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 or 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>
```

- <!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">

Q: Where do we specify the text displayed in “title bar”?

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?
    * HTML 4.01
        - Image: `<img src=“url”>` tag
        - Others：`<object ...>` tag
            - standard: `<object data=url type=content-type>`

```
<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
    * HTML5
        - 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 ...>` tag

- XHTML
  - Mostly the same thing, but much more strict formatting rules
    * tags and attributes MUST be lower case, not upper case.
    * ALL tags MUST have matching end tags (e.g.,)
    * always use quotes around attribute values
  - Not widely adopted because it is too strict
- HTML5
  - Add new tags, such as <audio>, <video>, <canvas>, …
  - Specifies scripting API that can be used with 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

Cascading Style Sheet

- Standard language for specifying document formatting and presentation
- A set of rules. Rule = selector + declaration block
- Example: http://oak.cs.ucla.edu/cs144/examples/example.css
- CSS page:

```
p, h2 {          <-- beginning of rule. selector and declaration block
  font-family: "Tahoma", serif;   <-- (property, value) pair
  color: black;
  background-color: white;
}
.note {          <-- for class "note" <h3 class="note">
  color: red;
  border: 1px solid black;
  background-color: yellow;
  font-weight: bold;
}
p.warning {      <-- for <p> element with "warning" class
  color: red;
}
#paragraph1 {    <-- for element with id "paragraph1" <p id="paragraph1>
  margin: 0;
}
h2 p {        <-- for <p> only if it is a descendent of <h2>
  color: red;
}
```

- Web page:

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

- "tag" an element or a set of elements using or
- : within a single element (inline element)
- : around a set of elements. breaks the line at the end (block-level element)
- It is also possible to directly embed CSS as part of HTML document like:

```
<style type="text/css"> @import "example.css"; </style>
```

- Note: In practice, hundreds of bugs even for the most popular browsers
  - CSS filter: exploit CSS bugs to make sure to apply certain CSS rules to only a particular browser family.
    - Any property name starting with * is recognized by IE7 and below, nothing else.

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

```
        -- 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
      - Q: When will it be useful?