SY110
The Web & HTML

Major Brian Hawkins, USMC

U.S. Naval Academy

Fall AY 2018

Outline

1. Review
2. The World Wide Web (WWW)
3. Protocols
4. HTML

Review

Wireless Networking

- What TCP/IP Layer(s)?
- WiFi - 802.11
- hardware
- SSID - Service Set Identifier
- ESSID - Extended Service Set Identifier
- Encryption - WEP, WPA, & WPA2

The World Wide Web

What is it?

- A global collection of clients and servers communicating over the Internet using HTTP (HyperText Transfer Protocol) or HTTPS (encrypted HTTP)
- What is a web server?
  - A computer and the software its running that supply content (usually as HTML) using the HTTP/HTTPS protocols on demand.
- What is a web client?
  - For the purposes of this class - a browser (e.g. Chrome, Firefox, IE, Opera)
- The web != the Internet
  - Why? What's the difference?

The World Wide Web

What is it?

- A global collection of clients and servers communicating over the Internet using HTTP (HyperText Transfer Protocol) or HTTPS (encrypted HTTP)
- What is a web server?
  - A computer and the software its running that supply content (usually as HTML) using the HTTP/HTTPS protocols on demand.
- What is a web client?
  - For the purposes of this class - a browser (e.g. Chrome, Firefox, IE, Opera)
- The web != the Internet
  - Why? What's the difference?

The World Wide Web

What is it?

- A global collection of clients and servers communicating over the Internet using HTTP (HyperText Transfer Protocol) or HTTPS (encrypted HTTP)
- What is a web server?
  - A computer and the software its running that supply content (usually as HTML) using the HTTP/HTTPS protocols on demand.
- What is a web client?
  - For the purposes of this class - a browser (e.g. Chrome, Firefox, IE, Opera)
- The web != the Internet
  - Why? What's the difference?

Browsers and URLs

Browsers

- A Program used for requesting and displaying web pages
- Primary interface – the input/address bar
- Uniform Resource Locators (URLs) are entered into the address bar, which initiates some action on the part of the browser

Examples

- `http://faculty.cs.usna.edu/~bhawkins/courses/sy110/`
- `http` is the protocol.
- `faculty.cs.usna.edu` is the domain name (or server name, for this class).
- `~bhawkins/courses/sy110/` is the path on the server's file system.

Protocols

The file protocol

- The browser can open files on a user's local file system

Example

Imagine we have a file called `vacation.jpg` on our desktop. Then we could open it in our browser by typing:

file:///C:/Users/m21xxxx/Desktop/vacation.jpg

- This is handled locally by your browser – we're not retrieving this file from the web. Note that the server portion of the domain name has been collapsed to nothing, since the file is on our local machine.

Protocols

HTTP - HyperText Transfer Protocol

- A protocol that consists of requests and responses that allow files to be transferred over the Internet
  - Your browser does this for you when you type in a URL, *e.g.* sends an HTTP `GET` request to the web server.

Netkitten example

type:

```
nk rona.academy.usna.edu 80
GET /~sy110/index.html HTTP/1.0
```

Hitting “enter” after the first line, after the second line, and once more.

How does that turn into a web page???

When the browser receives the reply (like we just saw) from the web server, it doesn't display that information – instead, it knows how to interpret the HTML (HyperText Markup Language)

- HTML is text – just like JavaScript . We can create it in an editor
- HTML uses tags (enclosed in angle brackets, < >) to tell the browser how to interpret the text. Most come in pairs, *e.g.* `<b>blah</b>` makes the text **blah** bold.
- Some tags are structural — describe sections of the file (like `<body> ... </body>`), others are for formatting (like `<b> ... </b>`)

HTML Examples

The basic frame:

```
<html>
<head>
</head>
<body>
Hello World!
</body>
</html>
```

Useful HTML Tags and Miscellany

- HTML uses tags to indicate the beginning and end of a particular logical block – the whole file, the body of the file, paragraphs, hyperlinks, etc
    - *e.g.* `<body> ...</body>` or `<p> This is a paragraph </p>`
- Colors – in order to give text a color, we use three bytes of hex characters in RGB (red-green-blue) order in this format
    - `<span style=‘‘color:#00ff00;’’>this will be what color?</span>`
- To create a link to another page or site, use the `<a>` tag
    - *e.g.* `<a href=‘‘www.google.com’’>a link to google</a>`
    - This will create a link that looks like this: a link to google
    - Can this be used for evil purposes? (Yes)
- Special characters can be escaped using their decimal ASCII (or Unicode) values using `&#X`, *e.g.*, < is written as &#60.

Client-server interaction

Sequence of events

Entering the URL `http://faculty.cs.usna.edu/~bhawkins/index.html` in your browser's address bar and hitting “enter” initiates the following:

1. Your browser contacts the server at `http://faculty.cs.usna.edu` and asks for the file `/∼bhawkins/index.html`
2. The server retrieves the file `/∼bhawkins/index.html` and sends it (serves it) to your browser.
3. The browser receives the file and renders it on the screen.

Status Bar and Misdirection

`http://rona.academy.usna.edu/~sy110/lec/wwwIntro/ex1.html`

A day in the life. . .

. . . of a web request

- User types: `http://rona.academy.usna.edu/~sy110/lec/wwwIntro/ex2.html` into the URL bar and presses “Enter”
- The browser sends `rona.academy.usna.edu` an HTTP `GET` request for the file `~sy110/lec/wwwIntro/ex2.html`.
- The server finds this file on its hard drive, and sends it back to the browser.
- The browser receives the file, and sees that additional files will be needed to display the page (*e.g.* `SleepyFace.JPG`).
- The browser issues `GET` requests to `rona.academy.usna.edu` for `SleepyFace.JGP` and `www.foxtrot.com` for `wp-content/uploads/2014/07/ft111002noncompliant.png`
- The browser eventually gets all the files it needs to display the page.

Web server logs

Logging

It's important to remember that we leave many traces when use the web. When we contact a web server asking for a file, the server is almost always going to log (keep track of) what particular file we requested, the time we asked for it, and our IP address. Further, our browser is going to record that we visited the site and keep a copy of the file that we received in a *cache*

Why?

What's the point of keeping a *cache*?

Questions

Questions?