Defining Information Technology & Data Organization

HTML
Learning Goals (lecture + lab)
you should be able to

• use HTML to design networked, hierarchical and tabular structures in webpage content, and use analysis and debugging skills to correct and avoid html errors
Why HTML?

Having completed our overview of what data structures are, and basic computers, we’re ready to peel back the curtain and see what’s behind it.

**HTML (HyperText Markup Language)** is a relatively simple language, and it’s easy to explore and create
Domain names

• http://www.publicaffairs.ubc.ca/2009/09/01/ubc-this-is-your-first-year-class/

• The following are also (higher-level) domain names:
  - publicaffairs.ubc.ca
  - ubc.ca  \((2^{nd}\text{-level domain})\)
  - ca  \((\text{top-level domain, TLD})\)
Servers vs. client

A server is a machine that “serves” content. It’s where the HTML (or other website info) lives. For example, the machine www.ugrad.cs.ubc.ca lives in the ICICS/CS building. This machine is networked with a file server (i.e., a machine that serves up the files), which allows different machines on the ugrad.cs.ubc.ca network to access the same files.

RQ: why web.cs.toronto.edu?
Breaking down the files

After the domain name, comes the file organization. It’s in a hierarchy as well.
Different web browsers interpret this code in different ways

HTML is supposed to be about content, not presentation. People with differing browsers can interpret tags the best way they can. So it was made with tags like `<h1>` which mean “top level header”.

RQ: How has HTML changed since the inception of the internet and how might it change/develop in the future?
Important concepts

• The same appearance / behaviour can be achieved in different ways; the same data can be represented in different ways (key CS concept!)

Recall:
graphical vs textual representation of hierarchical structure (formulae, phylogenetic trees, …)
Important concepts

• **Nesting of constructs** (**key CS concept!**): lists inside of lists, lists inside table, tables inside tables, …

**Think about this:**
What is the connection between nesting of constructs and hierarchical data structures?
Cascading Style Sheets (CSS)

**Cascading Style Sheets (CSS)** are responsible for much of the design of Web pages. It is a general styling system for documents that simplifies the task of creating complex page designs.
Cascading Style Sheets

They allow you to declare some things once and then use it all over the place. E.g., I could do:

```html
<h1 style="color:purple">text here</h1> everywhere. But what if the next person hates purple?
```

Using CSS, you just declare what you want things to look like ONCE, then reuse it many times:
```

h1,h3,h4,h5,h6
{
  color: purple;
}
```
Some HTML tips and tricks

- Use Ctrl-U to view sources of web pages
- Use indentation, line breaks and comment to make your HTML sources more readable
- Prefer semantic markup vs preformatted text
Some HTML tips and tricks

• Type opening and closing tags in pairs, then fill in the middle
  (watch out for missing ‘/’ in closing tags!)

• Write tags first, then add attributes

• Test all hyperlinks immediately
Some general advice

- Look and play with examples found on the web
- Experiment fearlessly
  (web browsers don’t break easily)
- Save your work frequently!
The term "bug" comes from Admiral Grace Hopper:

In 1946, Hopper joined the Harvard Faculty at the Computation Laboratory where she continued her work on the Mark II and Mark III. Operators traced an error in the Mark II to a moth trapped in a relay, coining the term bug. This bug was carefully removed and taped to the log book. Stemming from the first bug, today we call errors or glitch's [sic] in a program a bug.

http://en.wikipedia.org/wiki/Software_bug

Debug your html!

http://validator.w3.org/
Learning Goals [revisited]

you should be able to

• use HTML to design networked, hierarchical and tabular structures in webpage content, and use analysis and debugging skills to correct and avoid HTML errors