Introduction to the Use of Computers

Christophe Rhodes
c.rhodes@gold.ac.uk

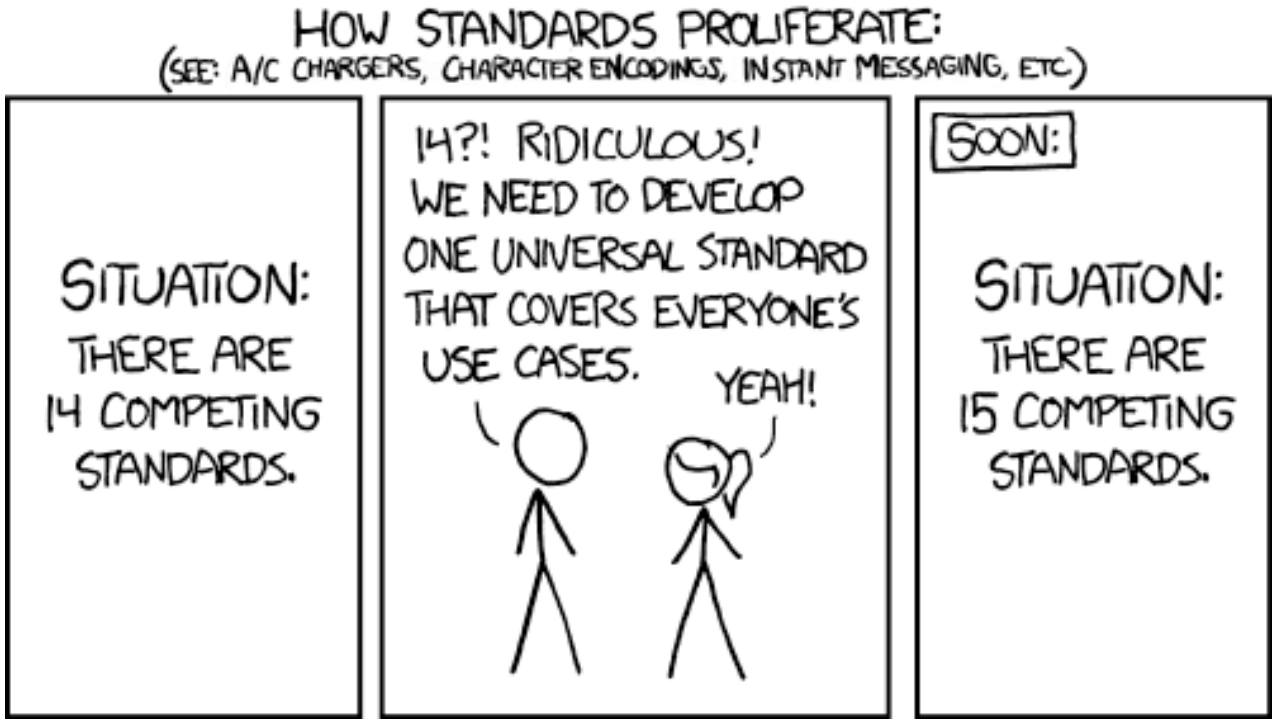
Autumn 2012, Fridays: 10:00–12:00: WTA & 15:00–17:00: WHB 300

HyperText Markup Language

HTML:

- 1990-91: Tim Berners-Lee at CERN
- 1995: HTML 2.0 (RFC 1866)
- 1997: HTML 3.2 (W3C recommendation)
- 1998: HTML 4.0 (W3C recommendation)
- 2000: ISO 15445:2000 (ISO JTC1/SC34)
- 2008-2012: HTML5 (WHATWG Living Standard)
- 2012: HTML5 (W3C Working Draft)

HyperText Markup Language



Randall Monroe, `http://xkcd.org/927`

HyperText Markup Language

Web Browsers

Web Browsers:

- Gecko-based:
  - Firefox
  - Seamonkey
  - conkeror, ...
- Webkit-based:
  - Safari
  - Chrome
  - uzbl, arora, ...
- Internet Explorer
- w3m, lynx, dillo, ...

HyperText Markup Language

HTML Basics

HTML documents have:

- a document type identifier;
- an `html` element.

HyperText Markup Language

HTML Basics

Document type identifiers:

- tell the *processor* how to interpret the document:
- examples:
  - HTML 5: `<!DOCTYPE html>`
  - HTML 4.01 Strict: `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">`

HyperText Markup Language

HTML Basics

`html` element:

- `<html>...</html>`
- what can I put in the '`...`'?

HyperText Markup Language

HTML Basics

`html` element:

- `<html>...</html>`
- what can I put in the '...'?
- what *must* I put in the '...'?
  - exactly one `head` element, followed by
  - exactly one `body` element

HyperText Markup Language

HTML Basics

`head` element:

- `<head>...</head>`
- what can I put in the '`...`'?

HyperText Markup Language

HTML Basics

`head` element:

- `<head>...</head>`
- what can I put in the '`...`'?
- what *must* I put in the '`...`'?
  - exactly one `title` element

HyperText Markup Language

HTML Basics

`head` element:

- `<head>...</head>`
- what can I put in the '`...`'?
- what *must* I put in the '`...`'?
  - exactly one `title` element

`title` element:

- `<title>...</title>`
- what can I put in the '`...`'?

HyperText Markup Language

HTML Basics

`head` element:

- `<head>...</head>`
- what can I put in the '`...`'?
- what *must* I put in the '`...`'?
  - exactly one `title` element

`title` element:

- `<title>...</title>`
- what can I put in the '`...`'?
- what *must* I put in the '`...`'?
  - Human-readable text describing the page;
  - Graphical browsers display the title in the window title bar.

HyperText Markup Language

HTML Basics

Story So Far:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Title Goes Here</title>
  </head>
  <body>
  </body>
</html>
```

HyperText Markup Language

HTML Basics

`body` element:

- `<body>...</body>`
- what can I put in the '`...`'?

HyperText Markup Language

HTML Basics

`body` element:

- `<body>...</body>`
- what can I put in the '`...`'?
- what *must* I put in the '`...`'?
  - paragraphs (`p` element);
  - headers (`h1`, `h2`, ..., `h6` elements);
  - lists (`ul`, `ol`, `dl` elements);
  - tables (`table` element);
  - (*block-level* elements).

HyperText Markup Language

HTML Basics

`p` element:

- `<p>...</p>`
- what can I put in the '`...`'?

HyperText Markup Language

HTML Basics

`p` element:

- `<p>...</p>`
- what can I put in the '`...`'?
  - text!
  - images (`img` element);
  - hyperlinks (`a` element);
  - *inline* elements.

HyperText Markup Language

HTML Basics

Block-level elements:

- create a new container for their own elements;
  - `<h1>Here is a top-level header</h1>`
  - `<p>Here is ordinary paragraph text</p>`
- may contain markup for their own structure:
  - `<ul><li>first item</li><li>second item</li></ul>`

HyperText Markup Language

HTML Basics

Block-level elements:

- create a new container for their own elements;
  - `<h1>Here is a top-level header</h1>`
  - `<p>Here is ordinary paragraph text</p>`
- may contain markup for their own structure:
  - `<ul><li>first item</li><li>second item</li></ul>`
  - `<ol><li>first item</li><li>second item</li></ol>`

HyperText Markup Language

HTML Basics

Block-level elements:

- create a new container for their own elements;
  - `<h1>Here is a top-level header</h1>`
  - `<p>Here is ordinary paragraph text</p>`
- may contain markup for their own structure:
  - `<ul><li>first item</li><li>second item</li></ul>`
  - `<ol><li>first item</li><li>second item</li></ol>`
  - `<table><tr><td>cell1</td><td>cell2</td></tr>`
    `<tr><td>cell3</td><td>cell4</td></tr></table>`
- (usually) may not contain other block-level elements.

HyperText Markup Language

HTML Basics

Inline elements:

- (usually) create a new container for their own elements;
  - `<p>text can be made to <a href="link.html">link</a> elsewhere.</p>`
  - `<p>we can include pictures, e.g. <img src="img.jpeg" alt="alternate"/>.</p>`
- (generally) don't introduce new structure;
- (often) imply certain meaning for their contents.

Inline elements:

- descriptive:
  - `abbr`, `acronym`
  - `cite`, `q`
  - `strong`, `em`, `code`/`kbd`/`samp`

HyperText Markup Language

HTML Basics

Inline elements:

- descriptive:
  - `abbr`, `acronym`
  - `cite`, `q`
  - `strong`, `em`, `code`/`kbd`/`samp`
- presentational:
  - `big`, `small`
  - `b`, `i`, `tt`
  - (obsolete, do not use): `s`, `u`

HyperText Markup Language

HTML Basics

Hyperlinks: the a element:

- creates `link` to a target:
  - `<a href="target.html#name">link text</a>`
  - relative URL in `href` attribute;
  - interpreted relative to the base URL of the document.
  - fragment part (after #) finds element with `id` attribute.

HyperText Markup Language

HTML Basics

Hyperlinks: the `a` element:

- creates `link` to a target:
  - `<a href="target.html#name">link text</a>`
  - relative URL in `href` attribute;
  - interpreted relative to the base URL of the document.
  - fragment part (after #) finds element with `id` attribute.

Images: the `img` element:

- includes an image into the document:
  - `<img src="picture.jpeg" alt="alternate text"/>`
  - relative URL in `src` attribute;
  - alternate text in `alt` attribute;
  - (optional) `width` and `height` attributes;
  - no content (*void* element): self-closing tag.

HyperText Markup Language

HTML: Accessibility

HTML markup:

- contains some meaningful information;
- largely surrounds text;
- can have customized presentation by the viewer:
  - printing, presentation (distinct *media*);

HyperText Markup Language

HTML: Accessibility

HTML markup:

- contains some meaningful information;
- largely surrounds text;
- can have customized presentation by the viewer:
  - printing, presentation (distinct *media*);
  - colour blindness;
  - partial sight;
  - total blindness.

HTML authors cannot assume that their readers will have a similar setup to themselves.

Style

- HTML markup defines structure of content;
- What about presentation?

Style:

- formatting instructions;
- applied to an element;

Style

Style examples:

- `text-decoration`, `text-transformation`, `text-align`, `text-justify`
- `color`, `background-color`, `background-image`
- `font-family`, `font-style`, `font-weight`, `font-size`

Style

Style examples:

- `text-decoration`, `text-transformation`, `text-align`, `text-justify`
- `color`, `background-color`, `background-image`
- `font-family`, `font-style`, `font-weight`, `font-size`
- `margin`, `padding`, `border-width`, `border-style`, `border-color`
- `width`, `height`
- `location`

Cascading Style Sheets

CSS:

- method for styling HTML documents;
- multiple standards:
  - 1996: CSS level 1
  - 1998: CSS level 2
  - 2004, 2005, 2007, 2009, 2010, 2011: CSS level 2 revision 1
  - 1999-2012: CSS level 3
  - some ‘level 4’ modules also exist
- uses styles and selectors.

Style

Cascading Style Sheets

Author styles:

- inline styles
  - use `style` attribute on any element
  - `this <a href="target.html" style="color: rgb(255,0,0)">link</a>is red.`
  - don't have an element? Use `span`.

Style

Cascading Style Sheets

Author styles:

- inline styles
  - use `style` attribute on any element
  - `this <a href="target.html" style="color: rgb(255,0,0)">link</a>is red.`
  - don't have an element? Use `span`.
- embedded style sheet
  - use `style` element in the header
  - select elements to affect using selectors
  - `<style>h1 {font-family: sans-serif } a {color: rgb(255,0,0) }</style>`

Style

Cascading Style Sheets

Author styles:

- inline styles
  - use `style` attribute on any element
  - `this <a href="target.html" style="color: rgb(255,0,0)">link</a>is red.`
  - don't have an element? Use `span`.
- embedded style sheet
  - use `style` element in the header
  - select elements to affect using selectors
  - `<style>h1 {font-family: sans-serif } a {color: rgb(255,0,0) }</style>`
- external style sheet
  - use `link` element in the header to include stylesheet
  - `<link rel="stylesheet" media="screen" href="sheet.css"/>`
  - relative URL in `href` attribute.

Style

Cascading Style Sheets

`media` attribute:

- `screen`: colour screens;
- `all`: all devices;
- `braille`: tactile feedback devices;
- `embossed`: paged braille printers;
- `handheld`: small-screen devices;
- `print`: printed pages;
- `projection`: projected presentations;
- `speech`: speech synthesizers;
- `tty`: fixed character grid devices;
- `tv`: televisions (low-resolution, sound available).

Style

Cascading Style Sheets

User styles:

- provide a default style sheet (e.g. to reflect colour choices)
- can override author styles if the `!important` declaration is given (e.g. to provide local customizations for accessibility)

User-agent style:

- Web browser itself provides a default style sheet;
- Different web browsers have different defaults.

Style

Cascading Style Sheets

Selectors:

- type: `a`
- descendant: `h1 a`
- child: `h1 >a`
- sibling: `p + ul`
- pseudo: `p:first-child`, `a:visited`, `a:hover`
- class: `div.warning`, `div[class ="warning"]`
- id: `p#myid`