CS314 2011 30

Web Protocols

- URI, URN, URL
- Internationalisation
- Role of HTML and XML
- HTTP and HTTPS
  - interacting via the Web

UR what?

- URI: Uniform Resource Identifier
  - Uniquely identifies a data entity
  - Obeys a specific syntax
  - *schemeName:specificStuff*
- URN: Uniform Resource Name
  - A URI that only names something
  - Example: *urn:isbn:0-534-38317-3*
- URL: Uniform Resource Locator
  - A URI that points to an actual resource
  - Example: *http://en.wikipedia.org/wiki/URL*

URI syntax

```
URI = scheme ":" hier-part
      [ "?" query ] [ "#" fragment ]
```

- The hierarchical part can start with // and uses / to separate components. There are other reserved characters

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

scheme name — top of hierarchy (note reversal - DNS writes right to left!) — next level — next level

(DNS is case-independent but URI is case-sensitive)

Internationalisation

- The Unicode standard defines character sets for any script
  - variable length character codes, usually encoded in bytes in UTF-8 format
  - 8-bit ASCII is a proper subset of UTF-8
- Internationalising DNS names, URIs and email addresses in UTF-8 is not simple
  - Yet most people in the world have names like Fältström, or write like this 中文 or this हिन्दी
- Web software should support fully internationalised content and interaction

*ML

- In 1969, three IBMers invented GML (Generalised Markup Language)
- In the early 1980s it became SGML (Standard GML)
- Around 1990, Tim Berners-Lee and Robert Cailliau invented HTML (HyperText Markup Language) as an application of SGML
  - HTML is the format for hypertext documents on the Web
- The WorldWide Web Consortium developed XML (eXtensible Markup Language) as a subset of SGML
  - primary format for data sharing in Web-based services
- XHTML (eXtensible HTML) is an XML-conformant redefinition of HTML

*ML parsers

- Strictly speaking, a pure SGML parser can parse HTML, XML or XHTML
- In practice, HTML is written sloppily with proprietary extensions
  - browsers and XML consumers have to be more tolerant than a strict SGML parser
  - different browsers tolerate different deviations from the standards
  - HTML files that don't cite a specific DTD (Document Type Definition) or omit some syntax elements are often tolerated by browsers

SGML DTDs

- SGML text starts with a DTD declaration such as

```
<!DOCTYPE elem1 PUBLIC "fpi" "path">
```

elem1= the first SGML element in the document

fpi = formal public identifier of the DTD

path = where to find the DTD text

- Example

```
<!DOCTYPE html PUBLIC
  "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
```


*ML document format

- A document must obey its declared DTD
- Thus, with the previous DTD the document must start with <html> and end with </html>
  - Internally, all elements must conform to the syntax defined in the DTD
  - The semantics expressed in comments in the DTD need to be coded into whatever software is interpreting the document

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
 <head>
 <title>*ML document format</title>
 </head>
 <body>
 <h1><font color=blue>*ML document format</font></h1>
  <ul>
   <li>A document must obey its declared DTD
   <li>Thus, with the previous DTD the document must start with
   &lt;html&gt; and end with &lt;/html&gt;
    <ul>
     <li>Internally, all elements must conform to the syntax defined in
     the DTD
     <li>The semantics expressed in comments in the DTD need to be
     coded into whatever software is interpreting the document
    </ul>
  </ul>
 </body>
</html>
```


Getting Hyper

- The key property of an HTML document is that it may contain links to other HTML documents
  - These are called hyperlinks because they may jump anywhere on the Internet, and the resulting interlinked documents are known as hypertext
- A link is formally an anchor in HTML terminology:

```
<a href="http://www.google.com">Click for Google</a>
```

a for Anchor

href for Hypertext Reference

HTTP: Hypertext Transfer Protocol

- Used for communication between Web clients ("browsers") and Web servers
  - Principal use is to carry HTML documents identified by a URL
  - Request/response protocol running over TCP (usually port 80)
- Request includes:
  - Method (see below)
  - URI (typically a URL)
  - Request modifiers and optional content
- Response includes:
  - Status or error code
  - Meta-information about content
  - Content (typically an HTML document)

Important HTTP Methods

- GET
  - retrieve the entity indicated by the URI, e.g. an HTML document
- POST
  - post (send) the content in the request message to the server for processing, e.g. filled in forms from an HTML page
- There are various other methods defined, but GET and POST are by far the most important
  - some methods allow for remote content update

Content Encoding in HTTP

- HTTP message format generally resembles email format, and similar methods are used to encode content (international character sets, graphics, etc.)
- Thus, an image represented in an HTML document like this:

```
<img src="CoffeeBreak.jpg">
```

leads to HTTP content headers like this:

```
Content-Length: 49398
Connection: close
Content-Type: image/jpeg
```

followed by encoded JPEG format

Nested content causes repeated HTTP transactions

```
<html><body>
Here's some text and an image in the same
  directory <img src="CoffeeBreak.jpg">.
Here's <a href="http://w3.org">a
  link.</a>
</body></html>
```

- There will be
  - a GET for the initial document,
  - a second automatic GET on the same HTTP connection for the image,
  - a third GET on a new HTTP connection when the user clicks on the link

POST and the Interaction Model

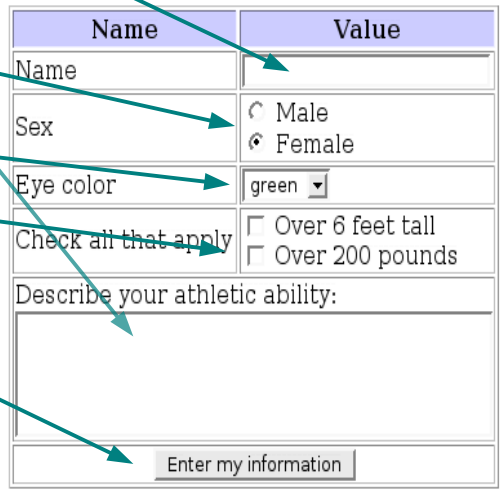
- HTML documents can include *forms,* and browsers support data entry into forms

```
<form action="myScript.cgi"
                  method=POST>
<p>Enter your choice:<br>
 <input type=text name=Choice size=50
value="Anything"></p>
<p><input type=submit value="Send">
</p></form>
```

- Browser generates a POST message when user clicks the Send button, and the data entered is delivered to the script

HTML Form Options

- Possible input fields
  - text (one line of free format text)
  - textbox (more text)
  - radio (somebody thought these look like radio buttons)
  - select (a drop-down list)
  - checkbox (tick mark)
  - submit



Thankyou, Wikipedia!

Scripting (Server Side)

- Data entered in a form is delivered to a script at the server
  - The script must know the fields in the form and their meanings
  - The web server reaches the script through the CGI (Common Gateway Interface)
  - Many choices of scripting language, and ways to use general languages like C and Java
  - Most common today (for complex applications) are Perl, PHP and Ruby on Rails
  - PHP5 is an object-oriented language

Scripting (Client Side)

- HTML content can also include scripts that run on the client machine
  - which, by the way, *could include malicious code*
  - mainly written in JavaScript
- JavaScript is not Java, but can be embedded in HTML documents
  - weakly typed and rather vaguely defined
  - art rather than science
- HTML content can also trigger Java code

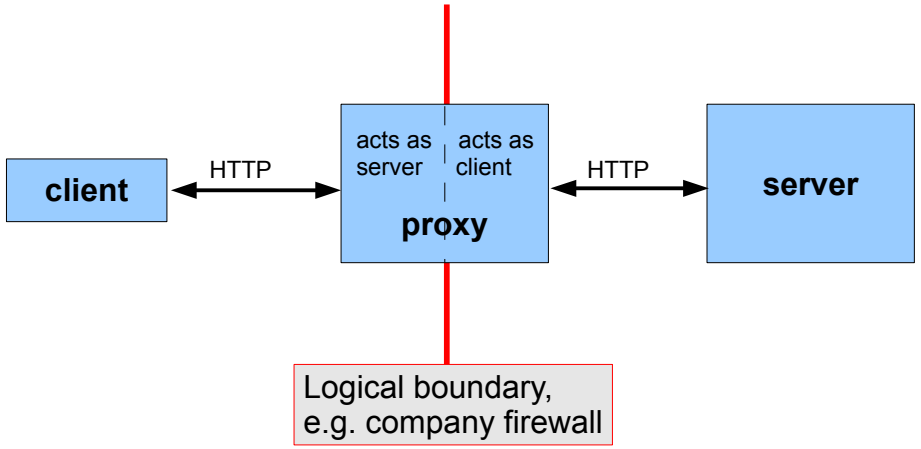
HTTP Caches

- HTTP supports caching
  - locally, in the client system
  - remotely, in an intermediate system
- Caching is obviously beneficial for large images etc.
  - dynamic content must not be cached
  - POST invalidates cached data
- HTTP cache-control directives include:

```
no-cache                #dynamic content
max-age = <seconds>#short-lived data
```


HTTP Proxies

- An intermediate system can proxy HTTP requests and responses
  - a proxy can be configured as caching or non-caching



HTTP security

- An HTTP connection can be opened securely over TLS (SSL) by using the *https:* scheme
  - HTTPS generally listens on port 443
- An insecure HTTP connection can use the HTTP *upgrade* header to upgrade to TLS, even when using port 80
  - the HTTP *CONNECT* method can be used to upgrade to TLS through a proxy
- Beware! Client authentication can be secured via TLS, but HTTP server authentication is a minefield

Congratulations! You have connected securely to StealMyPassword.net.

References

- Shay 12.3 - 12.5
- URIs: RFC 3305, RFC 3986
- Internationalised DNS: RFC 4690
- Internationalised URIs: RFC 3987
- HTML, XML and XHTML:
  http://www.w3.org/TR/1999/REC-html401-19991224/
  http://www.w3.org/TR/xml/
  http://www.w3.org/TR/html/
- HTTP(S): RFC 2616, 2817, 2818
- PHP: http://php.net/
- JavaScript: http://www.openjs.com/