Biocomputing II

Andrew C.R. Martin

SMB
University College London

Biocomputing II

- Web and Perl applications
  - Integrating web pages, Perl
- Good Programming
  - An introduction to good programming practice and large-scale projects

Biocomputing II

Session	Date	Lecture	Practical
1	20/2	Writing your first web pages - HTML and CSS [AM]	Writing your first web pages - HTML and CSS
2	22/2	Writing CGI scripts [AM]	Writing CGI scripts
3	27/2	Web Services and Remote Procedure Calling [AM]	Web Services and Remote Procedure Calling
4	1/3	Collaborative programming and APIs; Introduction to the project [AM/AS]	Project work
5	6/3	Software testing and test suites [AM] UDACITY lecture Notes on unit testing in Perl from James Heald (2013)	Project work
6	8/3	Approaches to debugging [AM] UDACITY lecture	Project work
7	13/3	Project progress discussion [AM/AS]	Project work.
8	15/3	Revision control and bug tracking [AM] Slides from 2016 A self-paced learning activity / tutorial Quick Git Intro Some useful BASH shell scripts that ease some complex Git tasks **Links to more information and tutorials** Source Control HOWTO https://try.github.io http://learngitbranching.js.org/?demo https://www.codecademy.com/learn/learn-git	Project work
9	20/3	An introduction to JavaScript and AJAX [AM]	Project work
10	22/3	Approaches to programming - group discussion [Staff and group members] Tjelvar Olsson's book on Python	Project work

HTML, CSS and XML

Dr. Andrew C.R. Martin
andrew.martin@ucl.ac.uk
http://www.bioinf.org.uk/

Aims and objectives

- Understand the nature of the web and markup languages
- Be able to write web pages in HTML
- Understand CSS styles
- Be able to write a CSS style for an HTML page
- Understand the relationship with XML
- Gain an overview of problems with the web and future directions

XML

Markup languages

- Plain text with ‘markup’
- **Markup**:
  - plain text labels introduced by some ‘special’ character
- For example:
  - **HTML**
  - **XML**
  - **LaTeX**

XML

eXtensible Markup Language

- Simply **marks up the content** to describe its semantics
  - Like a flat-file database
- Invent tags which identify the content

```
<mutants>
  <mutant_group native='1abc01'>
    <structure>
      <method>x-ray</method>
      <resolution>1.8</resolution>
      <rfactor>0.20</rfactor>
    </structure>

    <mutant domid='2bcd01'>
      <structure>
        <method>x-ray</method>
        <resolution>1.8</resolution>
        <rfactor>0.20</rfactor>
      </structure>
      <mutation residue='L24' native='ALA’ subs='ARG' />
    </mutant>

    <mutant domid='3cde01'>
      <structure>
        <method>x-ray</method>
        <resolution>1.8</resolution>
        <rfactor>0.20</rfactor>
      </structure>
      <mutation residue='L24' native='ALA' subs='HIS' />
    </mutant>

  </mutant_group>
</mutants>
```

XML

- XML is **strict**:
  - Case sensitive
  - Every opening tag must have a closing tag
  - Tags must be correctly nested
  - Attributes must always have associated values
  - Attribute values must always be in inverted commas

XML

- XML tells you **nothing about how** data should be presented
- Often used for data that are not for presentation
  - Many programs use XML to store **configuration** data
  - **Large databases** such as InterPro, dbSNP and the PDB are now distributed in XML

Pros and cons

Pros

- Simple format
- Familiarity
- Straightforward to parse
  - parsers available

Cons

- File size bloat
  - though files compress well
- Format perhaps too flexible
- Semantics may vary – what is a 'gene'?

Format flexibility

How to distribute data between tag content and attributes?

```
<mutation residue='L36' native='TRP' subs='ALA' />
```

...compared with...

```
<mutation>
   <residue>L36</residue>
   <native>TRP</native>
   <subs>ALA</subs>
</mutation>
```

DTDs

***D**ata **T**ype **D**efinitions*

- Formal definition of **allowed content**
- Two conflicting standards:
  - **XML-DTD** – original standard; still widely used
  - **XML-Schema** – newer standard; more flexible, is itself XML

XSLT

*e**X**tensible **S**tylesheet **L**anguage **T**ransformations*

- Specialist programming language
  - Code is written in XML!
- Takes XML as input
- Can produce output in
  - XML
  - HTML
  - plain text (e.g. SQL)
- Can be used to convert between XML formats or to generate reports in HTML

XSL

*e**X**tensible **S**tylesheet **L**anguage*

- Full stylesheet language for XML
- Describes how data should be presented

The World Wide Web

- Developed in early 1990s by **Tim Berners-Lee**
  - hyperlinks
  - documents on same or different servers
- Rapid evolution over the last 10-15 years.

Understanding a URL

http://www.biochem.ucl.ac.uk/bsm/index.html

http://www.biochem.ucl.ac.uk/bsm

Assumes this is a file
- finds it isn't and then tries again as a directory

http://www.biochem.ucl.ac.uk/bsm/

Knows that this is a directory

The World Wide Web

- Markup language of the Web is **HTML (HyperText Markup Language)**
- HTML was designed **before** XML

HTML

- Inspired by **SGML**
  - Standard Generalized Markup Language
- SGML, standardized in **1986**
  - ISO:8879, 1986
  - complex standard for document markup
  - used by a few large companies and research organizations

HTML

- Markup consists of '**tags**'.
- **Start tag:** label between angle brackets (< and >)
- Followed by the tag **content**
  - may include nested tags
- A **closing-tag:** (or end-tag) identical to the opening tag, but starts with </ rather than <

Web standards

Original HTML standard was very forgiving:

- allow missing end tags (e.g. for <p>)
- inaccurate nesting
- attributes without values
- attribute values not enclosed in inverted commas.

HTML4.0 (XHTML)

New standard is now **described in XML** and therefore much more strict

- Case sensitive
- Opening tags must have closing tags

```
<p>..foo..</p>
<br />
```

- Tags must be correctly nested

```
<b><i>..foo..</b></i> illegal
```

- Attributes must have values

```
<tr nowrap>          illegal
<tr nowrap='nowrap'>
```

- Attribute values must be in inverted commas

```
<img src=picturef.gif /> illegal
```

HTML

- For example, the following would indicate a piece of text to be set in a bold font:

```
<b>This text is in bold</b>
```

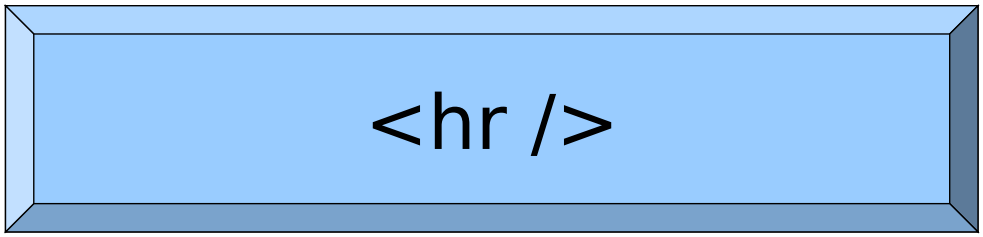
tag *closing-tag*

HTML

- Tags may contain '**attributes**': a type/value pair contained within an opening tag.

HTML

- Some tags do not require an end tag
  - the tag is started with < and should be ended with />



The most useful HTML tags

The most useful HTML tags

<html>

- Encompasses the whole of an HTML document

```
<html>
...my document goes here...
</html>
```

The most useful HTML tags

<head>

- Define things that relate to the whole document, but are not actually displayed.

```
<html>
<head>
...data go here...
</head>
</html>
```

The most useful HTML tags

<title>

- Placed within the <head> tag
- Gives the title to go in the browser window frame

```
<html>
<head>
<title>My document</title>
</head>
</html>
```

The most useful HTML tags

<body>

- All the HTML relating to what appears on the page is contained in this tag.

```
<html>
<head>
<title>My document</title>
</head>
<body>
...content goes here...
</body>
</html>
```

The most useful HTML tags

<h1>, <h2>, <h3>, <h4>, <h5>, <h6>

- Headings and subheadings
- By convention, only one <h1> tag

```
<h1>Page title</h1>
...some html...
<h2>A heading</h2>
...some html...
<h2>Another heading</h2>
...some html...
<h3>A subheading</h3>
...some html...
```

Page title
...some html...
A heading
...some html...
Another heading
...some html...
A subheading
...some html...

The most useful HTML tags

<p>

- Enclose a paragraph of text

```
<p>This is some text which will appear as a paragraph
in the web page.
</p>
```

The most useful HTML tags

<a href='url'>

- Indicates a hyper-link to another page
- URL may be absolute or relative

```
<a href='http://www.bioinf.org.uk/'>My bioinformatics page</a>
<a href='/pages/index.html'>More pages</a>
<a href='page.html'>Another page</a>
```

The most useful HTML tags

- Displays the specified image

```
<img src=’image.gif’ />
<img src=’image.gif’ border=’0’ />
```

The most useful HTML tags

- Forces a line break

<hr />

- Displays a horizontal rule

```
<br />
<hr />
<hr width=’50%’ />
```

The most useful HTML tags

<table>, <tr>, <th>, <td>

- Creates a table

```
<table>
<tr><th>Head1</th><th>Head2</th></tr>
<tr><td>data11</td><td>data12</td></tr>
<tr><td>data21</td><td>data22</td></tr>
</table>
```

Head1 Head2
data11 data12
data21 data22

The most useful HTML tags

<pre>

- 'pre-formatted' text

```
<pre>
This is pre-formatted text:
   very useful for code
int main(int argc, char **argv)
{
   print "hello world\n";
   return(0);
}
</pre>
```

```
This is pre-formatted text:
   very useful for code
int main(int argc, char **argv)
{
   print "hello world\n";
   return(0);
}
```

The most useful HTML tags

<b>

- Text displayed in bold

<i>

- Text displayed in italics

<tt>

- Text displayed in a mono-spaced font

<u>

- Text displayed underlined

The most useful HTML tags
, <ul>,

- Ordered and un-ordered lists

```
<ol>
<li>Ordered list</li>
<li>is numbered</li>
</ol>

<ul>
<li>Un-ordered list</li>
<li>is bulleted</li>
</ul>
```

1. Ordered list
2. Is numbered

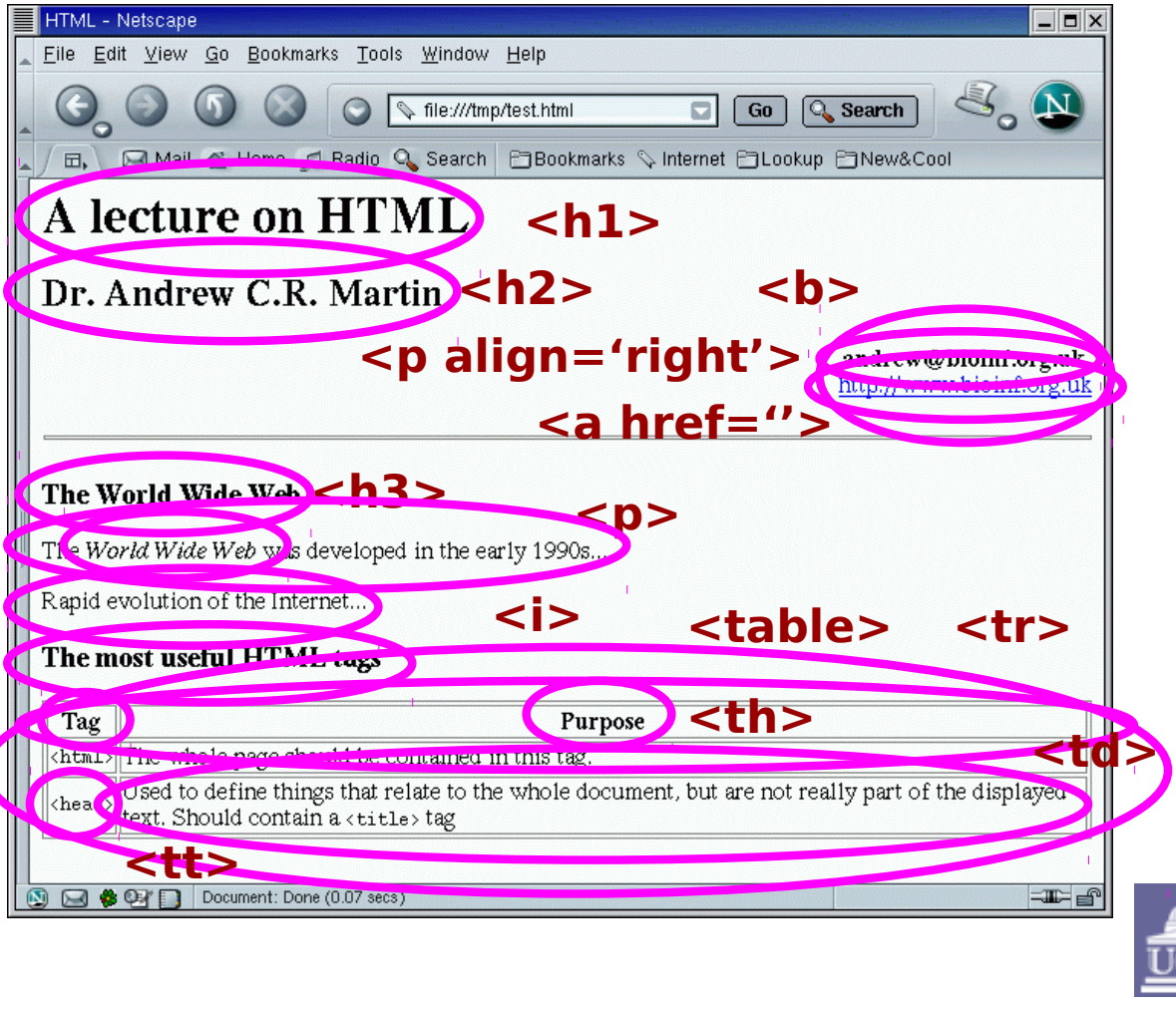
- Un-ordered list
- is bulleted

The most useful HTML tags

<font>

- Specifies the font to be used for the enclosed text

```
<font face='Helvetica,Arial,sans-serif'
      size='14pt' color='green'>
Some text in 14pt green Helvetica!
</font>
```



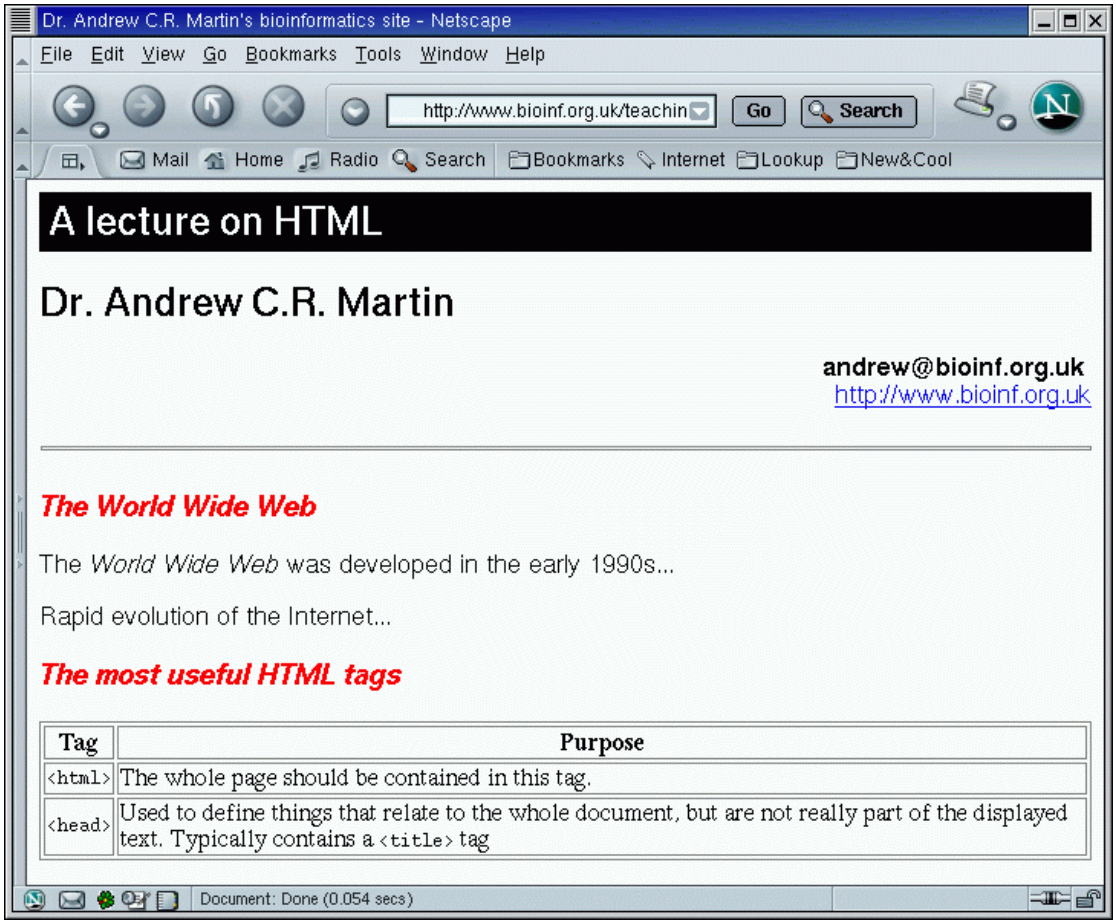
```
<html>
   <head>
      <title>
         HTML
      </title>
   </head>

   <body>
      <h1>A lecture on HTML</h1>
      <h2>Dr. Andrew C.R. Martin</h2>
      <p align='right'>
         <b>andrew@bioinf.org.uk</b> <br />
         <a href='http://www.bioinf.org.uk'>http://www.bioinf.org.uk</a>
      </p>

      <hr />

      <h3>The World Wide Web</h3>
      <p>The <i>World Wide Web</i> was developed in the early 1990s...
      </p>
      <p>Rapid evolution of the Internet...
      </p>

      <h3>The most useful HTML tags</h3>
      <table border='1'>
         <tr><th>Tag</th><th>Purpose</th></tr>
         <tr><td><tt>&lt;html&gt;</tt></td>
             <td>The whole page should be contained in this tag.</td>
         </tr>
         <tr><td><tt>&lt;head&gt;</tt></td>
             <td>Used to define things that relate to the whole document,
                 but are not really part of the displayed text. Should
                 contain a <tt>&lt;title&gt;</tt> tag
             </td>
         </tr>
      </table>
   </body>
</html>
```

Cascading style sheets (CSS)

- Provides additional **separation between content and presentation**
- Avoid display control within the HTML
- Easier to produce **consistent documents**
- Allows font, colour and other rendering styles for tags to be defined once

Cascading style sheets (CSS)

```
h1 { margin: 0em;
     border: none;
     background: black;
     color: white;
     font: bold 18pt Helvetica, Arial, sans-serif;
     padding: 0.25em;
   }
h2 { font: bold 18pt Helvetica, Arial,
  sans-serif;}
h3 { font: bold italic 14pt Helvetica, Arial,
  sans-serif; color: red;}
p { font: 12pt Helvetica, Arial, sans-serif;}
```



Cascading style sheets (CSS)

CSS can be placed within the <head> tag:

```
<head>
<style type='text/css'>
<!--
      ........ CSS GOES HERE ........
-->
</style>
</head>
```

or in a file referenced from the <head> tag:

```
<head>
<link rel='stylesheet' type='text/css'
      href='example1.css' />
</head>
```

Cascading style sheets (CSS)

- Even better! Can create 'classes' offering further control and

```
<p class='author'>
   <span class='email'>andrew@bioinf.org.uk</span>
   <br />
   <a href='http://www.bioinf.org.uk'>
      http://www.bioinf.org.uk</a>
</p>
```

```
.author { text-align: right; }
.email  { font-weight: bold; }
```

Cascading style sheets (CSS)

- CSS is depracating many HTML tags and attributes.
- e.g.
  - <font> tag
  - align='xxxx' attribute
  - color='xxxx' attribute

Web standards

- **Rapid evolution** of the Web has led to many new and evolving standards
- HTML evolution has led to new tags. **Unwieldy** and, in some cases, **browser-specific**
  - Some web pages only render properly only on a given browser

HTML4.0 (XHTML)

- XHTML **segregates core functionality** from additional packages
  - these support more advanced features (tables, forms, vector graphics, multimedia, maths, music and chemistry)
- Modular design useful for **pervasive web computing**
- **Document** provides a **profile** describing browser requirements.
- Browser provides **device profile** describing which tag modules it supports.

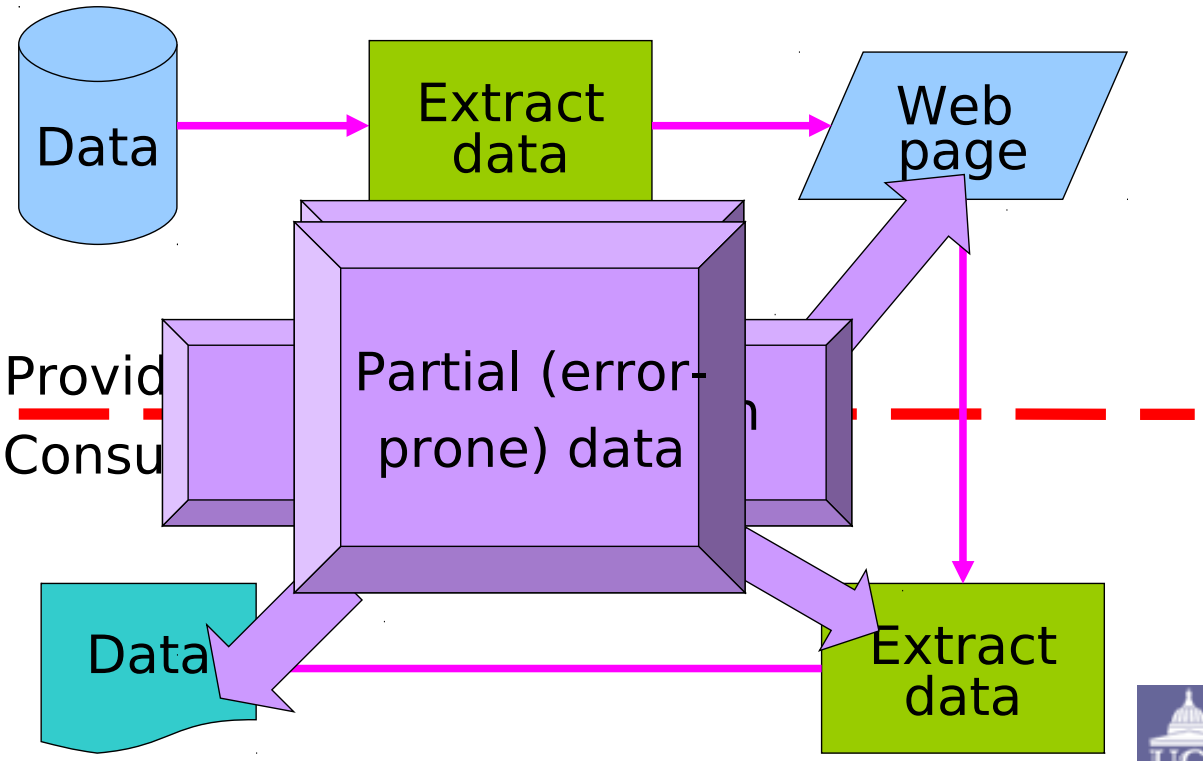
Problems of the web

- HTML is **computer readable**, not computer understandable
- **Separation** between content and presentation is only **minimal**
- Restructuring of data for display results in **information loss**
  - Semantic meaning is lost and perhaps only a subset of available information presented

Problems of the web

- Web page with attractive layout and tables – **powerful visual image**, easy to digest.
- Extraction of data from HTML visual markup results in further **information loss**.
- **If layout changes**, the parser no longer works.

Problems of the web



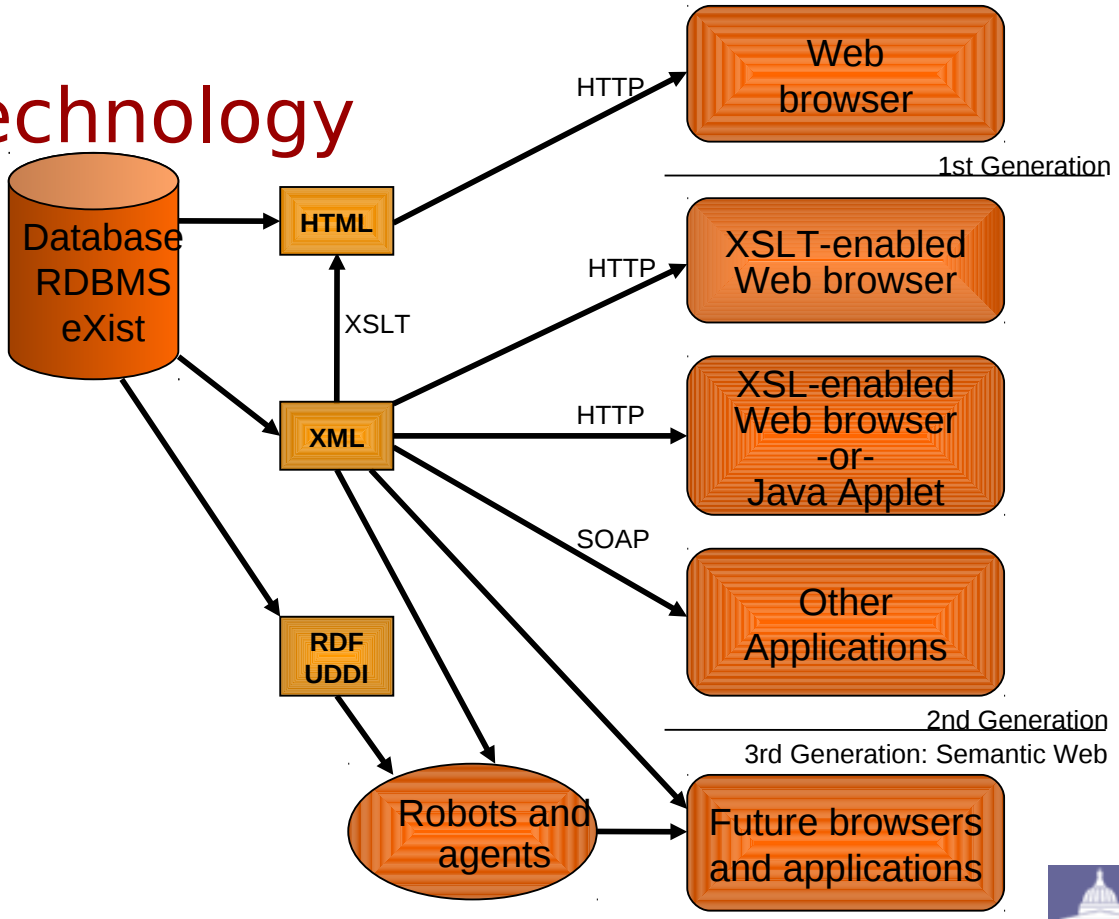
The Semantic Web

- Designed to **address these problems**
- **Semantic markup**
  - the Web will itself become a huge database
- **Software agents** to retrieve relevant results
- Tim Berners-Lee **Scientific American (Berners-Lee, et al., 2001)**

The Semantic Web

- Data **stored and sent in XML** together with a style sheet.
  - Responsible for formatting and display
  - **Key difference**: use of XML supported by ontologies
- Data presented by:
  - direct display of XML using XSL
  - translation to HTML using XSLT followed by formatting with CSS

Technology



Summary

- HTML and XML markup languages
- Problems of visual markup
- Separation of content and presentation
- Semantic web