HTML and XML

Dr. Andrew C.R. Martin, UCL

andrew@bioinf.org.uk

The World Wide Web

The World Wide Web was developed in the early 1990s by Tim Berners-Lee to share text via hyperlinks between documents on the same or different servers.

Rapid evolution of the Internet over the last 10 years.

Web pages contain plain text with 'markup' to indicate how text should be presented.

Markup simply consists of plain text labels introduced by some 'special' character. The current markup language of the Web is HyperText Markup Language (HTML).

HTML

The language of the Web.

Inspired by the ideas of SGML (Standard Generalized Markup Language).

SGML, standardized in 1986 (ISO:8879, 1986), is a complex standard for document markup used mostly by a few large companies and research organizations.

The markup consists of 'tags'.

A **start tag** is a label between angle brackets (< and >).

This is followed by the tag content – some text which may include further nested tags.

An **end-tag** is identical to the opening tag, but starts with </ rather than <.

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

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

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

e.g. the hyperlink consists of an 'anchor' (<a>) tag with which the text displayed as a hyperlink is indicated and an `href` attribute containing the URL to which the link is made.

For example:

```
<a href='http://www.bioinf.org.uk/'>My bioinformatics page</a>
```

Some tags do not require an end tag. In these cases, the tag is started with < and should be ended with a space and /> rather than >

For example <hr /> creates a horizontal rule.

The most useful HTML tags

Almost all tags can take attributes to modify their behaviour.

Tag	*Purpose*
<html>	The whole page should be contained in this tag.
<head>	Used to define things that relate to the whole document, but are not really part of the displayed text. Should contain a <title> tag.
<title>	Placed within the <head> tag to indicate the title to go in the browser window frame.
<body>	All the HTML relating to what appears on the page is contained in this tag.
<h1>, <h2>, <h3> <h4>, <h5>, <h6>	Used to define headings and sub-headings at descending levels. By convention, a document contains only one <h1> tag for the page title. <h2> is then used for section headings and so on.
<p>	Used to enclose a paragraph of text.
<a href='url'>	Indicates a hyper-link to another page.
	Displays the specified image (no closing tag).
 	Forces a line break (no closing tag).
<hr />	Displays a horizontal rule (no closing tag).
<table>	Creates a table. Must contain <tr> tags.
<tr>	Used within a <table> tag to creates a row in a table. Must contain <td> or <th> tags.
<td>, <th>	Used within a <tr> to specify the data to go in a column. <td> is used for normal data, <th> is used for a table header.
<pre>	The contained text is 'pre-formatted'. It is displayed in a mono-spaced font and line-breaks and spacing are all displayed.
<b>	Contained text is displayed in bold.
<i>	Contained text is displayed in italics.
<tt>	Contained text is displayed in a mono-spaced font .
<u>	Contained text is underlined.
	An ordered (numbered) list. Each item in the list is specified with (list item) tags.
	An un-ordered (bulleted) list. Each item in the list is specified with (list item) tags.
	Specifies an item in a list (either or).
<font>	Specifies the font to be used for the enclosed text.
<form>	Used with other tags to create a form for entering data.

A complete example

```
<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>
```

Spacing and indentation is only to show the structure of the document.

Note that you can't place < or > in the displayed text, since these will be interpreted as markup tags. Instead you use `<` for < and `>` for >.

You can view the resulting HTML page at:

`http://www.bioinf.org.uk/teaching/bbk/biocomp2/html_example.html`

Cascading Style Sheets (CSS)

HTML provides only limited clues about how text and images should be displayed. The `<font>` tag can be used to add some more control over presentation.

Cascading Style Sheets (CSS) provides some additional separation between content and presentation by avoiding lots of display control within the HTML. This makes it much easier to produce consistent documents.

CSS allows the font, colour and other rendering styles for elements to be defined once.

The following example describes styles for `<h1>`, `<h2>`, `<h3>` and `<p>` elements:

```
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;}
```

The CSS can be placed within the `<head>` element:

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

or in a separate file which is then referenced from within the `<head>` element:

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

View the same document with this CSS style applied here:

`http://www.bioinf.org.uk/teaching/bbk/biocomp2/css_example.html`

Even better, one can define 'classes' that can be applied selectively to paragraphs or other elements, so that (for example), the author information in the above example isn't right formatted in the HTML, but is assigned to an 'author' paragraph class which the CSS then formats appropriately:

```
<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; }
```

XML

XML is a markup language similar in style to HTML. However it's aim is not to describe how information should be displayed on a screen, but simply to mark up the content of a text file to describe its semantics. It is therefore rather more similar to a flat-file database.

Thus one can invent tags which identify the content of the data. XML is being used more and more to mark up data in Bioinformatics. Here is a fragment of XML that we use to contain information on protein structures containing single site mutations.

```
<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' substitution='ARG' />
      </mutant>
      <mutant domid='3cde01'>
         <structure>
            <method>x-ray</method>
            <resolution>1.8</resolution>
            <rfactor>0.20</rfactor>
         </structure>
         <mutation residue='L24' native='ALA' substitution='HIS' />
      </mutant>
      <mutant domid='4def01'>
         <structure>
            <method>x-ray</method>
            <resolution>1.8</resolution>
            <rfactor>0.20</rfactor>
         </structure>
         <mutation residue='L36' native='TRP' substitution='ALA' />
      </mutant>
   </mutant_group>
</mutants>
```

Note the similarity to HTML. There are tags (e.g. `<mutant_group>`) with matching end-tags (e.g. `</mutant_group>`). Tag/end-tag pairs may contain data or other tags. Tags may also have associated attribute/value pairs (e.g. `<mutant domid='4def01'>`).

However, XML is much stricter than HTML:

- 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

Note from the example above that XML tells you nothing about how the data should be presented. Indeed it is often used for data that are not to be shown on the web. Many programs now use XML to store configuration data and large databases such as InterPro and dbSNP are now distributed in XML.

Pros and cons

The major advantages of XML are:

- Simple format with which people are familiar from the web
- Straightforward to parse – standard parsers available for many programming languages

Major problems are:

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

Format flexibility

Have to decide how to distribute data between tag content and attributes. e.g.

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

...compared with...

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

DTDs

'***D**ata **T**ype **D**efinitions*' – formal definition of the allowed content for an XML file. 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 which takes XML as input and can produce output in XML, HTML, or plain text (e.g. SQL). XSLT is itself written in XML!

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 which describes how data should be presented

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.

Many browsers are very forgiving – allow missing end tags (e.g. for `<p>`); inaccurate nesting (e.g. `<i><b></i></b>` should not be allowed); attributes without values; attribute values not enclosed in inverted commas.

New HTML4.0 (XHTML) standard is now described in XML and therefore much stricter than earlier versions of HTML.

- All tags must be in lower case
- Every opening tag must have a closing tag, or an unclosed tag (e.g. `<br />`) must be used
- Tags must be correctly nested (e.g. `<b><i> ...foo... </b></i>` is illegal)
- Attributes must have values (e.g. `<tr nowrap>` is not allowed, you must do `<tr nowrap='nowrap'>`)
- Attribute values must be in inverted commas (e.g. `<img src=picture.gif />` is not allowed, you must do `<img src='picture.gif' />`)

XHTML also segregates core functionality from additional packages which support more advanced features (tables, forms, vector graphics, multimedia, maths, music and chemistry).

Modular design useful for pervasive web computing.

Document can provide a profile describing the minimum requirements of browser.

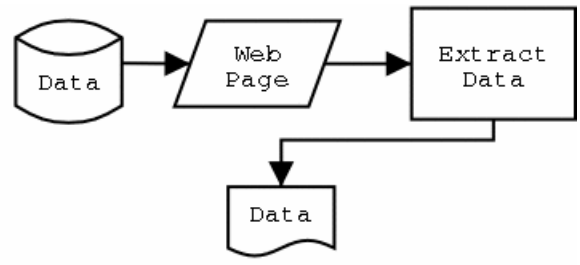
Browser can provide 'device profiles' describing which tag modules it supports.

Problems of the Web

HTML is computer readable, not computer understandable.

While CSS provides some separation between content and presentation, this 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.



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.

The Semantic Web

Designed to address these problems.

1. By using semantic markup, the Web will itself become a huge database.
2. Rather than browsing the Web for information using search engines, the user will be able to make a request of a software agent which will be able to browse the web, compile and sort information and provide the user with only the relevant results.

Tim Berners-Lee describes their vision in an article in Scientific American (Berners-Lee, et al., 2001).

Data will be stored and sent in XML together with a style sheet. This will be responsible for looking after formatting and display on the browser. They key difference is the use of XML supported by ontologies.

Data could be presented by direct display of XML using XSL, or via translation to HTML using XSLT followed by formatting with CSS.

References

HTML and XHTML: The Definitive Guide, Chuck Musciano and Bill Kennedy, O'Reilly, 2002. ISBN: 059600382X

Cascading Style Sheets: Designing for the Web, Hakon Lie and Bert Bos, Addison Wesley, 1999. ISBN: 0201596253

JavaScript: The Definitive Guide, David Flanagan, O'Reilly, 2001. ISBN: 0596000480

Towards the Semantic Web: Ontology-driven Knowledge Management, John Davies, John Wiley and Sons Ltd., 2003. ISBN: 0470848677

Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce, D. Fensel, Springer Verlag, 2003. ISBN: 3540009663

XSLT, Doug Tidwell, O'Reilly, 2001. ISBN: 0596000537