XML is based on open standards, and is text-based, thereby making it accessible to all. It is extensible, thus allowing anyone to customise it for their own needs, to publish for others to use, and to use others' efforts for representation of data (e.g. MathML) or for the sharing of data between autonomous organisations (e.g. FixML).
Importantly, it is *verifiable*, either with a DTD or schema, allowing its validity to be confirmed.

Don't forget that although textual, XML is more designed to allow computers to communicate, and humans are increasingly less likely to encounter raw XML, seeing instead its benefits, e.g. RSS feeds & news aggregators. Indeed, the most successful uses of XML will be those that you do not even notice!

XML can be *well-formed* (i.e. conform to XML layout – balanced tags, properly nested), and it can also be *valid* (well-formed *and* verified against a DTD/schema).

XML Syntax:
`<? … ?>` indicates processing instructions
`<!-- … -->` indicates comments
`<tag/>` is empty-tag shorthand for `<tag></tag>`
`<![CDATA[ … ]]>` indicates literal data, not to be interpreted as XML
`<!ENTITY entity-name "entity value">` defines an entity, like a #define in C

There are **FIVE** built-in *entities* in XML,
`&amp;` (&), `&lt;` (<), `&gt;` (>), `&quot;` ("), `&apos;` (')

Elements vs Attributes:
Elements permit nesting, have an order, are easier to search for. They are a little more complicated, but more flexible.
Attributes allow simple values only, have no implied order, couple information more tightly to their element, are a little simpler to write.

DTDs vs Schemas:
DTDs are non-XML, no data typing, do not support namespaces, cannot easily enforce numbers of elements.
No entities in schemas; however, schemas allow definition of data types (simple and complex) as well as elements and attributes, elements names can be scoped in schemas.
xsd:sequence ≡ , = sequence
xsd:choice ≡ | = alternatives
DTDs are shorter, if somewhat limited, and are not written in XML.
Schemas are longer winded, but more flexible, and are written in XML themselves.

DTDs
Similar to regular expression and Extended Backus-Naur Form (EBNF).
`<!DOCTYPE` dtd-name `(SYSTEM` localuri `| PUBLIC` identifier,uri`) >`
`<!ELEMENT … >`
`<!ATTLIST` ele-name `(`att-name, type, default`)+ >`
attribute types include: CDATA, ID, IDREF, IDREFS
attributes defaults include: `#IMPLIED`, `#REQUIRED`, `#FIXED x`, `x`
, = sequence, | = alternatives, + = one or more, * = zero or more, ? = optional

Schemas (.xsd files)
Specifies elements, e.g. <xsd:element name="greeting" type="xsd:string"/>
`xsd:element` ≡ `<!ELEMENT>`
`xsd:attribute` ≡ `<!ATTLIST>`
minOccurs and maxOccurs used to limit elements, sequences, etc. Not possible with DTDs.
Can restrict by enumeration or by regular expression too.

Namespaces
Disambiguates between like-worded elements.
Form is <element `xlmns:name`=URI>
URI doesn't have to be real, just used to distinguish between like-worded elements and attributes.
Default namespaces are defined in elements which are valid for that element and nested elements.
Namepsaces are NOT supported within DTDs.

XPath – primary purpose is to address parts of an XML document. XPointer is an *extension* to XPath.
Document can be viewed as a tree of nodes, from the root elements downwards.
There are **SIX** types of node: root, element, attribute, text, comment, processing instruction.
The *value* of a node is the concatenation of all its text node descendants.
XPath expressions can be absolute (start with '/') or relative. Each step in a path is separated by a '/'.
An empty step '//' refers to *all* nodes from the current node, e.g. //mynode refers to all instances of mynode throughout the document, e.g.2 //performance[composer] refers to all performance elements that have a composer child element.
Node set functions, used to filter nodes from a set *S*: `last()`, `position()`, `count(S)`, `name(S)`, `id(S)`.
e.g. `count(//performance)` returns the number of performance elements.
e.g.2 `//performance[not(date)]` returns the performance elements that do not have a date element as a child.
Location Steps have the form `axis::node-test predicates`
There are ***13*** axes defined (self, parent, attribute, namespace, child, ancestor, descendant, ancestor-or-self, descendant-or-self, preceding, following, preceding-sibling, following-sibling).

Full syntax	Abbreviation
`child::`	`nothing (child is default axis)`
`attribute::`	`@`
`/descendant-or-self::node()/`	`//`
`self::node()`	`.`
`parent::node()`	`..`
`[position()=i]`	`[i]`

Extensible Style Language (XSL)
XSL is more complicated but also more powerful than CSS (Cascading Style Sheets).
XSL allows element re-ordering, selection, text generation, extensibility.
XSL comprises XSLT and XSL-FO (formatting objects).
XSLT can in fact, turn any XML file into almost anything else – HTML, another XML document, anything.
XSL is the presentation aspect of XML, since XML has nothing to do with presentation itself.
An XSLT stylesheet `<xsl:stylesheet>` comprises a number of templates
```
<xsl:template match="…">
```
pattern is specified using an XPath expression in the `match` attribute value, e.g.
```
<xsl:template match="atom">
  <p>
    <xsl:value-of select="name"/>
  </p>
</xsl:template>
```
XSLT processor takes in an XML document and an XSLT stylesheet and, starting at the root node, recursively processes nodes.
XSL has programming constructs such as a switch statement,
```
<xsl:choose>
  <xsl:when …>
    …
  </xsl:when>
  <xsl:otherwise>
    …
  </xsl:otherwise>
</xsl:choose>
```
and a loop statement,
```
<xsl:for-each …>
</xsl:for-each>
```

Scripting
Can be executed on loading or events occurring. There are 18 events in HTML incl. OnLoad, OnUnload, OnClick, on MouseDown, onMouseMove, onKeyPress.
JavaScript can be used to generate HTML

Document Object Model (DOM)
Defines an API for HTML and XML documents. Document is modelled as a tree of nodes. Using DOM programmers can build documents, navigate their structure, modify the document. DOM is platform-neutral and language-netural (i.e. independent).

Javascript can use DOM to navigate through an XML document.
Current HTML page is also a document, accessed in Javascript through the object `document`. It can be modified on the fly.

Internet Protocols
Physical – e.g. Ethernet, DECnet, ATM (moving bits)
Link – e.g. IP (moving packets point-to-point)
Transport – e.g. TCP, UDP (moving packets end-to-end)
Application – e.g. HTTP, FTP, DNS, SMTP, POP3, NNTP, Telnet (moving information)

TCP acknowledges safe receipt of packets, deals with missing, corrupted, duplicated packets. Provides a stream of data to the application
UDP provides connectionless service, faster than TCP, adds only checksum, used for DNS and NFS.

DNS resolves names to numbers, domains arranged hierarchically. TLDs include country codes, e.g. uk, fe, de, ca, and generic, e.g. com, edu, mil, gov, net, org

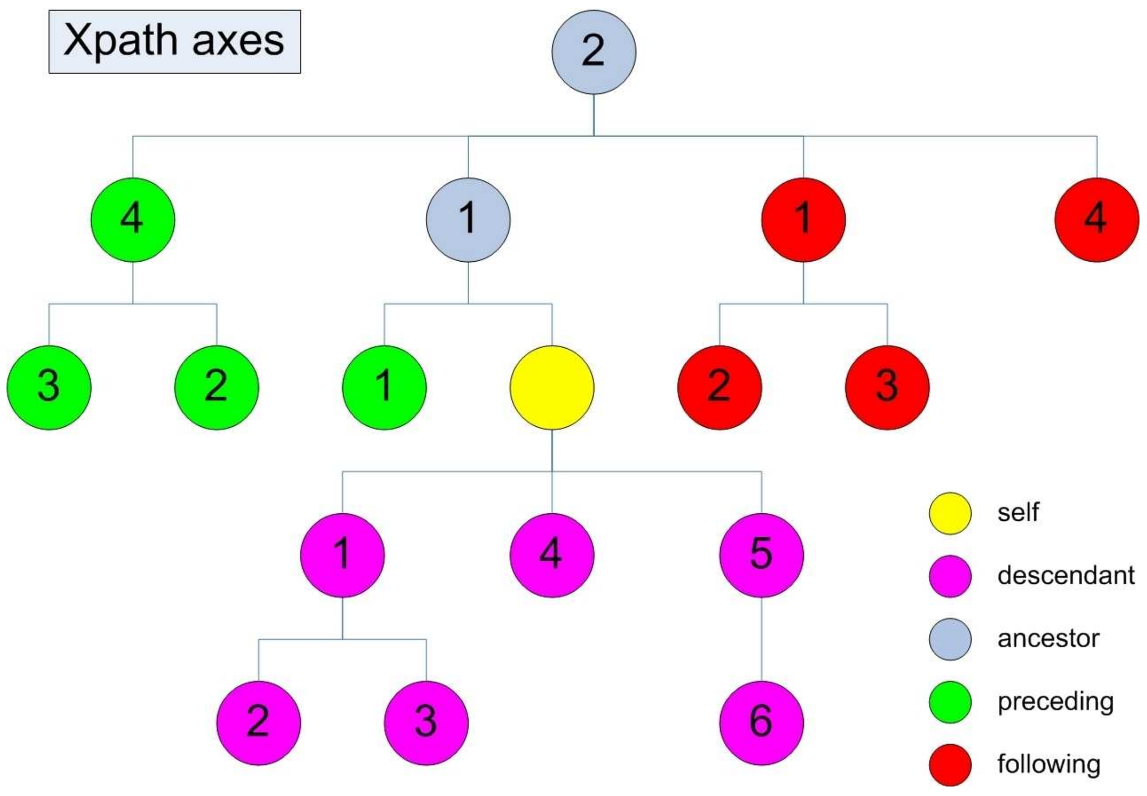
Multipurpose Internet Mail Extensions (**MIME**) uses 7-bit ASCII, allows non textual data to be transmitted.

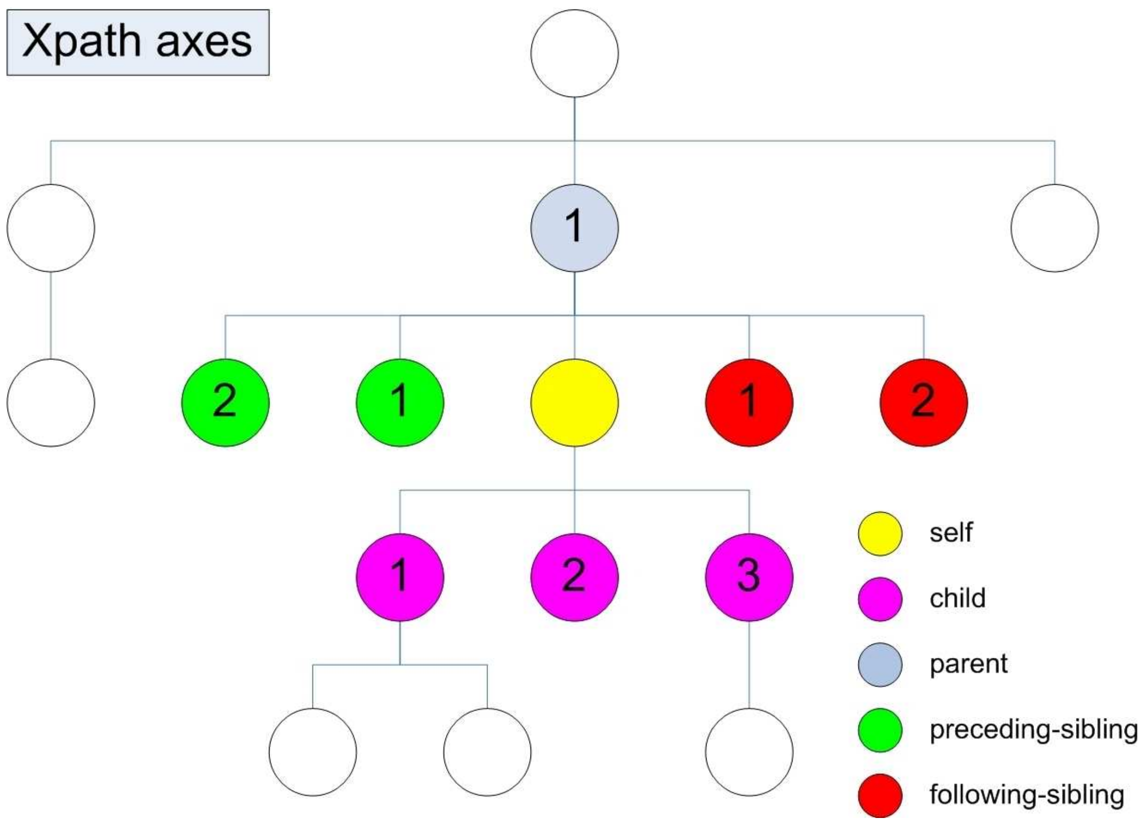
URIs – Uniform Resource *Identifiers*
% used as an escape character
# used to delimit a resource from a *fragment identifier*.
? used to delimit a resource from a query

URNs – Uniform Resource *Names* (not yet supported in browsers) RFC 2141 (1997)
Syntax, `urn:`<NID>`:`<NSS> where <NID> is namespace identifier, and <NSS> is namespace specific string. NISs registered with IANA include isbn, ietf, mpeg, uuid.
Resolved by sending NID to a *Resolver Discovery Service* (RDS) to get a URN resolver server address, then sending NSS to the resolver address to get a URL, then using the URL to access the resource.

HTTP Client requests, format `method request-URI HTTP-version`
e.g. GET /index.html HTTP/1.0
HTTP Server responses, format `HTTP-version status-code reason-phrase`
e.g. HTTP/1.0 200 OK
1.1 improvements over 1.0, more methods, persistent connections for performance (fewer TCP connection packets overhead), authentication, caching, can utilise proxies and gateways, content negotiation, pipelining.

Server-side processing, incl. CGI (Common Gateway Interface), Java servlets, Server-side includes (SSI). Java Server Pages are Sun's equivalent to Microsoft's ASP – Java programs are embedded in HTML. ASPs embed VBScript.
Desirable to separate content from presentation, e.g. Struts.


Xpath axes
2
4
1
1
4
3
2
1
2
3
1
4
5
2
3
6
self
descendant
ancestor
preceding
following


Xpath axes
1
2
1
1
2
1
2
3
self
child
parent
preceding-sibling
following-sibling