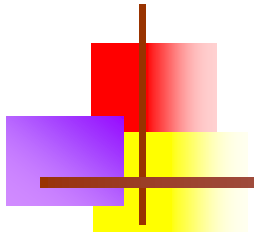
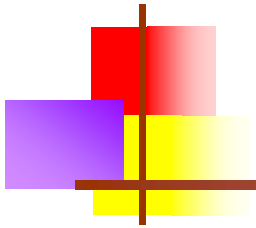


XML



HTML and XML, I

XML stands for eXtensible Markup Language

HTML is used to mark up text so it can be displayed to users

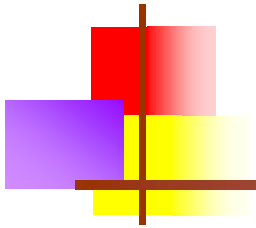
XML is used to mark up data so it can be processed by computers

HTML describes both structure (e.g. <p>, <h2>,) and appearance (e.g.
, <font>, <i>)

XML describes only content, or "meaning"

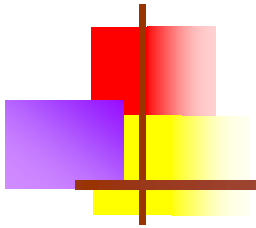
HTML uses a fixed, unchangeable set of tags

In XML, you make up your own tags



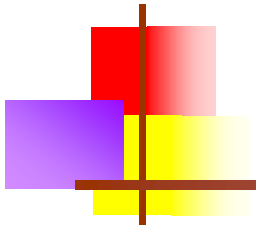
HTML and XML, II

- HTML and XML look similar, because they are both SGML languages (SGML = Standard Generalized Markup Language)
  - Both HTML and XML use elements enclosed in tags (e.g. <body>This is an element</body>)
  - Both use tag attributes (e.g., <font face="Verdana" size="+1" color="red">)
  - Both use entities (<, >, &, ", ')
- More precisely,
  - HTML is defined in SGML
  - XML is a (very small) subset of SGML



HTML and XML, III

- HTML is for humans
  - HTML describes web pages
  - You don’t want to see error messages about the web pages you visit
  - Browsers ignore and/or correct as many HTML errors as they can, so HTML is often sloppy
- XML is for computers
  - XML describes data
  - The rules are strict and errors are not allowed
    - In this way, XML is like a programming language
  - Current versions of most browsers can display XML
    - However, browser support of XML is spotty at best



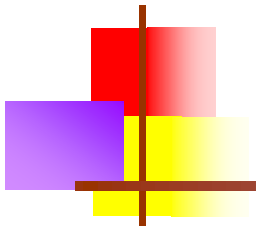
XML-related technologies

- DTD (Document Type Definition) and XML Schemas are used to define legal XML tags and their attributes for particular purposes
- CSS (Cascading Style Sheets) describe how to display HTML or XML in a browser
- XSLT (eXtensible Stylesheet Language Transformations) and XPath are used to translate from one form of XML to another
- DOM (Document Object Model), SAX (Simple API for XML, and JAXP (Java API for XML Processing) are all APIs for XML parsing

Example XML document

```
<?xml version="1.0"?>
<weatherReport>
   <date>7/14/97</date>
   <city>North Place</city>, <state>NX</state>
   <country>USA</country>
   High Temp: <high scale="F">103</high>
   Low Temp: <low scale="F">70</low>
   Morning: <morning>Partly cloudy, Hazy</morning>
   Afternoon: <afternoon>Sunny &amp;amp; hot</afternoon>
   Evening: <evening>Clear and Cooler</evening>
</weatherReport>
```

From: **XML: A Primer**, by Simon St. Laurent



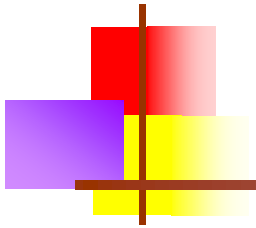
Overall structure

- An XML document may start with one or more processing instructions (PIs) or directives:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/css" href="ss.css"?>
```

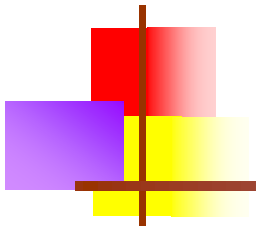
- Following the directives, there must be exactly *one* root element containing all the rest of the XML:

```
<weatherReport>
    …
</weatherReport>
```



XML building blocks

- Aside from the directives, an XML document is built from:
  - elements: high in <high scale="F">103</high>
  - tags, in pairs: <high scale="F">103</high>
  - attributes: <high scale="F">103</high>
  - entities: <afternoon>Sunny & hot</afternoon>
  - character data, which may be:
    - parsed (processed as XML)--this is the default
    - unparsed (all characters stand for themselves)



Elements and attributes

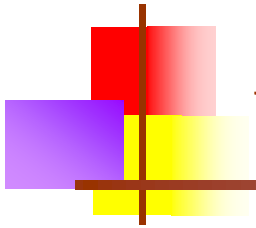
- Attributes and elements are somewhat interchangeable
- Example using just elements:

```
<name>
   <first>David</first>
   <last>Matuszek</last>
</name>
```

- Example using attributes:

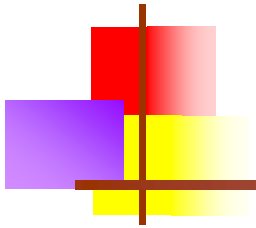
```
<name first="David" last="Matuszek"></name>
```

- You will find that elements are easier to use in your programs--this is a good reason to prefer them
- Attributes often contain metadata, such as unique IDs
- Generally speaking, browsers display only elements (values enclosed by tags), not tags and attributes



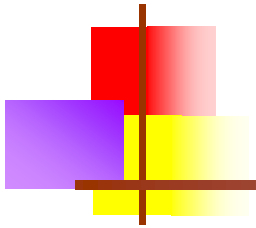
Well-formed XML

- Every element must have *both* a start tag and an end tag, e.g. <name> … </name>
  - But empty elements can be abbreviated: <break />.
  - XML tags are case sensitive
  - XML tags may not begin with the letters xml, in any combination of cases
- Elements must be properly nested, e.g. *not* bold and italic
- Every XML document must have one and only one root element
- The values of attributes must be enclosed in single or double quotes, e.g. <time unit="days">
- Character data cannot contain < or &



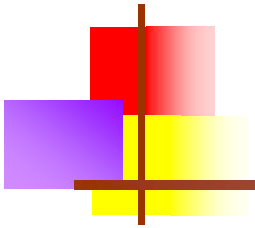
Entities

- Five special characters must be written as entities:
  - `&` for `&` (almost always necessary)
  - `<` for `<` (almost always necessary)
  - `>` for `>` (not usually necessary)
  - `&quot;` for `"` (necessary inside double quotes)
  - `&apos;` for `'` (necessary inside single quotes)
- These entities can be used even in places where they are not absolutely required
- These are the *only* predefined entities in XML



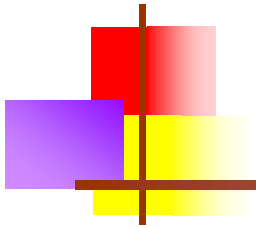
XML declaration

- The XML declaration looks like this:
  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  - The XML declaration is not required by browsers, but *is* required by most XML processors (so include it!)
  - If present, the XML declaration must be first--*not even whitespace* should precede it
  - Note that the brackets are <? and ?>
  - version="1.0" is required (this is the *only* version so far)
  - encoding can be "UTF-8" (ASCII) or "UTF-16" (Unicode), or something else, or it can be omitted
  - standalone tells whether there is a separate DTD



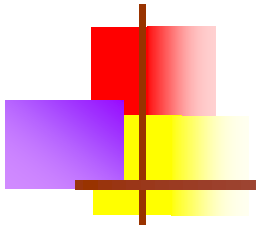
Processing instructions

- PIs (Processing Instructions) may occur anywhere in the XML document (but usually first)
- A PI is a command to the program processing the XML document to handle it in a certain way
- XML documents are typically processed by more than one program
- Programs that do not recognize a given PI should just ignore it
- General format of a PI: <?*target instructions*?>
- Example: <?xml-stylesheet type="text/css" href="mySheet.css"?>



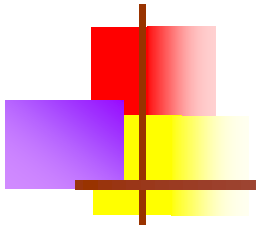
Comments

- `<!-- This is a comment in both HTML and XML -->`
- Comments can be put anywhere in an XML document
- Comments are useful for:
  - Explaining the structure of an XML document
  - Commenting out parts of the XML during development and testing
- Comments are not elements and do not have an end tag
- The blanks after `<!--` and before `-->` are optional
- The character sequence `--` cannot occur in the comment
- The closing bracket *must* be `-->`
- Comments are not displayed by browsers, but can be seen by anyone who looks at the source code



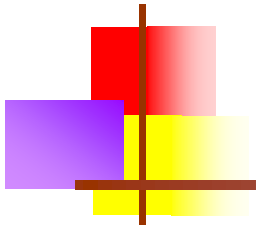
CDATA

- By default, all text inside an XML document is parsed
- You can force text to be treated as unparsed *character data* by enclosing it in `<![CDATA[ ... ]]>`
- Any characters, even `&` and `<`, can occur inside a CDATA
- Whitespace inside a CDATA is (usually) preserved
- The only real restriction is that the character sequence `]]>` cannot occur inside a CDATA
- CDATA is useful when your text has a lot of illegal characters (for example, if your XML document contains some HTML text)



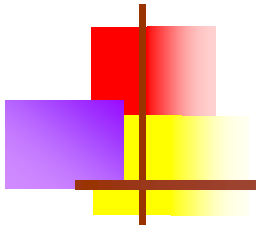
Names in XML

- Names (as used for tags and attributes) must begin with a letter or underscore, and can consist of:
  - Letters, both Roman (English) and foreign
  - Digits, both Roman and foreign
    - . (dot)
    - - (hyphen)
    - _ (underscore)
    - : (colon) should be used only for namespaces
  - Combining characters and extenders (not used in English)



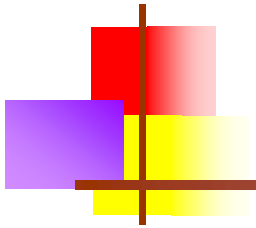
Namespaces

- Recall that DTDs are used to define the tags that can be used in an XML document
- An XML document may reference more than one DTD
- Namespaces are a way to specify which DTD defines a given tag
- XML, like Java, uses qualified names
  - This helps to avoid collisions between names
  - Java: `myObject.myVariable`
  - XML: `myDTD:myTag`
  - Note that XML uses a colon (`:`) rather than a dot (`.`)



Namespaces and URIs

- A namespace is defined as a unique string
  - To guarantee uniqueness, typically a URI (Uniform Resource Indicator) is used, because the author "owns" the domain
  - It doesn't have to be a "real" URI; it just has to be a unique string
  - Example: http://www.matuszek.org/ns
- There are two ways to use namespaces:
  - Declare a default namespace
  - Associate a prefix with a namespace, then use the prefix in the XML to refer to the namespace



Namespace syntax

- In *any* start tag you can use the reserved attribute name xmlns:

  ```
  <book xmlns="http://www.matuszek.org/ns">
  ```

  - This namespace will be used as the default for all elements up to the corresponding end tag
  - You can override it with a specific prefix

- You can use almost this same form to declare a prefix:

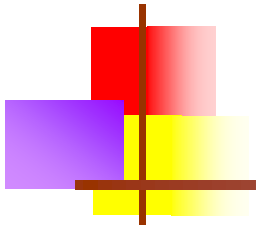
  ```
  <book xmlns:dave="http://www.matuszek.org/ns">
  ```

  - Use this prefix on *every tag and attribute* you want to use from this namespace, including end tags--it is *not* a default prefix

  ```
  <dave:chapter dave:number="1">To Begin</dave:chapter>
  ```

- You can use the prefix in the start tag in which it is defined:

  ```
  <dave:book xmlns:dave="http://www.matuszek.org/ns">
  ```



Review of XML rules

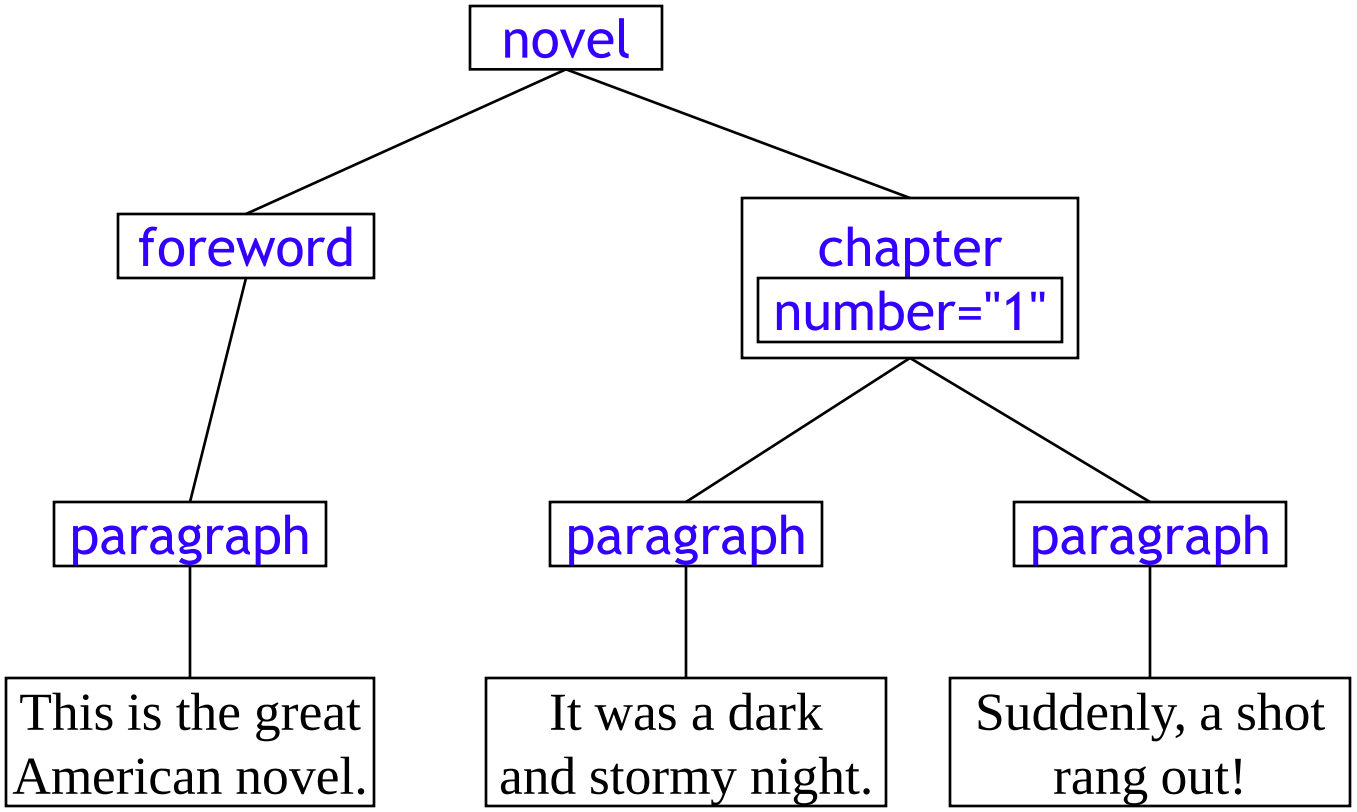
- Start with <?xml version="1"?>
- XML is case sensitive
- You must have exactly one root element that encloses all the rest of the XML
- Every element must have a closing tag
- Elements must be properly nested
- Attribute values must be enclosed in double or single quotation marks
- There are only five predeclared entities

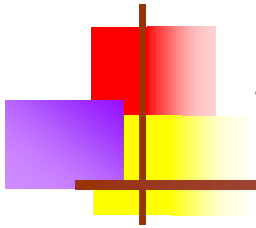
Another well-structured example

```
<novel>
   <foreword>
      <paragraph> This is the great American novel.
      </paragraph>
</foreword>
   <chapter number="1">
      <paragraph>It was a dark and stormy night.
     </paragraph>
      <paragraph>Suddenly, a shot rang out!
      </paragraph>
   </chapter>
</novel>
```

XML as a tree

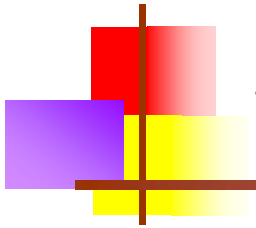
- An XML document represents a hierarchy; a hierarchy is a tree





Valid XML

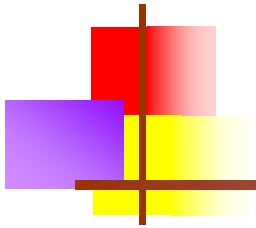
- You can make up your own XML tags and attributes, *but*...
  - ...any program that *uses* the XML must know what to expect!
- A DTD (Document Type Definition) defines what tags are legal and where they can occur in the XML
- An XML document *does not require* a DTD
- XML is well-structured if it follows the rules given earlier
- In addition, XML is valid if it declares a DTD and conforms to that DTD
- A DTD can be included in the XML, but is typically a separate document
- Errors in XML documents will *stop* XML programs
- Some alternatives to DTDs are XML Schemas and RELAX NG



Viewing XML

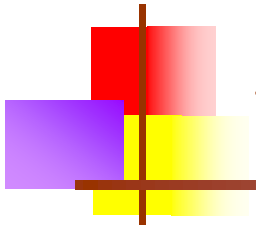
- XML is designed to be processed by computer programs, not to be displayed to humans
- Nevertheless, almost all current browsers can display XML documents
  - They don't all display it the same way
  - They may not display it at all if it has errors
  - For best results, update your browsers to the newest available versions
- Remember:

  HTML is designed to be *viewed*,
  XML is designed to be *used*



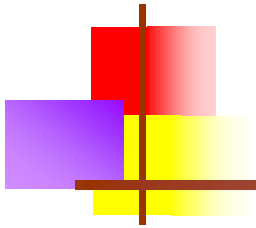
Extended document standards

- You can define your own XML tag sets, but here are some already available:
  - XHTML: HTML redefined in XML
  - SMIL: Synchronized Multimedia Integration Language
  - MathML: Mathematical Markup Language
  - SVG: Scalable Vector Graphics
  - DrawML: Drawing MetaLanguage
  - ICE: Information and Content Exchange
  - ebXML: Electronic Business with XML
  - cxml: Commerce XML
  - CBL: Common Business Library



Vocabulary

- SGML: Standard Generalized Markup Language
- XML : Extensible Markup Language
- DTD: Document Type Definition
- element: a start and end tag, along with their contents
- attribute: a value given in the start tag of an element
- entity: a representation of a particular character or string
- PI: a Processing Instruction, to possibly be used by a program that processes this XML
- namespace: a unique string that references a DTD
- well-formed XML: XML that follows the basic syntax rules
- valid XML: well-formed XML that conforms to a DTD



The End