

#### Enabling Grids for E-sciencE

### XML

Richard Hopkins National e-Science Centre, Edinburgh February 23 / 24 2005





www.eu-egee.org



- Goals
  - To understand the structure of an XML document
- Outline
  - Philosophy
  - General Aspects
  - Prolog
  - Elements
  - Namespaces
  - Concluding Remarks



### A Markup Language

### XML = eXtensible Markup Language

- "Markup" means document is an intermixing of
  - Content the actual information to be conveyed payload
  - Markup information about the content MetaData
     <date>22/10/1946</date>
    - <date> ... </date> is markup says that the content is a date
  - Self-describing document
  - date is part of a markup vocabulary -
    - a collection of keywords used to identify syntax and semantics of constructs in an XML document



### **Extensibility**

- XML = eXtensible Markup Language
- "Extensible" means the markup vocabulary is not fixed
- Compare with similar NON-extensible langhuage
  - HTML (Hypertext Markup Language)
  - Fixed markup vocabulary e.g
    - <strong> This </strong> is a paragraph. I like it. This is <strong> another </strong> paragraph
  - A presentation <u>language</u> for describing how a document should be presented for human consumption –
    - This is a paragraph. I like it.
    - This is **another** paragraph
  - For HTML the language is fixed and implicit in the fact that this is an HTML document – single-language document
- XML requires explicit definition of the language
- One document can combine multiple languages

4



### **Multi-lingual Documents**

Enabling Grids for E-sciencE

<businessforms:purchaseorder></businessforms:purchaseorder>	

	<date></date>	<usnotations:date></usnotations:date>	10/22/2004	
	<product></product>	<businessforms:barcode< th=""><th>&gt;123-768-252</th><th></th></businessforms:barcode<>	>123-768-252	
	<quantity></quantity>	<metricmeasures:kilos></metricmeasures:kilos>	17.53	

- businessForms:purchaseOrder
  - This is an instance of the purchaseOrder construct within the businessForms language
- BusinessForms (mythical)
  - A language defining structure of business documents
  - For business interoperability
  - Doesn't prescribe the language of individual items such as dates
- Language names are actually universally unique URIs www.DesperatelyTryingToStandardise.org/BusinessForms - see later



# Multilingual Pros & Cons

</..>

</..>

</ ></ >

6

Enabling Grids for E-sciencE

<businessforms:purchaseorder></businessforms:purchaseorder>			
<date></date>	<usnotations:date></usnotations:date>	10/22/2004	
<product></product>	<businessforms:barcode< td=""><td>&gt;123-768-252</td></businessforms:barcode<>	>123-768-252	
<quantity></quantity>	<metricmeasures:kilos></metricmeasures:kilos>	17.53	

</..>

#### • Separation of concerns – Design Factoring

- Design of purchase order structure and date format are independent concerns
- Re-use of language definitions, e.g. date formats in many languages
- Extensibility Purchase order accommodates new product identification schemes (e.g. ISBN for book stores)
- Of course, only works if both ends "understand" all languages used
- Makes things more complex
  - Creating and identifying the languages



# **Types of XML Language**

- Fundamental Standards, e.g.
  - SOAP
    - soap-envelope:header
- soap-envelope:body
- soap-envelope the language for soap messages
- A soap message is an XML document and its parts are identified using this vocabulary
- Goal is a factoring that gives pick-and-mix of combinable standards
- Associated with any WS standard will be a Schema definition of its XML language
- Community conventions
  - Perhaps, our BusinessForms language
- Specific Application Language
  - myProgram:parameter1
  - The language used in invoking particular operations of a web service

# egee

# **Human & Machine Oriented**

Enabling Grids for E-sciencE

#### How it really looks

<businessForms:purchaseOrder>

<date>

<USnotations:date>

10/22/2004

</USnotations:date>

</date>

<product>

<businessForms:barCode>

123-768-252

</businessForms:barCode>

</product>

<quantity>

<metricMeasures:kilos>

17.53

</metricMeasures:kilos>

</quantity>

</ businessForms:purchaseOrder >

Human readable

- Sort of OK with decent editor
- Is de-buggable
- Important for meta-data documents,
  - E.g. WSDL
- Machine processable
   Self description enables
  - General tools for producing and consuming XML documents
- Verbose
  - OK except for large data
  - Messages may have attachments not in XML

8



### **Philosophy Summary**

- Enabling Grids for E-sciencE
- XML goals
  - Self-describing documents
  - Hierarchic structure
  - Enabling multiple languages
  - Human readable and reasonably clear
  - Easy to write programs that generate them
  - Easy to write programs that process them
- For humans easier to read than to write
  - Leave detailed document creation to tools
  - But sometimes necessary to read them particularly meta-data such as WSDL
  - Often need to understand how to design them
- So rest of talk deals with some nitty gritty

9



# **GENERAL EMENTS**

Enabling Grids for E-sciencE

- Goals
  - To understand the structure of anXML document
- Outline
  - Philosophy
  - General Aspects
  - Prolog
  - Elements
  - Namespaces
  - Concluding Remarks





- Syntax
  - I will give syntax definitions of constructs
  - Mainly for your retrospective use
  - This uses notation similar to that used in the standard
    - <u>http://www.w3.org/TR/2004/REC-xml-20040204</u> (Ed. 3, Feb '04)
  - I will use some non-standard notation to make it a bit easier





[22] prolog [23] XMLDe		<u>XMLDecl? Misc* (doctypedecl Misc*)?</u> xml VersionInfo EncodingDecl? SDDecl S? ?		
[27] Misc ::=	= <u>Comment   PI   S</u>	/*syntax comment*/		
• [22]	definition number – sequer	definition number – sequentially numbered in the spec.		
<ul> <li>Prolog ::=</li> </ul>	construct is defined to be	construct is defined to be		
<u>XMLDecl</u>	include anything this const	include anything this construct (unerlined) can be		
• <i xml	itallic (times): exactly this (	itallic (times): exactly this (non-standard, spec. uses '>?xml')		
• ()	grouping (bold)	grouping (bold)		
• ? * +	? optional, * 0 or more, +	? optional, * 0 or more, + 1 or more,   alterantives		
• "" ""	content in matching quotes	content in matching quotes – "" or '' (non-standard)		
•	text with some natural rest	text with some natural restrictions (non-standard)		
•	as but allowing reference	as but allowing references (non-standard)		
• /* */	a comment on the syntax	a comment on the syntax		



### **Miscellaneous items**

Enabling Grids for E-sciencE

[27]

Misc ::=

#### Comment | PI | S

- A miscellaneous item is something outside the main structure –
- <u>S</u> Is white space
  - henceforth will ignore this aspect and leave it to common sense
  - there are specific rules
- Other two are "explanatory" material
- <u>Comment</u> for human consumption
- <u>PI</u> Processing Instruction
  - For S/W consumption
  - Information to assist the S/W that is processing the XML



- A valid comment
   <!-- This is a comment -->
- An invalid comment
  - <!--This is -- not a comment --->
- The "natural" restriction is
  - you can't have -- in a comment, except as the --> terminator



### **Processing Instructions**

Enabling Grids for E-sciencE

[27]	Misc ::=	Comment   PI   S	
[16]	PI ::=	<u PITarget ?>	/* excludes ?> */
[16]	PITarget ::=	Name	/*not xml or XmL etc. */

- Instructions to help the processing S/W
- <u>PITarget</u> identifies the intended S/W
   E.g.

<?xml-stylesheet type="text/ccs" href="greet.ccs" ?>

- There may some S/W processing this XML to present it in humanreadable form, using stylesheets to control formatting.
- Tells such S/W where the stylesheet is and what type it is.
- XML is a reserved target name standard instructions for basic XML processing. Likewise xml, XmL, xMl etc.



- Goals
  - To understand the structure of anXML document
- Outline
  - Philosophy
  - General Aspects
  - Prolog
  - Elements
  - Namespaces
  - Concluding Remarks



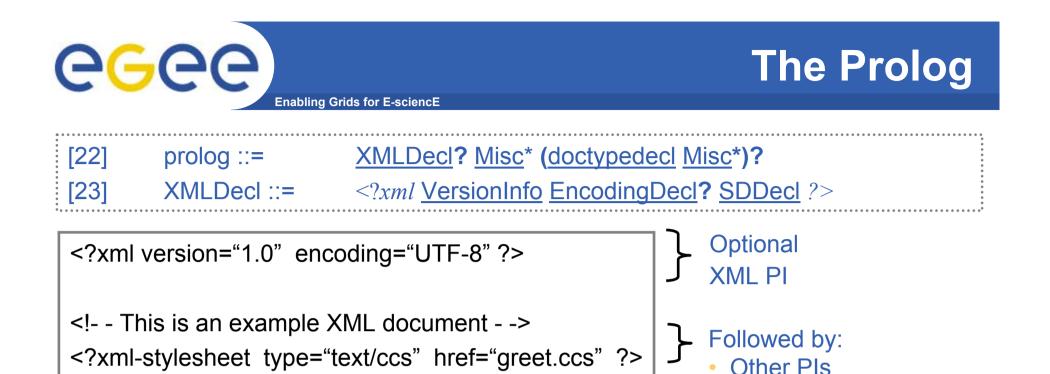
### **Document Structure**

#### Main structure of document is

- Prolog like headers
- Element the actual document



xml version="1.0" encoding="UTF-8" ?	ן   ן
This is an example XML document	> prolog
xml-stylesheet type="text/ccs" href="greet.ccs" ?	
<purchaseorder> </purchaseorder>	- Root element

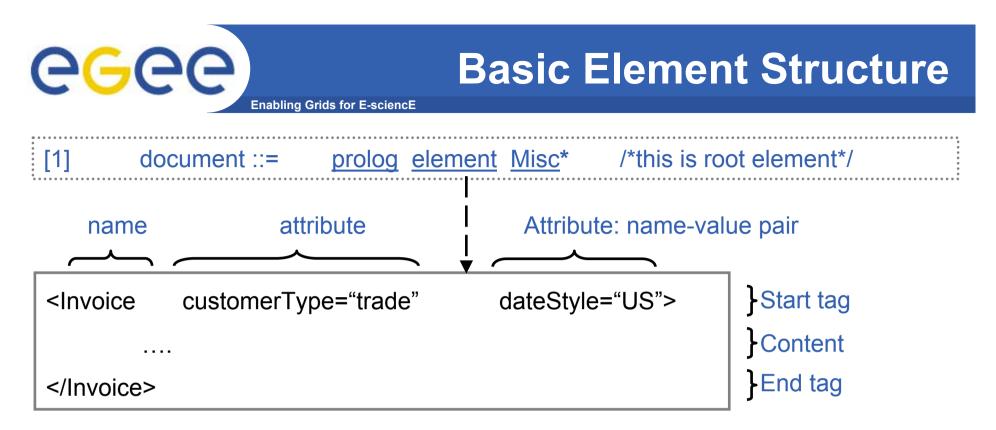


- <?XML ..?> PI is optional, but should be there;
  - if so must be first
  - gives version number must be 1.0 (for the 1.0 standard)
  - Could give the character encoding used –
     default is UTF-8, or something specifed at outer level (e.g HTTP header).
     ASCII is sub-set of UTF-8
  - <u>Doctypedecl</u> To do with Document Type Declarations (DTDs)
    - We are not using these, so ignore
  - <u>SDDecl</u> standalone declaration not clear when using schemas

Comments

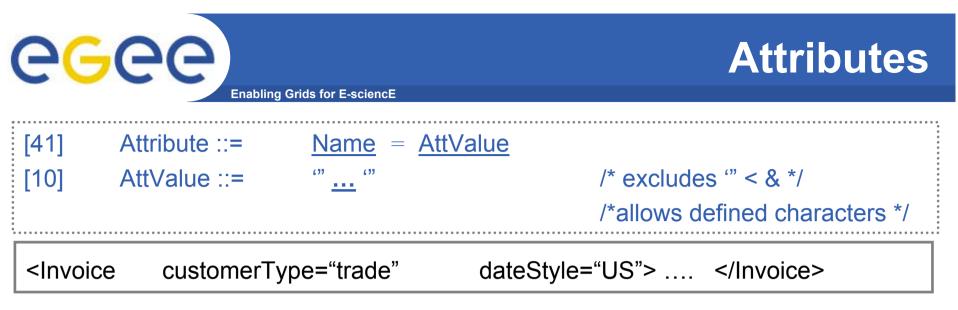


- Goals
  - To understand the structure of anXML document
- Outline
  - Philosophy
  - General Aspects
  - Prolog
  - Elements
  - Namespaces
  - Concluding Remarks



- Primary element structure
  - Start Tag <...>
    - Name of element
    - Zero or more attributes uniquely named; order insignificant
  - Content possibly nested <u>elements</u>, and other things
  - End Tag </ ... >
    - Name MUST be same name as in matching Start Tag
- Like HTML but stricter must have end tag

**Richard Hopkins** 



- A name-value pair that is included in the start tag of an element
- Name is part of specific language
- Value may also be part of a specific language QName qualified name
- More properly the above might be
  - < BusinessForms: Invoice

BusinessForms:customerType ="BusinessForms:trade"

BusinessForms:dateStyle="USnotations:date">

</BusinessForms:Invoice>

• This starts to get convoluted – necessary for designing for extensibility

. . .



# **Element Tags**

[39]	element ::=	STag content ETag	
0 0 0 0 0		EmptyElementTag	0 0 0 0 0 0 0
[40]	STag ::=	< <u>Name</u> ( <u>Attribute</u> )* >	
[42]	ETag ::=	<u Name >	
[40]	EmptyElemen	tTag ::= < <u>Name</u> ( <u>Attribute</u> )* />	
Empty Element - { <acceleration <aacceleration="" <accelerati<="" <acceleration="" <acceleration<="" acceleration="" td=""><td><invoice customertype="trade" datestyle="US"> <account accno="17-36-2" terms="days31"></account>  </invoice></td><td><pre>}Start tag } Content } End tag</pre></td></acceleration>		<invoice customertype="trade" datestyle="US"> <account accno="17-36-2" terms="days31"></account>  </invoice>	<pre>}Start tag } Content } End tag</pre>
Empty Element Tags -			

#### Empty Element Tags –

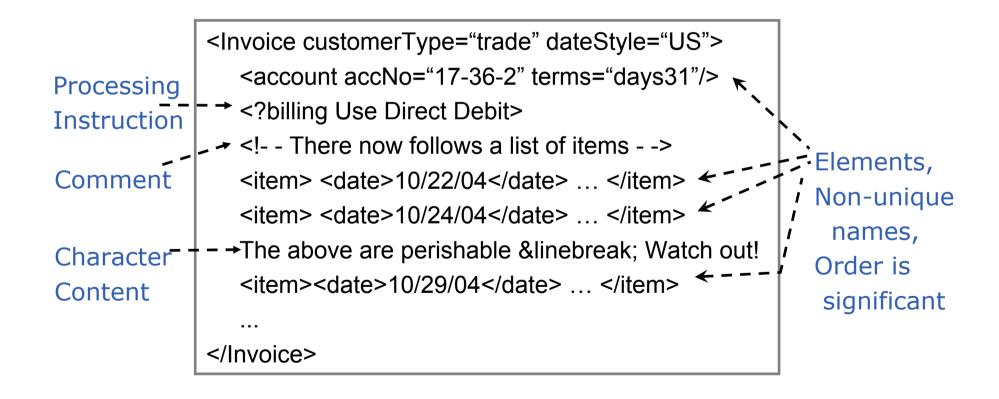
```
<account accNo="17-36-2" terms="days31"/>
```

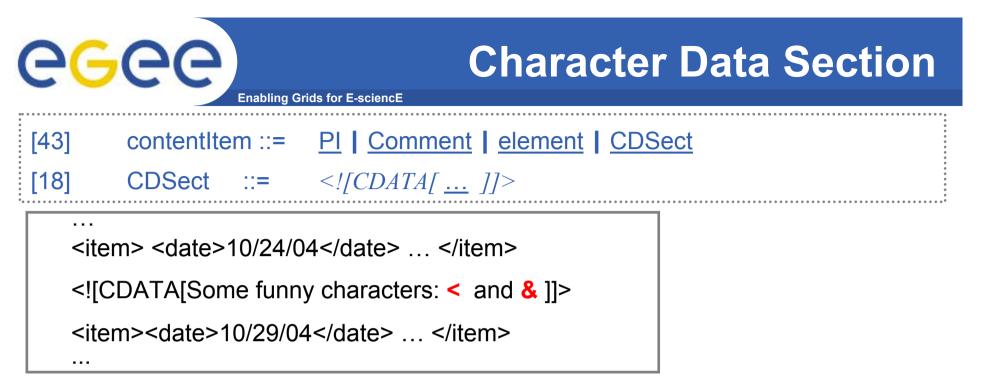
• Same as

```
<account accNo="17-36-2" terms="days31">
```

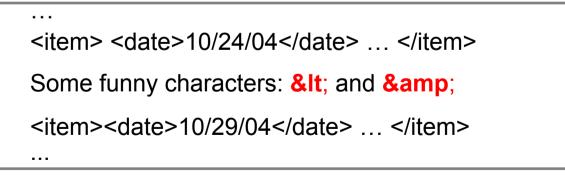
- </account >
- Shorthand for element with no content indicated by /> not >

ee	CCC Enabling G	ids for E-sciencE	Element Content
[39]	element ::=	STag content ETag   E	EmptyElementTag
[43]	content ::=	? ( contentItem? )*	
[43]	contentItem ::=	PI   Comment   Element	CDSect





- To make it easier to include characters which have special significance within XML – everything is taken literally except ]]>
- Alternative is -





<Invoice customerType="trade" dateStyle="US">

```
<item> <date>10/24/04</date> <price> 17.35 </price> ... </item>
```

```
The above are perishable &linebreak; Watch out!
```

```
<item><date>10/29/04</date> <price> 2173.35 </price> ... </item>
```

</Invoice>

- This is Mixed Content
  - Both direct character data and child elements (often excluded)
- Generally a bad idea for web services documents
- Better is each content item is either
  - Complex all child elements
  - Simple direct character data

```
<Invoice customerType="trade" dateStyle="US">
```

<item> <date>10/24/04</date> <price> 17.35 </price> ... </item>

<noteLine>The above are perishable &linebreak; Watch out!</noteLine>

<item><date>10/29/04</date> <price>2173.35</price> ... </item>

</Invoice>



# **Attribute vs Child**

Pure child element approach – no attributes anywhere

```
<customerType> trade </customerType>
<dateStyle> US </dateStyle>
<item>
<date> 10/24/04 </date>
<price>
<currency> Euro </currency>
<amount> 17.34 </amount>
</price>
...
```

Maximum attribute approach use attributes wherever possible

```
<Invoice
customerType="trade"
dateStyle="US" >
<item
date="10/24/04"
```

/>

</Invoice>

```
price-currency="Euro"
price-Amount="17.34"
```

</Invoice>

Can have unbounded number of item children

To use attribute approach for item would require defining infinite attributes item1-date item2-date .... Attribute names are unique within a tag Not possible

**Richard Hopkins** 



# **Attribute vs Child**

- Use Attributes for "control" information
  - Affects how we interpret/process the data
  - Typically a limited number of standard values Euro, USDollar, ...
  - Often essentially "type" info
- Use children for component data
  - Arbitrary values within the type (any date, any integer, any general string, ... )
- Distinction is fuzzy rather than absolute

#### Recommended style

```
<Invoice customerType="trade" dateStyle="US">
<item>
<date> 10/24/04 </date>
<price currency="Euro"> 17.34 </amount>
...
</item>
...
```





<invoice customertype="trade" datestyle="US"></invoice>			
<item></item>			
<date></date>	10/24/04		
<price currency="Euro"></price>	17.34		
<productcode></productcode>	17-23-57		
<quantity></quantity>	17.5		
<item></item>			
<date></date>	10/24/04		
<price currency="Euro"></price>	17.34		
<productcode></productcode>	17-23-57		
<quantity></quantity>	17.5		

- Will use XML a lot
  - Schemas, WSDL; Soap messages;
- Generally will use indentation to indicate structure and abbreviate End Tags to just </>
- Always have to actually put name in end tag !!!!



- Goals
  - To understand the structure of anXML document
- Outline
  - Philosophy
  - General Aspects
  - Prolog
  - Elements
  - Namespaces
  - Concluding Remarks



### Namespaces

```
<invoice> <!-- INT = International -->
<deliveryAddress>
<UK:address> ...<INT:street>...</> ...<UK:county>...</> <UK:postCode>...</></><billingAddress>
<US:address> ...<INT:street>...</> ...<US:state>...</> <US:zip>...</>
```

- A namespace (= "language")
  - Does define a collection of names (vocabulary)
    - For UK : {address, county, postCode, .... }
  - Would usually have an associated syntax (e.g. Schema definition)
    - address = ... county, postCode, ...
    - Syntax may be available to S/W processing it
  - Implies a semantics the (programmer writing) S/W processing a UK:address knows what it means.
  - Provides a unique prefix for disambiguating names from different originators
    - UK vs US vs INT



- To get uniqueness of namespace name, use a URI
  - UK:postCode is really

www.UKstandards.org/Web/XMLForms:postCode
(mythical)

- The URI <u>might</u> be a real URL, for accessing the syntax definition, documentation, ....
- But it may be just an identifier within the internet domain owned by the namespace owner



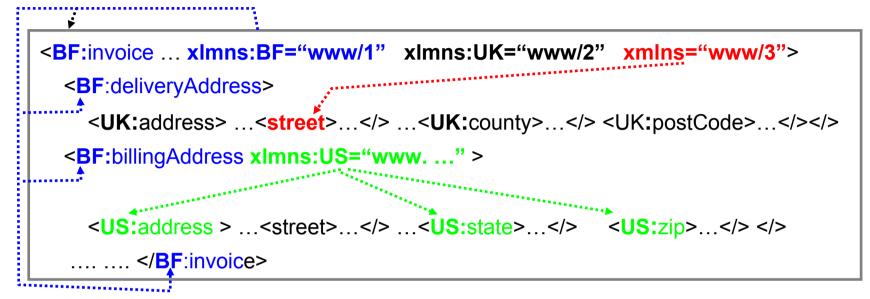
•

### **Namespace Names**

- To get uniqueness of namespace name, use a URI
  - UK:postCode is really www.UKstandards.org/Web/XMLForms:postCode
- But www.UKstandards.org/Web/XML/Forms:postCode is
  - Tediously long to use throughout the document
  - Outwith XML name syntax
    - Namespaces are not part of XML
    - A supplementary standard <u>http://www.w3.org/TR/REC-xml-names</u> A W3C recommendation
- In an XML document
  - declare a namespace prefix, as an attribute of an element
    - xmlns:UK="www.UKstandards.org/Web/XML/Forms"
  - then use that for names in that namespace UK:postCode
    - UK:post code is called a QName (qualified name)

### **Namespace Prefix Declarations**

Enabling Grids for E-sciencE



- Namespace declaration occurs as an attribute of an element
  - i.e. within a start tag
- Scope is from beginning of that start tag to matching end tag
  - Excluding scope of nested re-declarations of same prefix
- Can declare a default namespace
  - xlmns="www/3" this is the name space for all un-qualified names in the scope of this declaration, eg. Street
  - But not for attributes if no prefix, no namespace

GGGGG

### **Overriding namespace declarations**

Enabling Grids for E-sciencE

<document <u="">xmln:s1="www.1 xmlns="www/2" &gt;</document>				
< thing >	<s1:thing></s1:thing>			
< thing <u>xmlns:s1</u> ="www/1" >	<s1:thing></s1:thing>			
< thing >	<s1:thing></s1:thing>			

- xmlns:s1="www/1" Re-defines explicit namespace
- is bad idea –Unnecessary Confusion

- xmlns="www/xhtml" Re-defines default namespace reasonable
- Note if no default declared, then un-prefixed name has no namespace!

Richard Hopkins

**eGee** 

Web Services and WSRF, 24/25 Feb 2005, NeSc -- XML 34



- Goals
  - To understand the structure of anXML document
- Outline
  - Philosophy
  - General Aspects
  - Prolog
  - Elements
  - Namespaces
  - Concluding Remarks



- <u>Well-formed</u> means it confoms to the XML syntax, e.g.
  - Start and end tags nest properly with matching names
- <u>Valid</u> means it conforms to the syntax defined by the namespaces used
  - Can't check this without a definition of that syntax -
    - Normally a Schema
    - DTD (document Type Definitions) deprecated
    - Others type dfinition system
      - – some more sophisticated than Schemas



# **Final Comments**

- A specialisation of SGML a very general document markup language – any XML document is a an SGML document
- This is XML 1.0 Defined by WG3 a recommendation
  - <u>http://www.w3.org/TR/2004/REC-xml-20040204</u> (Ed. 3, Feb '04)
- Specification of the standard has a lot to do with DTDs which we have been ignoring – assume using Schemas instead
- A generalisation of HTML
  - But not an actual extension.
  - An HTML document is not an XML document
  - There is a XML specialisation XHTML which gives HTML functionality
- Definitions are now in terms of Infosets an abstraction of XML with XML being the standard representation





THE END