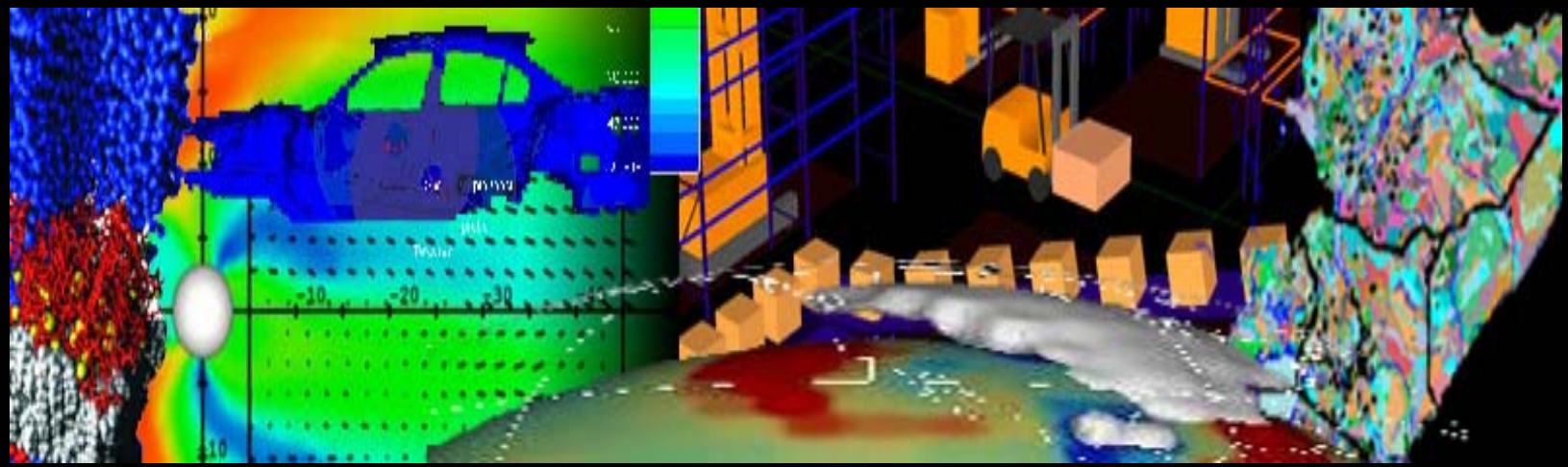
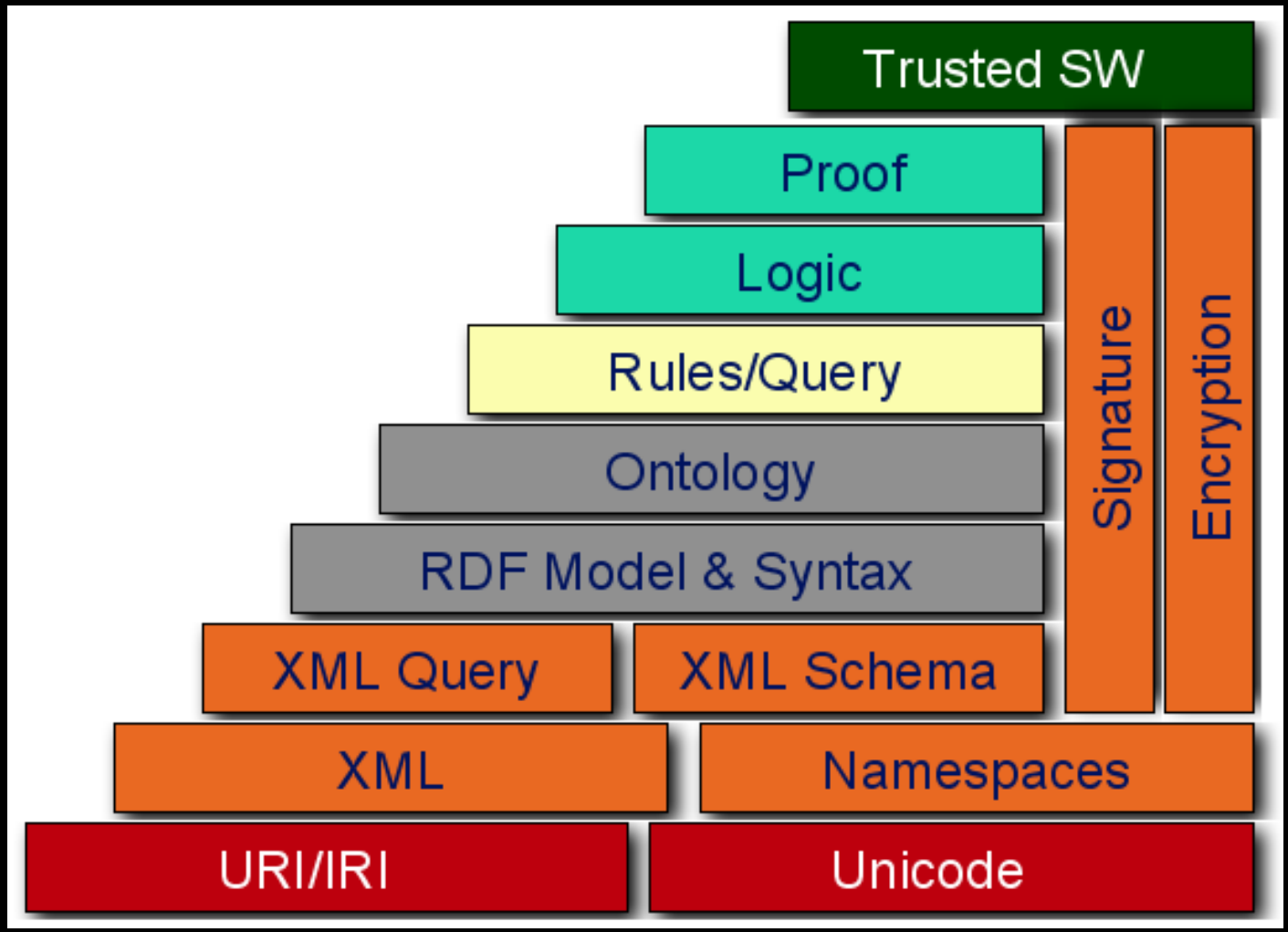
DATA CENTER

Make sense of your data

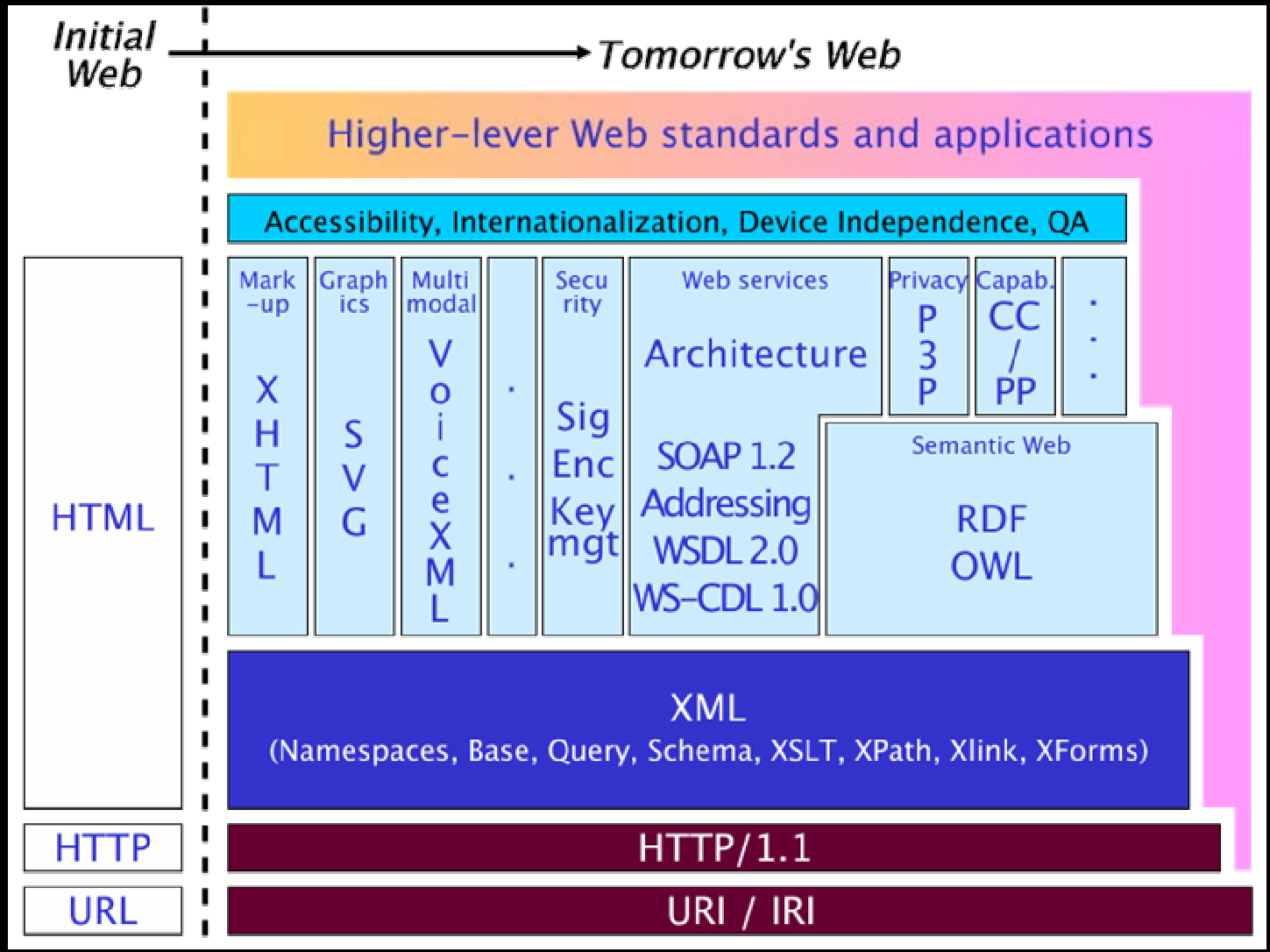


David Brock, Founder and Director
Data Center
Massachusetts Institute of Technology

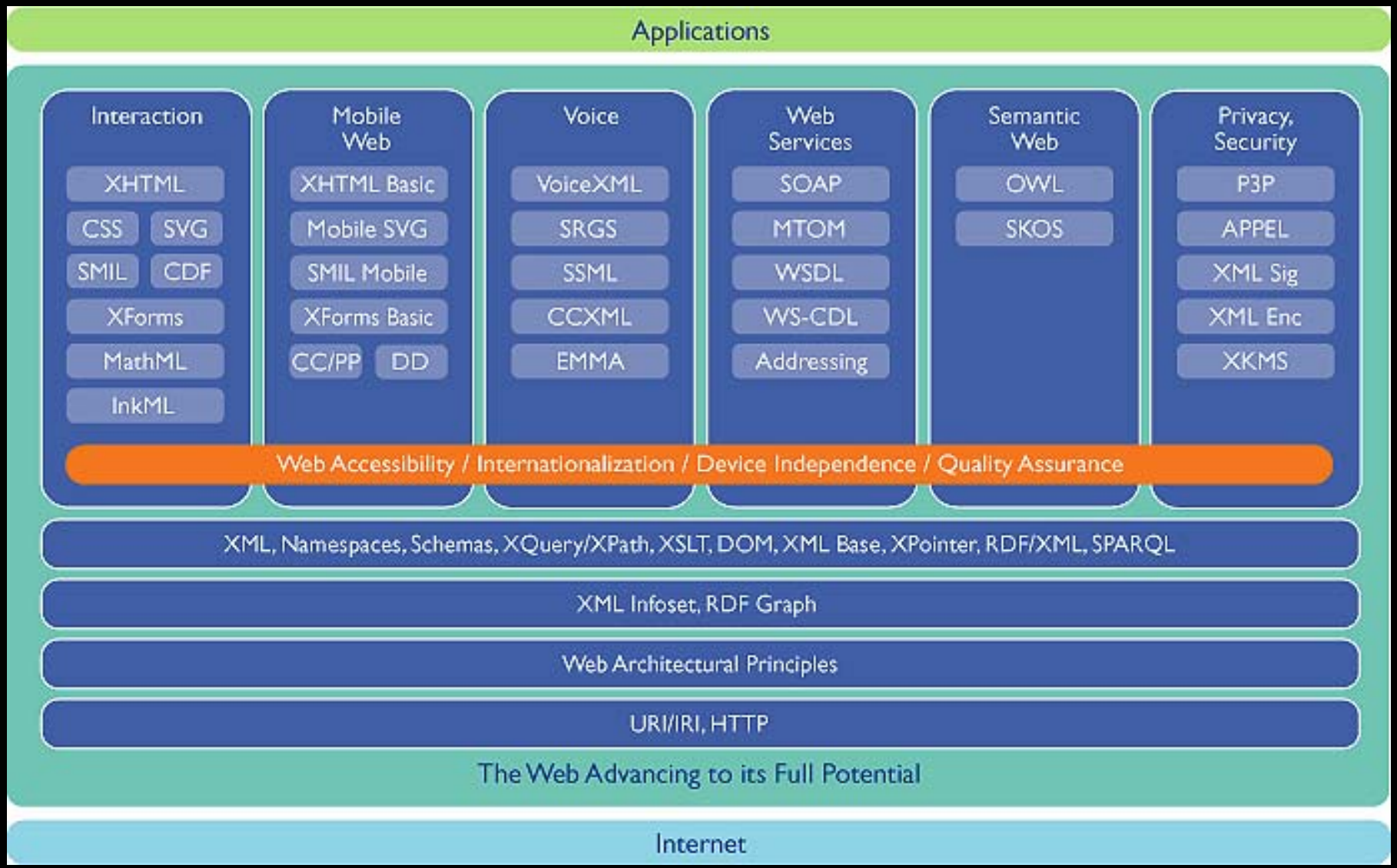
Web Standards

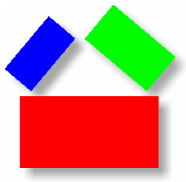


Web Standards



Web Standards





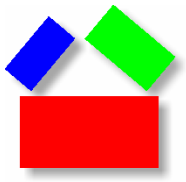
XML (eXtensible Markup Language)

Purpose

"Mark-up" or Tag Data

Standard

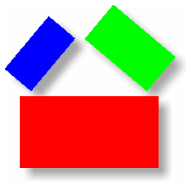
W3C Recommendation, February 1998



XML Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
  <book>
    <title lang="en">Harry Potter</title>
    <price>29.99</price>
  </book>
  <book>
    <title lang="en">Learning XML</title>
    <price>39.95</price>
  </book>
</bookstore>
```

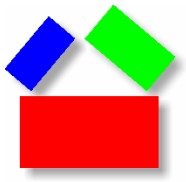
Examples from: W3 Schools (http://www.w3schools.com)



XML - Example

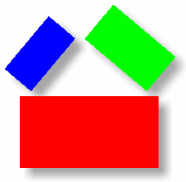
XML Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
  <book>
    <title lang="en">Harry Potter</title>
    <price>29.99</price>
  </book>
  <book>
    <title lang="en">Learning XML</title>
    <price>39.95</price>
  </book>
</bookstore>
```



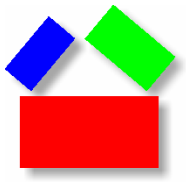
XML General Form

```
<root attribute="attribute">
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```



Comments on XML

- XML is designed to carry data
- XML does not “do” anything
- Must “make-up” your own tags
- Does not replace HTML
- Forms basis for other languages



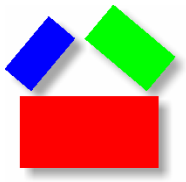
XML Namespace

Purpose

Defines source of ‘names’ in XML

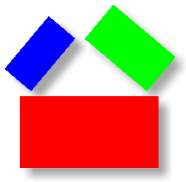
Standard

W3C Recommendation, February 1998



XML Namespace Example

```
<f:table xmlns:f="http://www.example.com/furniture">
   <f:name>African Coffee Table</f:name>
   <f:width>80</f:width>
   <f:length>120</f:length>
</f:table>
```



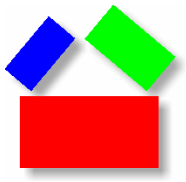
XML Schema

Purpose

- Describes an XML document

Standard

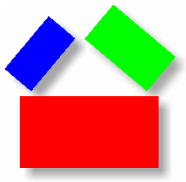
- W3C Recommendation, May 2001



XML Schema - Example

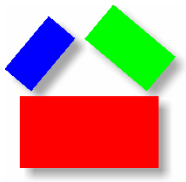
```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.example.com"
xmlns="http://www.example.com"
elementFormDefault="qualified">
<xs:element name="bookstore">
    <xs:complexType>
      <xs:sequence>
          <xs:element name="title">
             <xs:complexType>
                 <xs:simpleContent>
                     <xs:extension base="xs:string">
                         <xs:attribute name="lang" type="xs:string" default="en"/>
                     </xs:extension>
                 </xs:simpleContent>
             </xs:complexType>
          </xs:element>
          <xs:element name="price" type="xs:decimal"/>
      </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
  <book>
    <title lang="en">Harry Potter</title>
    <price>29.99</price>
  </book>
  <book>
    <title lang="en">Learning XML</title>
    <price>39.95</price>
  </book>
</bookstore>
```



Comments on XML Schema

- Specify the structure of XML nodes
- Specify type of XML content (e.g. string, integers, date, time, etc.)
- Specify restrictions (e.g. default, minimum, maximum, patterns, lists, etc.)
- Provides mechanisms for reuse and modularity via 'types'



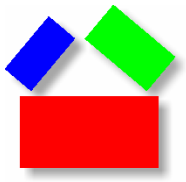
XML Schema – Data types and restrictions

Types

- xs:string
- xs:decimal
- xs:integer
- xs:boolean
- xs:date
- xs:time

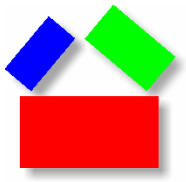
Restrictions

- enumeration
- fractionDigits
- length
- minExclusive , maxExclusive
- minInclusive, maxInclusive
- minLength, maxLength
- pattern
- totalDigits
- whiteSpace



XML Schema – Modularity and reuse

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>


<xs:element name="person" type="persontype"/>

<xs:complexType name="persontype">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```



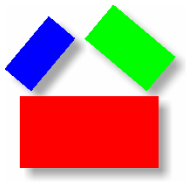
XPath

Purpose

Describes path to nodes in XML

Standard

W3C Recommendation, November 1999

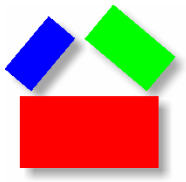


XPath - Example

XPath Example

bookstore/book

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
  <book>
    <title lang="eng">Harry Potter</title>
    <price>29.99</price>
  </book>
  <book>
    <title lang="eng">Learning XML</title>
    <price>39.95</price>
  </book>
</bookstore>
```

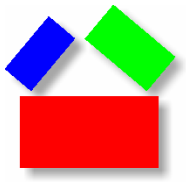


XPath - Example

XPath Example

bookstore/book

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
  <book>
    <title lang="eng">Harry Potter</title>
    <price>29.99</price>
  </book>
  <book>
    <title lang="eng">Learning XML</title>
    <price>39.95</price>
  </book>
</bookstore>
```

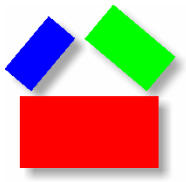


XPath - Example

XPath Example

bookstore//title

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
  <book>
    <title lang="eng">Harry Potter</title>
    <price>29.99</price>
  </book>
  <book>
    <title lang="eng">Learning XML</title>
    <price>39.95</price>
  </book>
</bookstore>
```

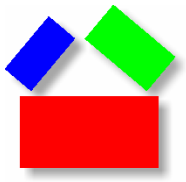


XPath - Example

XPath Example

bookstore/book/price/text()

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
  <book>
    <title lang="eng">Harry Potter</title>
    <price>29.99</price>
  </book>
  <book>
    <title lang="eng">Learning XML</title>
    <price>39.95</price>
  </book>
</bookstore>
```

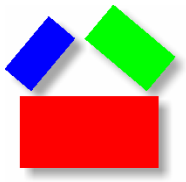


XPath - Example

XPath Example

bookstore/book[price>35]/price/text()

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
  <book>
    <title lang="eng">Harry Potter</title>
    <price>29.99</price>
  </book>
  <book>
    <title lang="eng">Learning XML</title>
    <price>39.95</price>
  </book>
</bookstore>
```



XPath – General form

XPath – general form

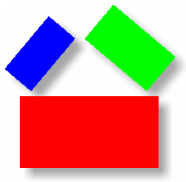
```
step/step/step …
```

```
axis::nodetest[predicate]
```

Axis - specifies the tree relationship between the nodes selected by the location step and the current node.

Node Test - specifies the node type and expanded-name of the nodes selected by the location step.

Predicates - expressions to further refine the set of nodes selected by the location step.



XPath – Axis Names

ancestor Contains all ancestors (parent, grandparent, etc.) of the current node

ancestor-or-self Contains the current node plus all its ancestors (parent, grandparent, etc.)

attribute Contains all attributes of the current node

child Contains all children of the current node

descendant Contains all descendants (children, grandchildren, etc.) of the current node

descendant-or-self Contains the current node plus all its descendants (children, grandchildren, etc.)

following Contains everything in the document after the closing tag of the current node

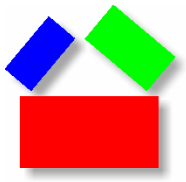
following-sibling Contains all siblings after the current node

namespace Contains all namespace nodes of the current node

parent Contains the parent of the current node

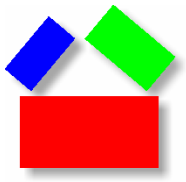
preceding Contains everything in the document that is before the starting tag of the current node

preceding-sibling Contains all siblings before the current node



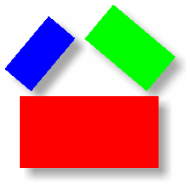
XPath – Expressions

operator	description	example	result
+	addition	6 + 4	10
-	subtraction	6 - 4	2
*	multiplication	64	24
div	division	8 div 4	2
mod	modulus (division remainder)	5 mod 2	1
=	like (equal)	price=9.80	true / false
!=	not like (not equal)	price != 9.80	true / false
<	less than	price < 9.80	true / false
<=	less or equal	price <= 9.80	true / false
>	greater than	price > 9.80	true / false
>=	greater or equal	price >= 9.80	true / false
or	or	price=9.80 or price = 9.70	true / false
and	and	price <= 9.80 and price=9.70	true / false



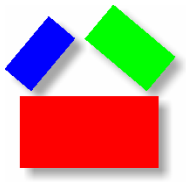
XPath – Functions (node set)

Name	Description	Syntax
count()	Returns the number of nodes in a node-set	number=count(node-set)
id()	Selects elements by their unique ID	node-set=id(value)
last()	Returns the position number of the last node in the processed node list	number=last()
local-name()	Returns the local part of a node. A node usually consists of a prefix, a colon, followed by the local name	string=local-name(node)
name()	Returns the name of a node	string=name(node)
namespace-uri()	Returns the namespace URI of a specified node	uri=namespace-uri(node)
position()	Returns the position in the node list of the node that is currently being processed	number=position()



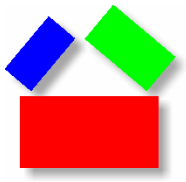
XPath – Functions (string)

Name	Description	Syntax & Example
concat()	Returns the concatenation of all its arguments	string=concat(val1, val2, ..) Ex: concat('The',' ','XML') Result: 'The XML'
contains()	Returns true if the second string is contained within the first string, otherwise it returns false	bool=contains(val,substr) Ex: contains('XML','X') Result: true
normalize-space()	Removes leading and trailing spaces from a string, and replaces all internal sequences of white with one white space	string=normalize-space(string) Ex: Normalize-space(' The XML ') Result: 'The XML'
starts-with()	Returns true if the first string starts with the second string, otherwise it returns false	bool=starts-with(string,substr) Ex: starts-with('XML','X') Result: true
string()	Converts the value argument to a string	string(value) Ex: string(314) Result: '314'
string-length()	Returns the number of characters in a string	number=string-length(string) Ex: string-length('Beatles') Result: 7
substring()	Returns a part of the string in the string argument	string=substring(string,start,length) Ex: substring('Beatles',1,4) Result: 'Beat'
substring-after()	Returns the part of the string in the string argument that occurs after the substring in the substr argument	string=substring-after(string,substr) Ex: substring-after('12/10','/') Result: '10'
substring-before()	Returns the part of the string in the string argument that occurs before the substring in the substr argument	string=substring-before(string,substr) Ex: substring-before('12/10','/') Result: '12'
translate()	Performs a character by character replacement. It looks in the value argument for characters contained in string1, and replaces each character for the one in the same position in the string2	string=translate(value,string1,string2) Ex: translate('12:30','30','45') Result: '12:45' Ex: translate('12:30','03','54') Result: '12:45' Ex: translate('12:30','0123','abcd') Result: 'bc:da'



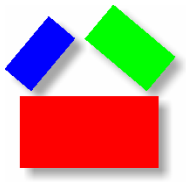
XPath – Functions (number)

Name	Description	Syntax & Example
ceiling()	Returns the smallest integer that is not less than the number argument	number=ceiling(number) **Example:** ceiling(3.14) Result: 4
floor()	Returns the largest integer that is not greater than the number argument	number=floor(number) **Example:** floor(3.14) Result: 3
number()	Converts the value argument to a number	number=number(value) **Example:** number('100') Result: 100
round()	Rounds the number argument to the nearest integer	integer=round(number) **Example:** round(3.14) Result: 3
sum()	Returns the total value of a set of numeric values in a node-set	number=sum(nodeset) **Example:** sum(/cd/price)



XPath – Functions (Boolean)

Name	Description	Syntax & Example
boolean()	Converts the value argument to Boolean and returns true or false	bool=boolean(value)
false()	Returns false	false() Example: number(false()) Result: 0
lang()	Returns true if the language argument matches the language of the xsl:lang element, otherwise it returns false	bool=lang(language)
not()	Returns true if the condition argument is false, and false if the condition argument is true	bool=not(condition) Example: not(false())
true()	Returns true	true() Example: number(true()) Result: 1



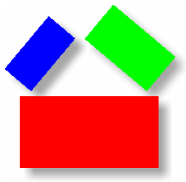
XLink

Purpose

Describes hyperlinks in XML

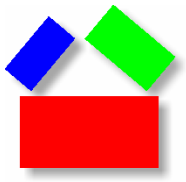
Standard

W3C Recommendation, June 2001



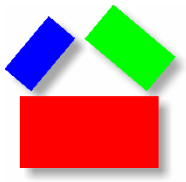
XLink - Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore xmlns:xlink="http://www.w3.org/1999/xlink">
  <book>
    <title lang="eng">Harry Potter</title>
    <description
       xlink:type="simple"
       xlink:href="http://book.com/images/HPotter.gif"
       xlink:show="new">
       Hogwarts School
    </description>
    <price>29.99</price>
  </book>
  <book>
    <title lang="eng">Learning XML</title>
    <price>39.95</price>
  </book>
</bookstore>
```



XLink - Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore xmlns:xlink="http://www.w3.org/1999/xlink">
  <book>
    <title lang="eng">Harry Potter</title>
    <publisher
       xlink:type="simple"
       xlink:href="http://www.bloomsbury.com/info.xml" />
    <price>29.99</price>
  </book>
  <book>
    <title lang="eng">Learning XML</title>
    <price>39.95</price>
  </book>
</bookstore>
```



XPointer

Purpose

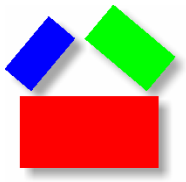
- Describes hyperlinks to parts of XML

Standard

- W3C Recommendation, March 2003

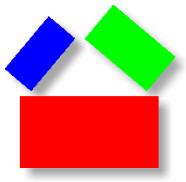
Uses

- XPath



XPointer - Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore xmlns:xlink="http://www.w3.org/1999/xlink">
  <book>
    <title lang="eng">Harry Potter</title>
    <publisher
       xlink:type="simple"
       xlink:href="http://www.bloomsbury.com/
       info.xml#xpointer(corp//public)" />
    <price>29.99</price>
  </book>
  <book>
    <title lang="eng">Learning XML</title>
    <price>39.95</price>
  </book>
</bookstore>
```



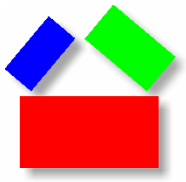
XSLT (eXtensible Stylesheet Language Transformation)

Purpose

- Describes transformations of XML to XML
  (originally intended to transform XML to HTML)

Standard

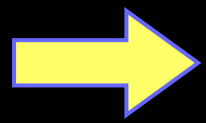
- W3C Recommendation, November 1999



XSLT - Example

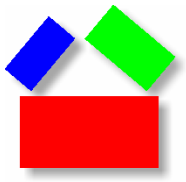
XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
.
.
.
</catalog>
```



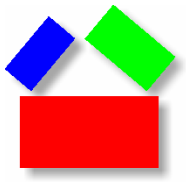
XSLT

```
<xsl:transform version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
    <tr bgcolor="#9acd32">
      <th align="left">Title</th>
      <th align="left">Artist</th>
    </tr>
    <xsl:for-each select="catalog/cd">
    <tr>
      <td><xsl:value-of select="title"/></td>
      <td><xsl:value-of select="artist"/></td>
    </tr>
    </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>

</xsl:transform>
```



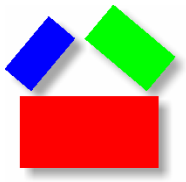
XSLT - Example

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
        <xsl:apply-templates/>
</xsl:template>

<xsl:template match="Order">
        <order.4><xsl:apply-templates/></order.4></xsl:template>
<xsl:template match="Cardnum">
        <credit_card.1><xsl:apply-templates/></credit_card.1></xsl:template>
<xsl:template match="Name">
        <name.2><xsl:apply-templates/></name.2></xsl:template>
<xsl:template match="Title">
        <title.8><xsl:apply-templates/></title.8></xsl:template>
<xsl:template match="Quantity">
        <quantity.1><xsl:apply-templates/></quantity.1></xsl:template>
</xsl:stylesheet>
```



XSLT – Example Elementss

```
<xsl:template  match="XPath expression">

<xsl:value-of select="XPath expression">

<xsl:for-each select="XPath expression">

<xsl:if         test="XPath expression">

<xsl:choose>
  <xsl:when test="expression">
    ... some output ...
  </xsl:when>
  <xsl:otherwise>
    ... some output ....
  </xsl:otherwise>
</xsl:choose>
```



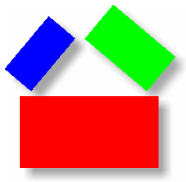
XSL-FO (eXtensible Stylesheet Language Formatting Objects)

Purpose

- Formats XML data for output
  (originally named XSL)

Standard

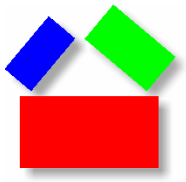
- W3C Recommendation, October 2001



XSL-FO - Example

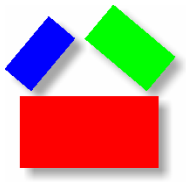
XSL-FO Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">

    <fo:layout-master-set>
      <fo:simple-page-master master-name="name">
        <!-- Page template goes here -->
      </fo:simple-page-master>
    </fo:layout-master-set>

    <fo:page-sequence master-reference="name">
      <!-- Page content goes here -->
    </fo:page-sequence>

</fo:root>
```



XSL-FO - Notes

- Uses “boxes” or “frames” to display output
- Hierarchical frame formatting (pages, regions, block areas, line areas and inline areas)
- Blocks “flow” into pages
- Intended for print and web presentation
- Outputs PDF and HTML



XQuery (XML Query Language)

Purpose

- Language to query XML data
  (analogous to SQL)

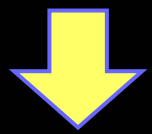
Standard

- W3C Proposal (XQL), February 1998
- W3C Candidate, November 2005

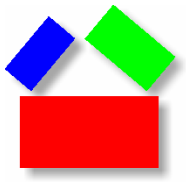
XQuery Example

```
doc("books.xml")/bookstore/book[price<30]
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
  <book>
    <title lang="en">Harry Potter</title>
    <price>29.99</price>
  </book>
  <book>
    <title lang="en">Learning XML</title>
    <price>39.95</price>
  </book>
</bookstore>
```



```
<book>
  <title lang="en">Harry Potter</title>
  <price>29.99</price>
</book>
```



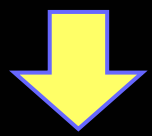
XQuery - Comments

- Uses XPath and FLWOR expression
- FLWOR (FOR, LET, WHERE, ORDER BY, RETURN)
- Provides iteration (for-loops) and conditionals (if-then-else)
- Allows XML or XHTML output
- Query 1.0, XPath 2.0, and XSLT 2.0 share same functions library

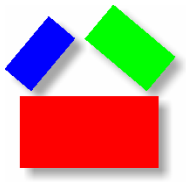
XQuery Example

```
for $x in doc("books.xml")/bookstore/book
where $x/price>30
order by $x/title
return $x/title
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
  <book>
    <title lang="en">Harry Potter</title>
    <price>29.99</price>
  </book>
  <book>
    <title lang="en">Learning XML</title>
    <price>39.95</price>
  </book>
</bookstore>
```



```
<title lang="en">Harry Potter</title>
```



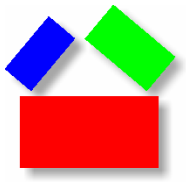
SOAP (Simple Object Access Protocol)

Purpose

- Communication protocol bases on XML

Standard

- W3C Proposal, May 2000
- W3C Draft, December 2001
- W3C Recommendation,



SOAP

SOAP Message

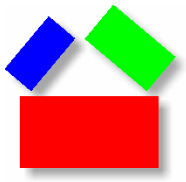
```
<?xml version="1.0"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Header>
    ...
    ...
  </soap:Header>
  <soap:Body>
    ...
    ...
    <soap:Fault>
      ...
      ...
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

envelope - root element, defines XML document as SOAP message

header - (optional) application information, such as authentication or payment

body - (required) actual message

fault - (optional) error and status information



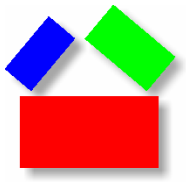
SOAP - Example

SOAP Request

```
OST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body xmlns:m="http://www.example.org/stock">
    <m:GetStockPrice>
      <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```

SOAP Response

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body xmlns:m="http://www.example.org/stock">
    <m:GetStockPriceResponse>
      <m:Price>34.5</m:Price>
    </m:GetStockPriceResponse>
  </soap:Body>
</soap:Envelope>
```



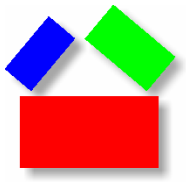
WSDL (Web Services Description Language)

Purpose

- Describes web services

Standard

- W3C Proposal, March 2001
- W3C Draft, July 2002
- W3C Recommendation,



WSDL

WSDL

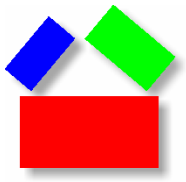
```
<definitions>

   <types>
      definition of types........
   </types>

   <message>
      definition of a message....
   </message>

   <portType>
      definition of a port.......
   </portType>

   <binding>
      definition of a binding....
   </binding>

</definitions>
```

types - defines data types used by the web service

message - defines data elements of an operation

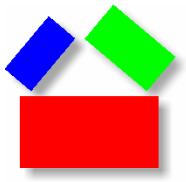
portType - describes web service - its operations and messages (most important)

binding - defines message format and protocol for each port



WSDL - Example

```
<message name="getTermRequest">
   <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
   <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="getTerm">
      <input message="getTermRequest"/>
      <output message="getTermResponse"/>
  </operation>
</portType>

<binding type="glossaryTerms" name="b1">
<soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http" />
  <operation>
    <soap:operation
     soapAction="http://example.com/getTerm"/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
</binding>
```



UDDI

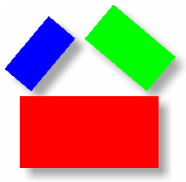
UDDI (Universal Description, Discovery and Integration)

Purpose

Describing, discovering and integrating business web services

Standard

UDDI cross industry consortium



REST

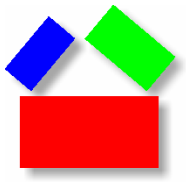
REST (representational state transfer)

Purpose

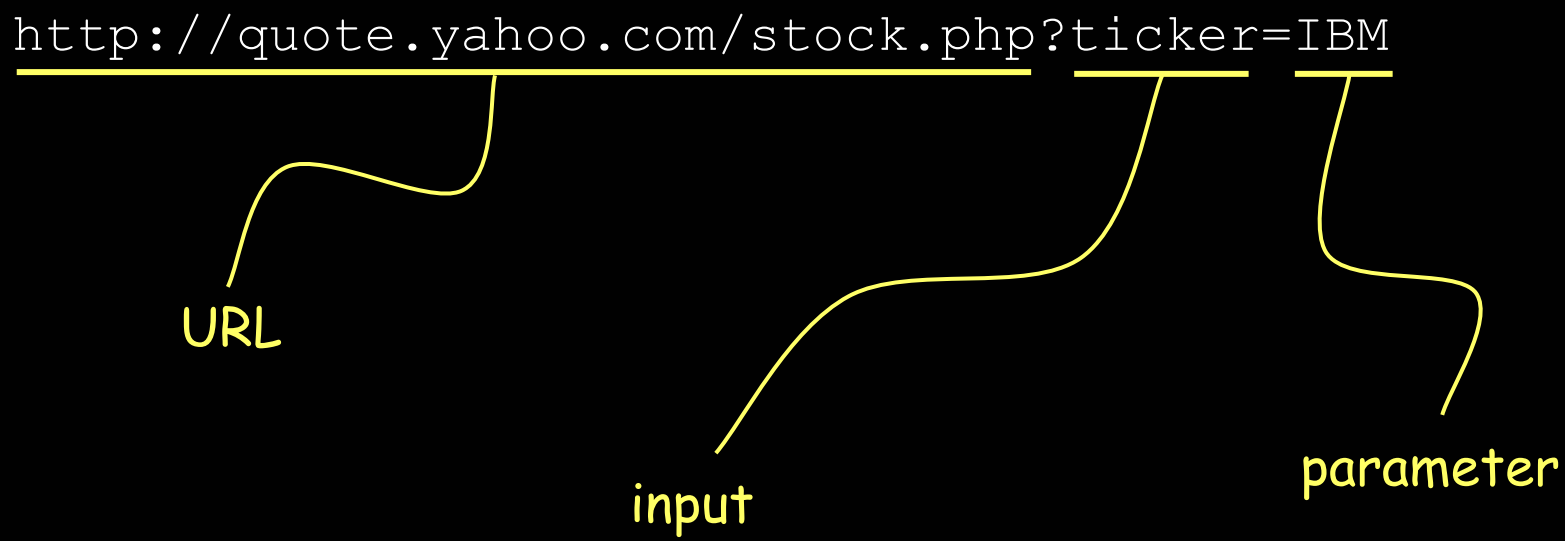
- Simplified web services description

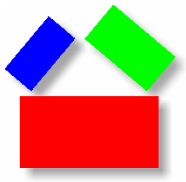
Standard

- W3C Existing standards



REST - Example





AJAX

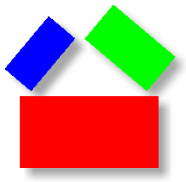
AJAX (asynchronous javascript plus XML)

Purpose

Simplified web services description

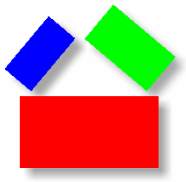
Standard

W3C Existing standards

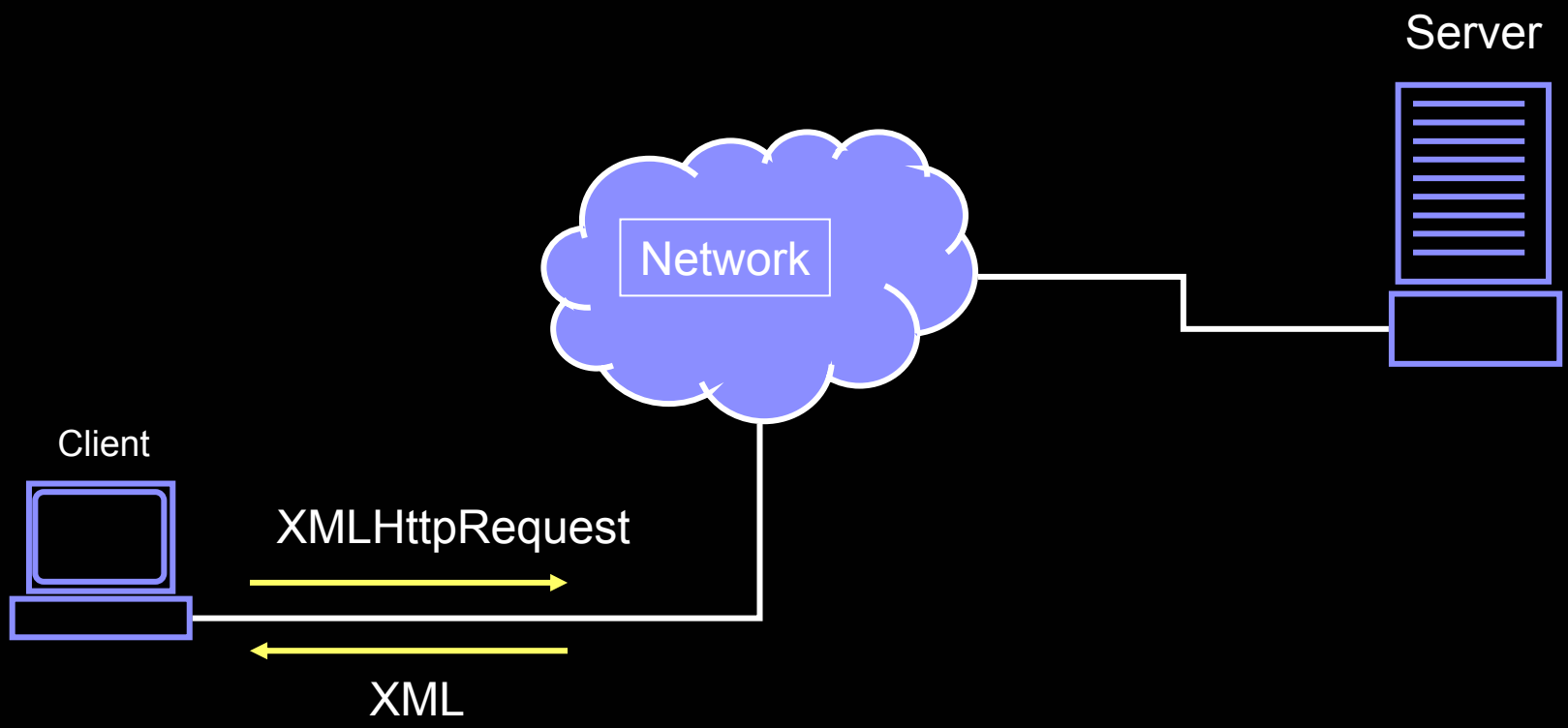


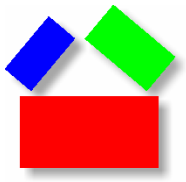
AJAX - Example

```
//-------------------------------------------------------------------------------
//  GetXML
//-------------------------------------------------------------------------------
function GetXML() {
  http = GetHTTPObject();
  http.open("GET", "http://quote.yahoo.com/stock.php?ticker=IBM", true);
  http.onreadystatechange = HandleHttpResponse;
  http.send(null);
}


//-------------------------------------------------------------
// HandleHttpResponse
//-------------------------------------------------------------
function HandleHttpResponse() {
          if (http.readyState == 4) {
                    var xmlDocument = http.responseXML;
                    var xmlObj = xmlDocument.documentElement;
                    writeXML(xmlObj, 0);
          }
}
```



AJAX





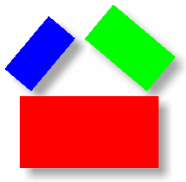
RDF (Resource Description Framework)

Purpose

- Describes web resources

Standard

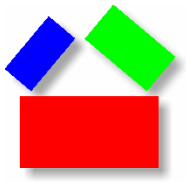
- W3C Proposal, August 1999
- W3C Recommendation, February 2004



RDF

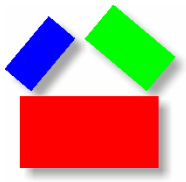
RDF – Resource, Property and Value

1. Resource – anything with a URI (e.g. http://www.example.com/RDF)
2. Property – a city (e.g. “city” or “temperature”)
3. Value – a value (e.g. “Las Vegas” or “72.0”)



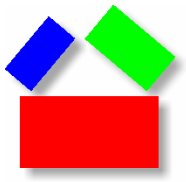
RDF – Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<RDF:rdf
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:noaa="http://www.noaa.gov/city/weather/#">

  <rdf:Description
   rdf:about="http://www.noaa.gov/city/weather/LAS">
    <noaa:city>Las Vegas</noaa:city>
    <noaa:temperature>72.0</noaa:temperature>
  </rdf:Description>
</RDF:rdf>
```



RDF Notes

- Resource, property and value form a “statement”
- Define “subject”, “predicate” and “object” of statement
- Provides groupings (group, sequence and collection)



RDFS (RDF Schema)

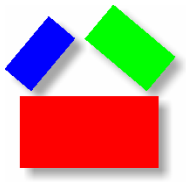
RDFS (Resource Description Framework Schema)

Purpose

- Framework for describing classes and properties

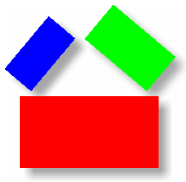
Standard

- W3C Proposal, August 1999
- W3C Recommendation, February 2004



RDFS – Example

```
<?xml version="1.0"?>

<rdf:RDF
xmlns:rdf= "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xml:base=  "http://www.example.com/animals#">

<rdf:Description rdf:ID="animal">
  <rdf:type
   rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>

<rdf:Description rdf:ID="horse">
  <rdf:type
   rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#animal"/>
</rdf:Description>

</rdf:RDF>
```



OWL (Web Ontology* Language)

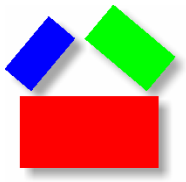
Purpose

Language for ontologies on the internet

Standard

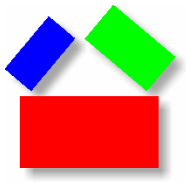
W3C Recommendation, February 2004

*An ontology is a description of things and relations.



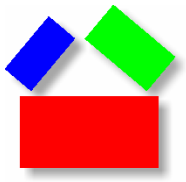
OWL Example

```
<owl:Class rdf:ID="Female">
    <rdfs:subClassOF rdf:resource="#Animal"/>
    <owl:disjointWith rdf:resource=#Male"/>
<owl:Class>
```



OWL Example

```
<owl:Class rdf:ID="Female">
    <rdfs:subClassOF rdf:resource="#Animal"/>
    <owl:disjointWith rdf:resource=#Male"/>
<owl:Class>
```



OWL

rdfs:Class	rdf:Property	rdfs:domain	rdfs:range
owl:AllDifferent	owl:allValuesFrom	owl:Restriction	rdfs:Class
owl:AnnotationProperty	owl:backwardCompatibleWith	owl:Ontology	owl:Ontology
owl:Class	owl:cardinality	owl:Restriction	xsd:nonNegativeInteger
owl:DataRange	owl:complementOf	owl:Class	owl:Class
owl:DatatypeProperty	owl:differentFrom	owl:Thing	owl:Thing
owl:DeprecatedClass	owl:disjointWith	owl:Class	owl:Class
owl:DeprecatedProperty	owl:distinctMembers	owl:AllDifferent	rdf:List
owl:FunctionalProperty	owl:equivalentClass	owl:Class	owl:Class
owl:InverseFunctionalProperty	owl:equivalentProperty	rdf:Property	rdf:Property
owl:Nothing	owl:hasValue	owl:Restriction	
owl:ObjectProperty	owl:imports	owl:Ontology	owl:Ontology
owl:Ontology	owl:incompatibleWith	owl:Ontology	owl:Ontology
owl:OntologyProperty	owl:intersectionOf	owl:Class	rdf:List
owl:Restriction	owl:inverseOf	owl:ObjectProperty	owl:ObjectProperty
owl:SymmetricProperty	owl:maxCardinality	owl:Restriction	xsd:nonNegativeInteger
owl:Thing	owl:minCardinality	owl:Restriction	xsd:nonNegativeInteger
owl:TransitiveProperty	owl:oneOf	owl:Class	rdf:List
	owl:onProperty	owl:Restriction	rdf:Property
	owl:priorVersion	owl:Ontology	owl:Ontology
	owl:sameAs	owl:Thing	owl:Thing
	owl:someValuesFrom	owl:Restriction	rdfs:Class
	owl:unionOf	owl:Class	rdf:List
	owl:versionInfo		