

XML and friends

- **history/background**
 - GML (1969)
 - SGML (1986)
 - HTML (1992)
 - World Wide Web Consortium (W3C) (1994)
- **XML** (1998)
 - core language
 - vocabularies, namespaces
XHTML, SVG, MathML, Schema, ...
 - validation
Schema, DTD
 - parsers
SAX, DOM
 - processing XML documents
XPath, XSLT
 - web services based on XML
SOAP, WSDL, UDDI, ...
- **sources** (subset of a huge number)
 - www.w3.org
 - *Professional XML, 2nd ed* (Birbeck et al, Wrox, 2001)
 - *XML in a Nutshell* (Harold & Means, O'Reilly, 2001)
 - *Building Web Services with Java* (Graham et al, Sams, 2002)
 - *Processing XML with Java* (Harold, Addison-Wesley, 2003)

Markup languages

- **"mark up" documents with human-readable tags**
 - content is separate from description of content
 - not limited to describing visual appearance
- **SGML and XML are meta-languages for markup**
 - languages for describing grammar and vocabularies of other languages
 - element: data surrounded by markup that describes it
<person>George Washington</person>
 - attribute: named value within an element
<body bgcolor="green">
 - extensible: tags & attributes can be defined as necessary
not a fixed set
 - strict rules of syntax
 - where tags appear
 - what names are legal
 - what attributes are associated with elements
 - instances are specialized to particular applications
 - HTML: tags for document presentation
 - XHTML: HTML with precise syntax rules
- **XML is compatible with SGML**
 - a simplified, inter-operable form

XML: eXtensible Markup Language

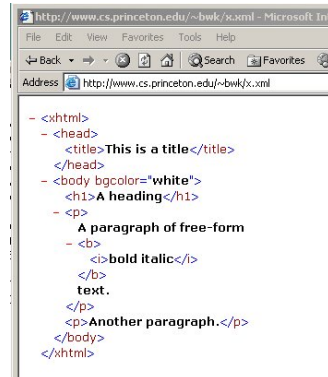
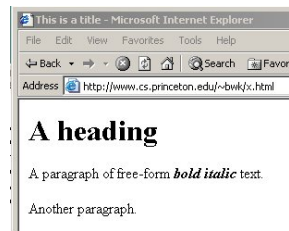
- an extensible way to describe any kind of data
- a notation for describing trees (only)
- each internal node in the tree is an element
- leaf nodes are either attributes or text
- "well formed": the instance is a tree
 - everything balanced, terminated, quoted, etc.
- "valid": satisfies stronger syntactic rules
 - as given in a DTD or schema
 - valid tags & attribs, proper order, right number, ...
- text only (Unicode), not binary
 - human-readable
 - process with standard tools
 - independent of proprietary tools and representations
- not a programming language
 - XML doesn't do anything, just describes
 - programs read, process, and write it
- not a database
 - programs convert between XML and databases

XML in use

- two common kinds of use
 - document-centric
 - ordinary text documents with markup
 - data-centric:
 - representation and exchange of data with applications
- XHTML
 - example of document-centric view
 - XHTML is HTML with more stringent rules
 - everything balanced and terminated and quoted
 - names are case sensitive

```
<xhtml>
  <head>
    <title> This is a title </title>
  </head>
  <body bgcolor="white">
    <h1> A heading </h1>
    <p> A paragraph of free-form
      <b><i>bold italic</i></b> text.
    </p>
    <p> Another paragraph. </p>
  </body>
</xhtml>
```

XML as seen by browsers



Why XML?

- **increasing use of web services**
 - too hard to extract semantics from HTML
 - closed and/or binary systems are too hard to work with, too inflexible
- **XML is open, non-proprietary**
- **text-based**
 - can see what it does
 - standard tools work on it
 - standard parsers, transformers, generators, etc.
- **simple, extensible**
 - existing vocabularies for important areas
 - can easily define new vocabularies for specific areas
- **most XML use is data-centric**
 - becoming standard medium of exchange for web services



XML vocabularies and namespaces

- a *vocabulary* is an XML description for a specific domain

- Schema
- XHTML
- CML (chemical markup language)
- NewsML (news story markup language)
- SVG (scalable vector graphics)
- MathML
- others
 - SMIL, VoiceML, patent applications, ...
 - Bball, Geoserver, XAMPL

- namespaces

- mechanism for handling name collisions between vocabularies

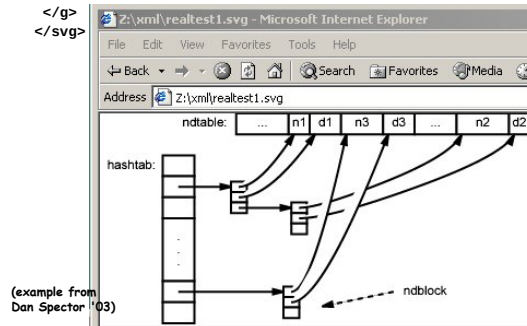
<ns:tag> ... </ns:tag>

<ns2:tag> ... </ns2:tag>

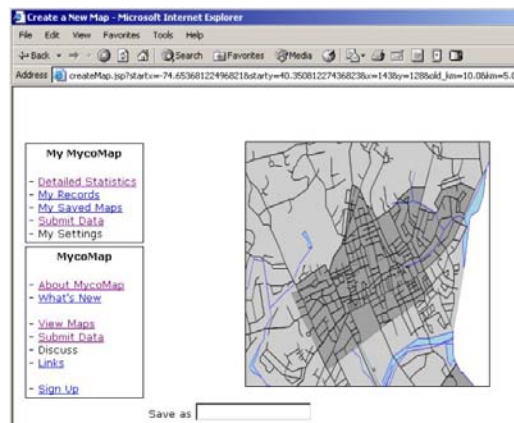
SVG: scalable vector graphics

• XML vocabulary for diagrams

```
<svg> ...  
<g font-size="120" >  
  <text x="3100" y="1700" text-anchor="middle"  
    baseline-shift="-25%">ndblock</text>  
</g>  
<line x1="2800" y1="1700" x2="2119" y2="1840"  
  stroke="black" stroke-width="20"  
  stroke-dasharray="50,50" />  
<polygon fill="black" stroke="black"  
  stroke-width="20"  
  points="2212,1795 2119,1840 2222,1844" />  
<g font-size="120" >  
  <text x="1000" y="100" text-anchor="middle"  
    baseline-shift="-25%">ndtable:</text>  
</g>  
</svg>
```



SVG for cartography



XML data-centric document

```
<?xml version="1.0" encoding="UTF-8"?>
<amazon
  xmlns:amazon="./amazon.xml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="./amazon.xml ./amazon.xsd">
  <book isbn="2468">
    <title>Algorithms in Python</title>
    <author>Sedgewick</author>
    <list>79</list>
    <sale qty="1" price="70"/>
    <sale qty="2" price="79"/>
    <sale qty="33" price="50"/>
  </book>
  <book isbn="4321">
    <title>TPOP</title>
    <author>Kernighan</author>
    <author>Pike</author>
    <list>25</list>
    <sale qty="1" price="20"/>
  </book>
  <customer id="11">
    <name>Brian</name>
    <address>Princeton, NJ</address>
  </customer>
  <customer id="22">
    <name>Bill</name>
    <address>Redmond, WA</address>
  </customer>
</amazon>
```

XML describes trees

- **"well formed": it is a valid tree structure**
 - properly nested
 - syntactically correct
 - everything properly quoted
 - nothing about semantics or relationships among elements
- **"valid": well formed AND satisfies set of rules about what is legal**
- **two mechanisms for defining validity:**
- **DTD: document type definition**
 - (comparatively) simple pattern specification
 - not very powerful (no data types)
 - not written in XML syntax (needs separate tools)
- **Schema**
 - (comparatively) complicated specification
 - much stronger language for expressing structure sequencing and counting of complex types
 - built-in basic types like integer, double, string
 - can attach validation constraints to basic types ranges of integers, patterns of strings, etc.
 - written in XML
 - can apply all XML tools to it

Example schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!--W3C Schema generated by XMLSPY v2004 rel. 3 U -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="amazon">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="book" maxOccurs="unbounded"/>
        <xs:element ref="customer" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="amazon" type="xs:string"
use="required"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="book">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="title"/>
        <xs:element ref="author" maxOccurs="unbounded"/>
        <xs:element ref="list"/>
        <xs:element ref="sale" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="isbn" use="required">
        <xs:simpleType>
          </xs:simpleType>
        </xs:attribute>
      </xs:complexType>
    </xs:element>
```

Example schema continued

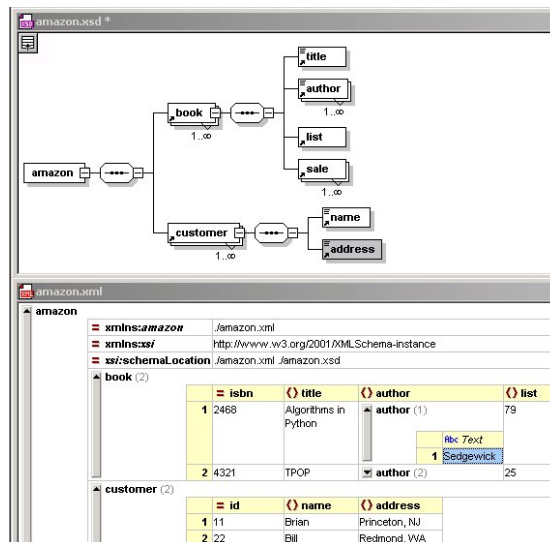
```
  <xs:element name="customer">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="name"/>
        <xs:element ref="address"/>
      </xs:sequence>
      <xs:attribute name="id" use="required"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="list"/>
  <xs:element name="name">
    <xs:simpleType>
      <xs:restriction base="xs:string"/>
    </xs:simpleType>
  </xs:element>

  <xs:element name="sale">
    <xs:complexType>
      <xs:attribute name="qty" use="required"/>
      <xs:attribute name="price" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="title" type="xs:string"/>

  <xs:element name="address">
    ...
  <xs:element name="author">
    ...
</xs:schema>
```

XML tools / XMLSpy



XML processing by program

- two basic kinds of parsers
- **DOM (Document Object Model)**
 - read entire XML document into memory
 - create a tree
 - provide methods for walking/processing the tree
- **SAX (Simple API for XML)**
 - read through XML document
 - nothing stored implicitly
 - call user-defined method for each document element
 - callbacks
- **other processing tools**
 - CSS (cascading style sheets)
 - XSLT (extensible stylesheet language for XML transformations)
 - XPath (query/filter language for XML)

DOM: document object model

- standard language-independent interface for manipulating structured documents
- allows dynamic access and modification
- methods for traversing tree and accessing nodes
 - does not define any semantics other than
 - walking the tree
 - accessing elements
 - adding or deleting elements
- implementations in Java, C++, VB, etc.

DOM reader in Java

```
import java.io.*;
import org.w3c.dom.*;
import javax.xml.parsers.*;

public class domreader {
    public static void main(String[] args) {
        domreader r = new domreader(args[0]);
    }

    public domreader(String f) {
        try {
            DocumentBuilderFactory dbf =
                DocumentBuilderFactory.newInstance();
            // dbf.setValidating(true);
            DocumentBuilder b =
                dbf.newDocumentBuilder();
            Document doc = b.parse(f);

            Element root = doc.getDocumentElement();
            print_node(root, "");

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

DOM reader, page 2

```
void print_node(Node n, String pfx) {
    if (n == null)
        return;
    if (n.getNodeType() == Node.ELEMENT_NODE) {
        Node cn = n.getFirstChild();
        String s = "";
        if (cn != null)
            s = ((CharacterData)cn).getData();
        s = s.trim();
        System.out.println(pfx +
            n.getNodeName() + " [" + s + "]");
        print_attrs(n, pfx + " ");
        print_children(n, pfx);
    }
}

void print_children(Node n, String pfx) {
    NodeList nl = n.getChildNodes();
    for (int i = 0; i < nl.getLength(); i++) {
        print_node(nl.item(i), pfx + " ");
    }
}
```

DOM reader, page 3

```
void print_attrs(Node n, String pfx) {
    NamedNodeMap nnm = n.getAttributes();
    if (nnm != null) {
        for (int j=0; j < nnm.getLength(); j++){
            System.out.println(pfx +
                nnm.item(j).getNodeName() + "="
                + nnm.item(j).getNodeValue());
        }
    }
}
```

Output of DOM code...

```
amazon []
  xmlns:amazon=./amazon.xml
  xmlns:xsi=http://www.w3.org/2001/XMLSchema-instanc
  xsi:schemaLocation=./amazon.xml ./amazon.xsd
  book []
    isbn=2468
    title [Algorithms in Python]
    author [Sedgewick]
    list [79]
    sale []
      qty=1
      price=70
    sale []
      qty=2
      price=79
    sale []
      qty=33
      price=50
  book []
    isbn=4321
    title [TPOP]
    author [Kernighan]
    author [Pike]
    list [25]
    sale []
      qty=1
      price=20
  customer []
    id=11
    name [Brian]
    address [Princeton, NJ]
  customer []
    id=22
    name [Bill]
    address [Redmond, WA]
```

SAX reader in Java

```
import java.io.*;
import org.xml.sax.*;
import org.xml.sax.helpers.*;
import javax.xml.parsers.*;

public class saxreader extends DefaultHandler
{
    public static void main(String[] args) {
        saxreader r = new saxreader(args[0]);
    }

    public saxreader(String f) {
        try {
            SAXParserFactory spf =
                SAXParserFactory.newInstance();
            SAXParser sp = spf.newSAXParser();

            sp.parse(new File(f), this);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

SAX reader, page 2

```
public void startDocument() {
    System.out.println("startDoc");
}

public void endDocument() {
    System.out.println("endDoc");
}

public void startElement(String nsURI,
    String localname, String qualname,
    Attributes attr) {
    if (localname.equals(""))
        localname = qualname;
    System.out.println("startElement " +
        localname);
    if (attr != null) {
        for (int i=0; i<attr.getLength(); i++) {
            String s = attr.getLocalName(i);
            if (s.equals(""))
                s = attr.getQName(i);
            System.out.println(s + "=" +
                attr.getValue(i));
        }
    }
}
```

SAX reader, page 3

```
public void endElement(String nsURI
    String localname, String qualname) {
    if (localname.equals(""))
        localname = qualname;
    System.out.println("endElement "
        + localname);
}

public void characters(char buf[],
    int offset, int len) {
    System.out.println("chars [" +
        new String(buf, offset, len) + "]);
}
```

Output of SAX reader

```
startDoc
startElement amazon
xmlns:amazon=../amazon.xml
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:schemaLocation=../amazon.xml ../amazon.xsd
chars []
chars [
]
chars [ ]
startElement book
isbn=2468
chars []
chars [
]
chars [ ]
startElement title
chars [Algorithms in Python]
endElement title
chars []
chars [
]
chars [ ]
startElement author
chars [Sedgewick]
endElement author
chars []
chars [
]
chars [ ]
startElement list
chars [79]
endElement list
chars []
chars [
]
chars [ ]
startElement sale
qty=1
price=70
endElement sale
```

DOM reader in VB inside Excel

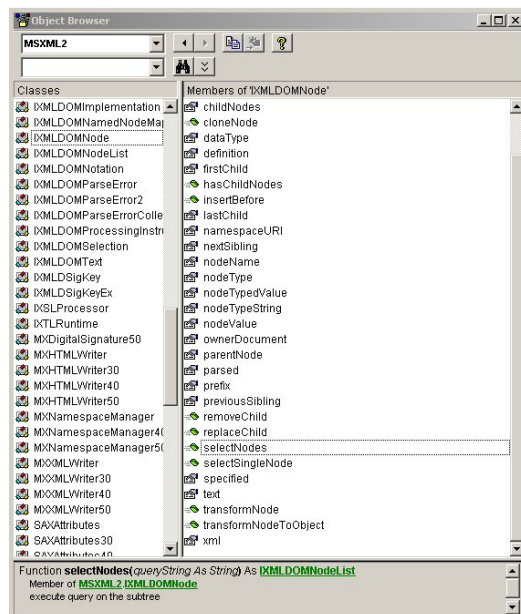
```
Dim xmlDoc As MSXML2.DOMDocument50

Sub Startit(r As Range)
    ...
    xmlfile = "z:\cs333\XML\class.xml"
    Set xmlDoc =
        CreateObject("Msxml2.DOMDocument.5.0")
    xmlDoc.async = False
    xmlDoc.validateOnParse = False
    If xmlDoc.load(xmlfile) <> True Then
        MsgBox "load is false"
    If xmlDoc.parseError.errorcode = 0 Then
        Call tree_walk(xmlDoc)
    End If
End Sub

Sub attribute_walk(node As IXMLDOMNode)
    Dim attrib As Variant
    For Each attrib In node.attributes
        ro = ro + 1
        rng.Offset(ro, co) = attrib.nodeTypeString
        rng.Offset(ro, co + 1) = attrib.nodeName
        rng.Offset(ro, co + 2) = attrib.nodeValue
    Next
End Sub
```

	A	B	C	D	E	F	G	H
1								
2		processingInstruction						
3		element	amazon	Algorithms in Python Sedgewick	79	TPOP	Kernighan Pike	25
4		attribute	xmlns:amazon	/amazon.xml				
5		attribute	xmlns:xsl	http://www.w3.org/2001/XMLSchema-instance				
6		attribute	xsl:schemaLoc	/amazon.xml	/amazon.xsd			
7		element	book	Algorithms in Python Sedgewick	79			
8		attribute	isbn	2468				
9		element	title	Algorithms in Python				
10			text					
11		element	author	Sedgewick				
12		element	list					
13		element	text	79				
14		element	sale					
15			attribute	qty				1
16			attribute	price				70
17		element	sale					
18			attribute	qty				2
19			attribute	price				79
20		element	sale					
21			attribute	qty				33
22			attribute	price				50
23		element	book	TPOP	Kernighan Pike	25		
24		attribute	isbn	4321				
25		element	title	TPOP				
26			text					
27		element	author	Kernighan				
28		element	text					
29		element	author	Pike				
30			text					
31		element	list					25
32			text					
33		element	sale					
34			attribute	qty				1
35			attribute	price				20
36		element	customer	Brian Princeton, NJ				
37		attribute	id					11
38		element	name	Brian				
39			text					
40		element	address	Princeton, NJ				
41			text					
42		element	customer	Bill Redmond, WA				
43		attribute	id					22
44								

MSXML objects



Web services

- **Web service is**
 - interface that describes set of operations
 - that are accessible by network
 - using XML and standard protocols
- **SOAP**
 - simple object access protocol
- **WSDL**
 - web services description language
- **UDDI**
 - universal description, discovery and integration

SOAP

- "an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses." (W3C)
- **communication protocol for invoking methods on servers, services, components and objects**
 - language independent "wire protocol"
 - COM, CORBA, etc., can use it
- **XML vocabulary for defining parameters, return values, and exceptions**
- **uses HTTP to carry info**
 - interface & method names included in header
 - supposed to be checked by recipient
- **formalizes use of XML and HTTP for invoking remote methods**
- **open standard, widely supported**

SOAP Request

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV=
    "http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle=
    "http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:GetLastTradePrice xmlns:m="Some-URI">
      <symbol>DIS</symbol>
    </m:GetLastTradePrice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV=
    "http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle=
    "http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:GetLastTradePriceResponse
      xmlns:m="Some-URI">
      <Price>34.5</Price>
    </m:GetLastTradePriceResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

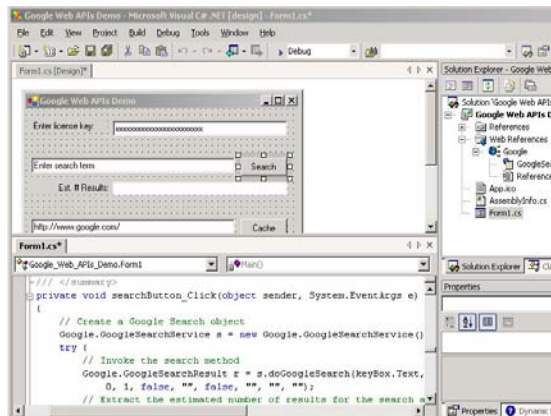

Google SOAP envelope

```
<?xml version='1.0' encoding='UTF-8'?>

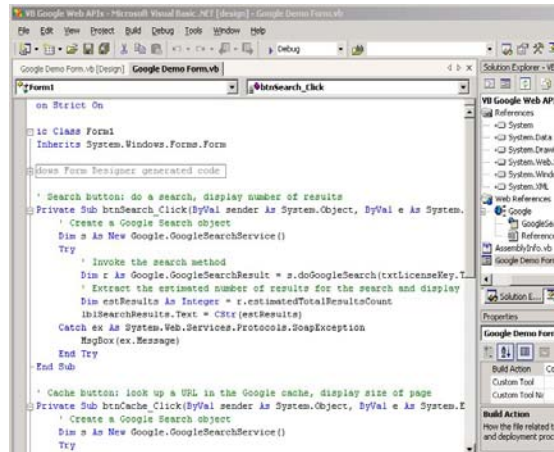
<SOAP-ENV:Envelope xmlns:SOAP-ENV=
    "http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org
    /1999/XMLSchema">
  <SOAP-ENV:Body>
    <ns1:doGoogleSearch xmlns:ns1="urn:GoogleSearch"
      SOAP-ENV:encodingStyle=
        "http://schemas.xmlsoap.org/soap/encoding/">
      <key xsi:type="xsd:string">
        00000000000000000000000000000000</key>
      <q xsi:type="xsd:string">whatever</q>
      <start xsi:type="xsd:int">0</start>
      <maxResults xsi:type="xsd:int">10</maxResults>
      <filter xsi:type="xsd:boolean">true</filter>
      <restrict xsi:type="xsd:string"></restrict>
      <safeSearch xsi:type="xsd:boolean">>false</safeSearch>
      <lr xsi:type="xsd:string"></lr>
      <ie xsi:type="xsd:string">latin1</ie>
      <oe xsi:type="xsd:string">latin1</oe>
    </ns1:doGoogleSearch>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Web services

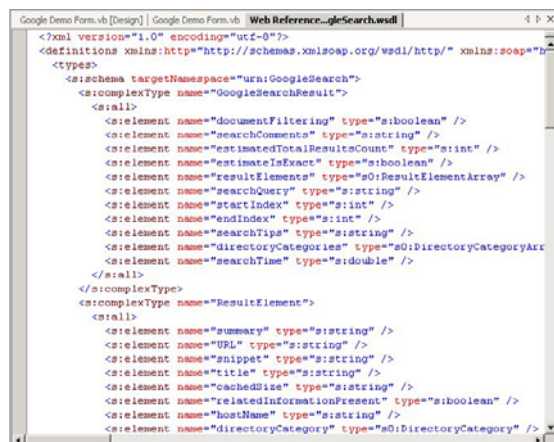
- interfaces from C# and VB (and others)
- analogous to Java interfaces



VB-based client for google api



WSDL for google api



Web Service — Hello, world (server)

```
<%@ WebService Language="C#"
    Class="GreetingService" %>

using System;
using System.Web.Services;

class GreetingService : WebService {
    [WebMethod]
    public string GetGreeting() {
        return "Hello, world";
    }
}
```