Languages and Tools for Web Programming

Uri Dekel

ISRI, Carnegie Mellon University

Presented in UI course

Some examples taken from w3schools.com

Outline

- Part 1: Static document formats for the web
  - Document forms: HTML and CSS
  - Data forms: XML, DTDs and Schemas, XSL
  - High-end graphics forms: VRML, SVG
- Part 2: Client-side interactive web pages
  - Client-Side Scripting languages: JavaScript, VBScript
  - Client-Side embedded applications: Java applets, ActiveX, Flash
- Part 3: Server-side web page creation
  - Scripting languages: CGI and Perl, PHP, ColdFusion
  - High-level frameworks: Servlets and JSP, ASP, ASP.NET
- Part 4: Web service architectures
  - WSDL, SOAP

Document Formats: The evolution of HTML

HTML

- HyperText Markup Language
- Primary document type for the web
  - Transmitted using the HyperText Transfer Protocol
    - Client sends request string (with parameters)
    - Server returns a document
      - **Stateless protocol**
- Describes document content and structure
  - Precise formatting directives added later
  - Content and structure in same document
- Browser or formatter responsible for rendering
  - Can partially render malformed documents
  - Different browsers render differently

HTML structure

- HTML document is a text based representation of a tree of tags
    - General structure:

```
<OUTERTAG attribute1=‘val1’ attribute2=‘val2’>
<INNERTAG attribute3=‘val3’>some text</INNERTAG>
</OUTERTAG>
```

HTML evolution

- HTML 1 [early ‘90s]
  - Invented by Tim Berners-Lee of CERN
    - Aimed as standard format to faciliate collaboration between physicists
  - Based on the SGML framework
    - Old ISO standard for structuring documents
      - Tags for paragraphs, headings, lists, etc.
    - HTML added the hyperlinks, thus creating the web
  - Rendered on prototype formatters

HTML evolution

- HTML+ [mid '94]
  - Defined by small group of researchers
  - Several new tags
    - Most notably, IMG for embedding images
  - Many browsers
    - First text-based browser (Lynx) released in 03/93
    - First graphical browser (Mosaic) released in 04/93
- First W3 conference [5/94]
  - HTML+ presented

HTML evolution

- HTML 2 [7/94-11/95]
  - Prompted by variety of diverging language variants and additions of different browsers
  - Adds many widely used tags
    - e.g., forms
  - No custom style support
    - e.g., no colors
- W3 consortium formed [Late 94]
  - Mission: Open standards for the web

HTML evolution

- Netscape formed [11/94]
  - Becomes immediate market leader
    - Support for home users
  - Forms a de-facto standard
    - Use of “Netscape proprietary tags”
      - Difficult for other browsers to replicate
      - Documents start rendering differently
    - **Addition of stylistic tags**
      - e.g., font color and size, backgrounds, image alignment
      - Frowned upon by structure-only advocates

HTML evolution

- HTML 3.0 draft proposed
  - Huge language overhaul
    - Tables, math, footnotes
    - Support for style sheets (discussed later)
  - Too difficult for browsers to adapt
    - Every browser implemented different subset
      - But claimed to support the standard
        - And added new tags…
    - Standard abandoned
      - Incremental changes from here on

HTML evolution

- Microsoft introduces Internet explorer [8/95]
  - First serious competition to Netscape
  - Starts introducing its own tags
    - e.g., `MARQUEE`
    - Effectively splitting web sites into Microsoft and Netscape pages
      - Many sites have two versions
- Microsoft starts supporting interactive application embedding with ActiveX
  - Netscape responds with the emerging Java technology
  - Starts supporting JavaScript
    - Microsoft introduces VBScript

HTML evolution

- HTML 3.2 [1/97]
  - Implements some of the HTML 3.0 proposals
    - Essentially catches up with some widespread features.
  - Supports applets
  - Placeholders for scripting and stylesheet support

HTML evolution

- HTML 4 [12/97]
  - Major overhaul
    - Stylesheet support
    - Tag identifier attribute
    - Internationalization and bidirectional text
    - Accessibility
    - Frames and inline frames
    - `<object>` tag for multimedia and embedded objects
  - Adapted by IE (market leader)
    - Slow adaptation by Netscape
- XML 1.0 standard [2/98]
- XHTML 1.0 [1/00, 8/02]

Limitations of HTML

- No support for accessibility until HTML 4
- No support for internationalization until HTML 4
- No dynamic content in original definition
- No inherent support for different display configurations (e.g., grayscale screen)
  - Except for `alt` tag for images
  - **Added in CSS2**
- **<u>No separation of data, structure and formatting</u>**
  - **<u>Until version 4</u>**

Wireless Markup Language (WML)

- Markup language for WAP browsers
  - WAP = Wireless Application Protocol
  - Based on limited HTML, uses XML notation
  - Uses WMLScript scripting language, based on JavaScript
- A page is called a “deck”, displayed in individual sections called “cards”
  - Tasks are used to perform events
  - Variables used to maintain state between cards

Client-side: Cascading Style Sheets

Why CSS?

- HTML was not meant to support styling information
  - But browsers started supporting inline style changes to make web look better
- Inline styling information is problematic
  - Difficult to change
  - Lack of consistency
  - No support for different display formats
  - Bloats pages
  - No support for some styling features

Connecting HTML to CSS

- HTML document typically refers to external style sheet

```
<HEAD>
  <LINK rel="stylesheet" type="text/css“
  href="fluorescent.css">
</HEAD>
```

- Style sheets can be embedded:

```
<HEAD><STYLE type="text/css">
    <!-- …CSS DEFINITIONS.. -->
</STYLE></HEAD>
```

Connecting HTML to CSS

- Styles can be embedded inline with the *style* attribute
- Style sheets can be chosen by media type
  - Simply add a *media* attribute to the *link* or *style tags*
  - Choose from: screen, tty, tv, projection, handheld, braille, aural, all
- HTML document can provide several stylesheet options
  - Give titles to each stylesheet
  - One preferred (default) style, the rest are alternates
    - e.g., http://www.w3.org/Style/Examples/007/alternatives.html
- Default configuration in internal browser stylesheet and user stylesheet

Style sheet structure

- Declaration gives value to property
  - Property: value
    - e.g., `color: red`
- Styles are applied to selectors
  - Selector describes element
    - Simplest form is tag type
    - e.g., `P {color:red; font-size: 16px}`
- Style sheet is a series of style applications
  - Can import other stylesheets

    -
      ```
      @import url(corestyles.css);
      BODY {color: red; background-color: black}
      ```
- Style of enclosing element is inherited by enclosed

Selectors

- Type selectors
  - Name of HTML elements
- Pseudo-class
  - Specific subset of an HTML elements
    - e.g., `:link`, `:visited`, `:active` for the `A` tag
- Pseudo-element
  - Specific subset of any element
    - e.g., `:first-line`, `:first-letter`
- Context sensitive elements
  - e.g., `H2 I {color:green}`

Selectors

- Element classes
  - HTML document can classify tags
    - e.g., `<P class=“warning”>…</P>`
  - Can refer to element type with specific class
    - e.g., `P.warning {color:red}`
  - Can refer to all elements with specific class
    - e.g., `.warning {color:red}`
  - Use HTML tags `<div>` and `<span>`
- Element IDs
  - HTML entity can have a unique id attribute
    - e.g., `<P id=“copyright”>…</P>`
      `#copyright {color:blue}`

Cascading

- Most properties are inherited
  - From enclosing element to internal element
- Sort order for conflict resolution:
  - Origin (page>user>browser)
  - Weight (!`important` symbol allows overriding)
  - Specificity
  - Order

How is CSS applied?

1. Source document is parsed into a DOM tree
2. Media type is identified
3. Relevant stylesheets obtained
4. DOM tree annotated with values to every property
5. Formatting structure generated
6. Formatting structure presented (rendered)

CSS2

- Extends CSS1
  - Many new properties and built-in classes
  - Better support for media types
    - Stylesheet can specify type in selector
  - Better support for accessibility
    - Properties for aural rendering
  - Better support for internationalization

Document Formats: XML

XML, SAX, DOM, DTD, XML-SCHEMA, XSL, XMLFO

XML

- Extensible Markup Language
  - Based on SGML format
  - Intended to facilitate data exchange
- Documents consist of tags and data
  - Data is usually untyped characters
  - Tags have names and attributes
- Document has tree structure
  - Tags are nested
  - Data areas are considered leafs
  - One root

```
<?xml version="1.0"?>
<person>
    <name type=“full”>John Doe</name>
    <tel type=“home”>412-555-4444</tel>
    <tel type=“work”>412-268-5555</tel>
    <email>johndoe@anon.net</email>
</person>
```

XML Structure

- XML documents have no semantics
    - It is up to the programs using them
- XML does not enforce structure
    - No restriction on possible tags
    - No restriction on order, repeats, etc.
    - Mixed content allowed
        - Text followed by tags followed by text
        - Allows HTML compatibility (XHTML)
- “Well-Formed Document”
    - Tree structure with proper nesting
    - Attributes are not repeated in same tag

XML Programming with SAX

- Lightweight simple event-based parser
  - Originally in Java, ports for other languages
- Programmer instantiates SAX parser
- Parser is invoked on input and an implementation of Document Handler
  - Parser invokes callback functions on handler during DFS traversal
  - e.g., `startDocument`, `endDocument`, `startElement`, `endElement`, `etc.`

XML Programming with DOM

- A heavyweight XML-based API
  - Supported in multiple languages
- A programmatic representation of the XML document tree
  - Variety of interfaces representing elements, attributes, etc.
- User instantiates a DOM parser of specific vendor and supplies XML file
  - Receives `Document` interface
  - Different parsers use different optimizations

DTD

- Document Type Descriptor
  - Impose structure on XML document
  - Usually placed in separate file
    - XML refers to HTML file using following header:
      - `<!DOCTYPE root-element SYSTEM "filename">`
    - DTD can be placed inline
- An XML document is Valid if it conforms to the DTD
- DTD consists of a series of declarations

DTD Element Declarations

- `<!ELEMENT element-name category>`
  - Category can be:
    - `ANY`
    - `(#PCDATA)`
      - Text… Element becomes leaf
    - `EMPTY`
      - No tags or text can be nested
    - Sequence of nested elements
      - Essentially a regular expression
      - e.g., `<!ELEMENT note (to+,from,cc*,subject?, header,(message|body))>`
- `<!ENTITY entity-name "entity-value">`
  - Declares a symbolic constant

DTD Attribute Declaration

- `<!ATTLIST element-name attribute-name attribute-type default-value>`
- Attribute types include:
  - `CDATA` for text
  - `(en1|en2|en3…)` for enumeration
  - `ID` for unique element identifiers
  - `IDREF` for referring to other elements
    - Must refer to existing IDs
- Default value can be:
  - String for actual default value
  - `#REQUIRED` for forcing user to specify value
  - `#IMPLIED` for optional attributes
  - `#FIXED` for constant

Limitations of DTD

- DTD is weaker than database schemas
  - Only one type
    - Writer and reader must agree on implied types
  - No abstractions such as sets
  - ID References are untyped
  - No constraints
  - Tag definitions are global
- XML-Schema provides these capabilities
  - Important for e-commerce

XML-Schema

- Replacement for DTDs
- Written in XML
  - More extensible to future additions
- Support built-in and user-defined data types
  - Including typed references and complex data types
- Support constraints

XML-Schema Example

- Schema document:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs=“…“ targetNamespace=“…“ xmlns=“…"
  elementFormDefault="qualified">
<xs:element name=“person">
  <xs:complexType><xs:sequence>
      <xs:element name=“name" type="xs:string"/>
      <xs:element name=“tel" type="xs:string"/>
      <xs:element name=“email" type="xs:string"/>
  </xs:sequence> </xs:complexType>
</xs:element> </xs:schema>
```

XML-Schema

- <xs:schema> header has following attributes:
  - Namespace for XML Schema built-in tags
    - xmlns:xs="http://www.w3.org/2001/XMLSchema"
  - Namespace for elements defined by schema
    - targetNamespace=“http://www.uridekel.com”
  - Default namespace
    - xmlns=“http://www.uridekel.com”
  - Whether documents must use qualified names
    - elementFormDefault="qualified"
- XML file refers to schema :
  - <note xmlns="http://www.uridekel.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation=http://www.uridekel.com/pers.xsd>

XML-Schema: Defining simple elements and attributes

- Defining a simple element
  - `<xs:element name="xxx" type="yyy"/>`
  - Common built-in types are `xs:string, xs:decimal, xs:integer, xs:boolean xs:date, xs:time`
  - Default and fixed attributes for values
- `<xs:attribute name="xxx" type="yyy"/>`
  - Default and fixed attributes for values
  - Add `use=“optional”` or `use=“required”`

XML-Schema: Restricting values

- Nest `<xs:simpleType>` and `<xs:restriction base=“xs:type">` inside the element or attribute definition
  - Simple type can be named for reuse
- Further nest the following restrictions:
  - `<xs:minInclusive>` and `<xs:maxInclusive>`
  - A sequence of `<xs:enumeration value=“val”>`
  - A regexp: `<xs:pattern value=“regexp"/>`
  - Whitespace: `<xs:whiteSpace value=“mode"/>`
  - `<xs:minLength>` and `<xs:maxLength>`
  - Many others

XML-Schema: Defining complex elements

- Create a new type with `<xs:complexType>`
- Extend an existing type by nesting `<xs:complexContent>` and `<xs:extension base="personinfo">`
- Specify child ordering with the following tags:
  - `<xs:all>` – Each child appears exactly once, but can permutate
  - `<xs:choice>` – Exactly one of the children will occur
  - `<xs:sequence>` – Each child appears exactly once, in order
- Specify child recurrence with `minOccurs` and `maxOccurs`
- Elements can be grouped with `<xs:group name=“groupname”>`
- Attributes group with `<xs:attributeGroup>`

XSL

- Extensible Stylesheet Language
  - Intended to assist in presenting XML data
    - CSS is not enough because it refers to HTML tags that have some display semantics
  - Responsible for transforming an XML document into an XHTML document
    - Essentially a tree transformation
- Consists of three languages:
  - XSLT for transforming XML documents
  - XPath for defining parts of XML documents
  - XSL-FO for formatting the elements

XPath

- A system for referring to XML tree elements
  - Used in XSLT for matching templates
- Similar to directory structure
  - Absolute paths start with `/`
  - Relative paths starts start with child name
  - Parent is selected with `..`
  - Ignore ancestors with `//`
    - e.g., `//cd` selects all cd elements
- Variety of special functions

XSLT

- Conditional Selection
  - e.g., `/catalog/cd[price>10.80]`
- Wildcard Selection
  - e.g., `/catalog/*/price`
- Selection of specific child
  - e.g., `/catalog/cd[1]`
  - e.g., `/catalog/cd[last()]`
- Referencing attributes
  - e.g., `//cd[@country='UK']`

XSLT

- XSLT is used to recursively transform a tree
  - XSL sheet consists of templates
    - Matching condition
    - Transformation
  - Transformation of the source tree is a recursive traversal
    - No recursive search on matched nodes
      - Use `<xsl:apply-templates>` to force
      - Add `select` attribute to apply to a subset
  - If match found, transformation is applied to matching part
  - in result document
    - Transformation can query nodes in the subtree
    - Nonmatching parts are unmodified in result document

XSLT Example

- **XML Document**

```
<?xml version="1.0"
  encoding="ISO-8859-1"?>
<?xml-stylesheet
type="text/xsl‟
  href="cdcatalog.xsl"?>
<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>
```

Stylesheet

```
<?xml version="1.0" encoding="ISO-8859-
1"?> <xsl:stylesheet version="1.0‟
xmlns:xsl="http://www.w3.org/1999/XSL/Tran
sform">
<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:stylesheet>
```

XSLT Structure

- Every `<xsl:template>` element attempts to match a set of XML nodes.
  - The `match` attribute associates the template with particular nodes
- The `<xsl:value-of>` element extracts data from the source node
  - The `select` attribute specifies what to extract, relative to the node matched by the template
- The `<xsl:for-each>` element enables iteration over a specific subset of nodes
  - Selection can be filtered
  - e.g., `<xsl:for-each select="catalog/cd[artist='Bob Dylan']">`
- Nodes can traversformed in a sorted order with `<xsl:sort>`
  - e.g., `<xsl:sort select="artist"/>`

XSLT Structure

- Use `<xsl:if>` for simple conditional on output:
  - `<xsl:if test=“test“>…output…</xsl:if>`
- Use `<xsl:choose>` for more complex conditionals

  -
    ```
    <xsl:choose>
       <xsl:when test=“test1">
          ... some code ...
       </xsl:when>
       <xsl:when test=“test2">
          ... some code ...
       </xsl:when>
       <xsl:otherwise>
          ... some code ....
       </xsl:otherwise>
    </xsl:choose>
    ```

Activating XSL

- Include <?xml-stylesheet directive in XML
  - XML can be displayed in browser
  - Couples data and presentation
- Use offline XSLT transformator
  - Typically useful for data processing
- Programmatically perform transformation in HTML file using scripting

```
<html><body>
<script type="text/javascript">
xml = new ActiveXObject("Microsoft.XMLDOM")
xml.load("cdcatalog.xml")
var xsl = new ActiveXObject("Microsoft.XMLDOM")
xsl.load("cdcatalog.xsl")
document.write(xml.transformNode(xsl))

</script> </body> </html>
```

XML-FO

- Extensible Stylesheet Language Formatting Objects
  - A W3C language for formatting XML data
  - Now part of the XSL standard, a target language for transformed documents
- Supports a variety of output targets
- Output is in “pages”
  - Further separated into rectangular areas

XQuery

- A standard for SQL-like queries on XML data
  - Still at the W3C draft stage
  - Relies on XPath and uses its data model
- Supports simple queries:
- e.g., `doc("books.xml")/bib/book[price<50]`
- Supports complex queries with FLWOR:
  - e.g., `for $x in doc("books.xml")/bib/book where $x/price>50 order by $x/title return $x/title`

XForms

- A new infrastructure for web forms
- Separates form functionality from presentation
- Single XML form definition model
- Form data maintained as XML instance data
  - Supports suspend and resume
- XForms UI replaces XHTML form controls
- Proprietary UIs provide alternative presentation
- Extensible for new form elements and widgets

Client Side: Scripting Languages

JavaScript, VBScript, DHTML

JavaScript

- The most common scripting language
  - Originally supported by Netscape, eventually by IE
- Typically embedded in HTML page
  - Executable computer code within the HTML content
  - Interpreted at runtime on the client side
- Can be used to dynamically manipulate an HTML document
  - Has access to the document’s object model
  - Can react to events
  - Can be used to dynamically place data in the first place
  - Often used to validate form data
- Weak typing

JavaScript Syntax

- Code written within <script> element
  - e.g.,
    ```
    <script type="text/javascript">
      document.write("Hello World!")
    </script>
    ```
  - Use `src` attribute for scripts in external files
- Placement determines execution time
  - Scripts in header must be invoked explicitly
    - e.g., during events
  - Scripts in body executed when that part is being processed.

JavaScript Syntax

- User can declare variables
  - e.g., `var name = “user”;`
  - Variables can be global to the page
- User can declare functions
  - `function func(`*`argument1,argument2,…`*`) { `*`some statements`*` }`
  - Function can return values with `return`
- Standard conditionals
  - `if..then..else, switch, ?: operator`
- Standard loops
  - `while, do..while, for`

JavaScript Syntax

- JavaScript has built-in “Object” types
  - Variety of operators and built-in functions
  - Arrays, Booleans, Dates, Math, Strings
- Direct access to the HTML DOM model
- HTML Elements have script-specific event attributes
  - e.g., `<body onmousedown="whichButton()">`
  - e.g., `<input type="button" onclick="uncheck()" value="Uncheck Checkbox">`

VBScript

- Microsoft’s answer to JavaScript
  - Never been supported by Netscape
  - Less in use now
- Use `<script type="text/vbscript">`
- Similar to JavaScript
  - Follows Visual Basic look and feel
  - Possible to declare variables
    - Use “option explicit” to force declaration
  - Separates procedures and functions

DHTML

- DHTML is a marketing buzzword
  - It is not a W3C standard
  - Every browser supports different flavour
  - It is HTML 4 + CSS stylesheets + scripting language with access to document model