Quick and Dirty XML Intro

P. Conrad for CISC474

4/19/2005

Not intended to be a complete introduction;
This is just to "get you started".

Sources:

- [1] Schaum's Easy Outlines: XML, Ed Tittel, McGraw-Hill, 2004. ($8.95 at Liebemann's).
- [2] http://www.w3.org/TR/xmlschema-1
- [3] http://www.w3.org/TR/xmlschema-2
- [4] For the data itself: Google searches on "NASCAR Driver Numbers" http://www.nascar.com/drivers/list/cup/dps/ and "New York Times Bestsellers" http://www.nytimes.com/2005/04/24/books/bestseller/0424besthardnonfiction.html
- [5] http://www.w3schools.com/xlink/xlink_intro.asp For info on XPointer and XLink.

XML provides a way to structure data

NASCAR data

```
<driver>
   <name>Jeff Gordon</name>
   <number>24</number>
   <make>Chevrolet</make>
   <sponsor>DuPont</sponsor>
</driver>

<driver>
   <name>Dale Earnhardt, Jr.</name>
   <number>8</number>
  <make>Chevrolet</make>
  <sponsor>Budweiser</sponsor>
</driver>
```

Book Data

```
<book>
   <title>The World Is Flat<title>
   <author>Thomas L. Friedman</author>
   <publisher>Farrar, Straus & Giroux</publisher>
</book>

<book>
   <title>Blink</title>
   <author>Malcolm Gladwell</author>
   <publisher>Little, Brown</publisher>
</book>
```

The structure is application-specific; you develop it yourself

If you are programming a system for NASCAR drivers, you decide what the *elements* should be through your knowledge of NASCAR:

A <name> element:

```
<name>Jeff Gordon</name>
```

A <make> element:

```
<make>Chevrolet</make>
```

```
<driver>
   <name>Jeff Gordon</name>
   <number>24</number>
   <make>Chevrolet</make>
   <sponsor>DuPont</sponsor>
</driver>

<driver>
   <name>Dale Earnhardt, Jr.</name>
   <number>8</number>
  <make>Chevrolet</make>
  <sponsor>Budweiser</sponsor>
</driver>
```

Elements and attributes

Elements:

The book element include the opening <book> tag, the closing </book> tag, and everything in between

The **author** element is nested inside the book element

```
<book>
   <title>My Life So Far</title>
   <author>Jane Fonda</author>
   <publisher>Random House</publisher>
</book>
```

Attributes:

name,value pairs inside the open tag
rookie, position and positionLastWeek are attributes

```
<driver rookie="true">
   <name>Kyle Busch</name>
   <number>5</number>
   <make>Chevrolet</make>
   <sponsor>Kellogg's/Delphi</sponsor>
</driver>
```

```
<book position="3" positionLastWeek="1">
  <title>Blink</title>
  <author>Malcolm Gladwell</author>
  <publisher>Little, Brown</publisher>
</book>
```

You specify the proper nesting of elements in one of two ways:

```
<driver>
   <name>Jeff Gordon</name>
   <number>24</number>
   <make>Chevrolet</make>
   <sponsor>DuPont</sponsor>
</driver>
```

DTD (Document Type Definition)

- The old school way
- Syntax: cryptic, compact, limited

👎 👍 👎

```
<!ELEMENT driver (name, number, make, sponsor) >
<!ELEMENT name (#PCDATA)>
<!ELEMENT number (#PCDATA)>
<!ELEMENT make (#PCDATA)>
<!ELEMENT sponsor (#PCDATA)>
```

XML Schema

- A newer way
- Syntax: XML (familiar), verbose, powerful

👍 👎 👍

```
<schema xmlns="http://www.w3.org/2001/XMLSchema”
  xmlns:nascar="http://copland.udel.edu/~pconrad/xmlns/nascar"
  targetNamespace="http://copland.udel.edu/~pconrad/xmlns/nascar"
<element name="driver">
  <complexType>
     <sequence>
      <element ref="nascar:name"/>
      <element ref="nascar:number"/>
      <element ref="nascar:make"/>
     <element ref="nascar:sponsor"/>
    </sequence>
   </complexType>
</element>
<element name="name" type="string" />
<element name="number" type=" nonNegativeInteger" />
<element name="make" type="string" />
<element name="sponsor" type="string" />
</schema>
```

Validation of XML documents

- "Without a DTD or a schema, there is no way to validate a document. Validation means conformity to a schema or DTD". [1, p. 26].

DTD Types:

for elements:

#PCDATA
(parsed character data)

for attributes:

ID
IDREF
CDATA
NOTATION
ENTITY
ENTITITES

Schema Types
(for both elements and attributes)

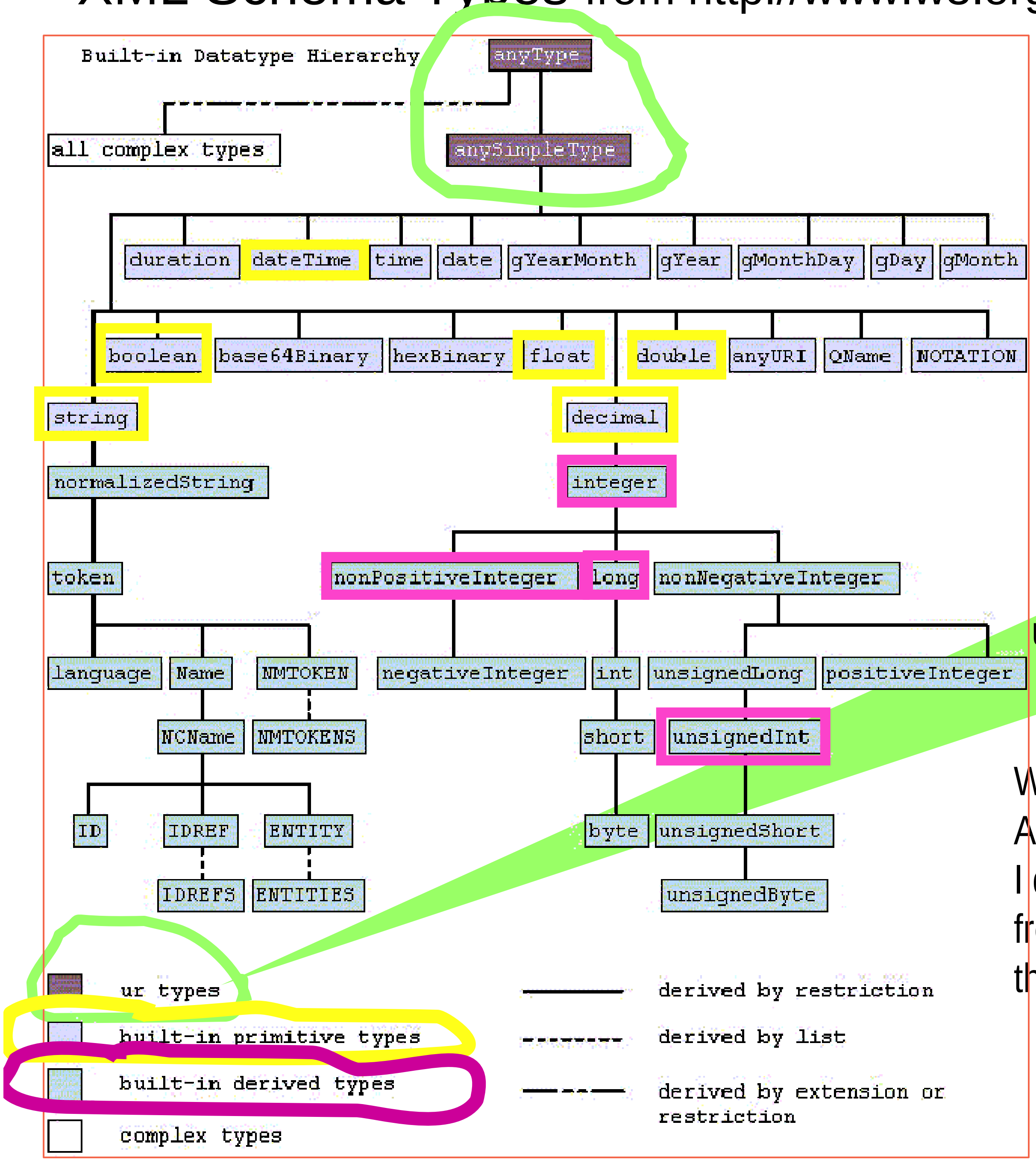
3.2 Primitive datatypes

3.2.1 string
3.2.2 boolean
3.2.3 decimal
3.2.4 float
3.2.5 double
3.2.6 duration
3.2.7 dateTime
3.2.8 time
3.2.9 date
3.2.10 gYearMonth
3.2.11 gYear
3.2.12 gMonthDay
3.2.13 gDay
3.2.14 gMonth
3.2.15 hexBinary
3.2.16 base64Binary
3.2.17 anyURI
3.2.18 QName
3.2.19 NOTATION

3.3 Derived datatypes

3.3.1 normalizedString
3.3.2 token
3.3.3 language
3.3.4 NMTOKEN
3.3.5 NMTOKENS
3.3.6 Name
3.3.7 NCName
3.3.8 ID
3.3.9 IDREF
3.3.10 IDREFS
3.3.11 ENTITY
3.3.12 ENTITIES
3.3.13 integer
3.3.14 nonPositiveInteger
3.3.15 negativeInteger
3.3.16 long
3.3.17 int
3.3.18 short
3.3.19 byte
3.3.20 nonNegativeInteger
3.3.21 unsignedLong
3.3.22 unsignedInt
3.3.23 unsignedShort
3.3.24 unsignedByte
3.3.25 positiveInteger

XML Schema Types from http://www.w3.org/TR/xmlschema-2



built-in *primitive* types

built-in *derived* types

ur-types: base of the type hierarchy

What is "ur-type"?
An abbreviation for "un-restricted"?
I don't know, but my guess is it comes from the German word "ur-text", meaning the "original" text of a document....

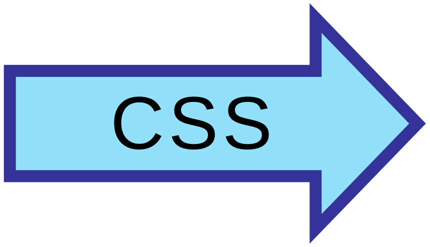
Things you can do with an XML document

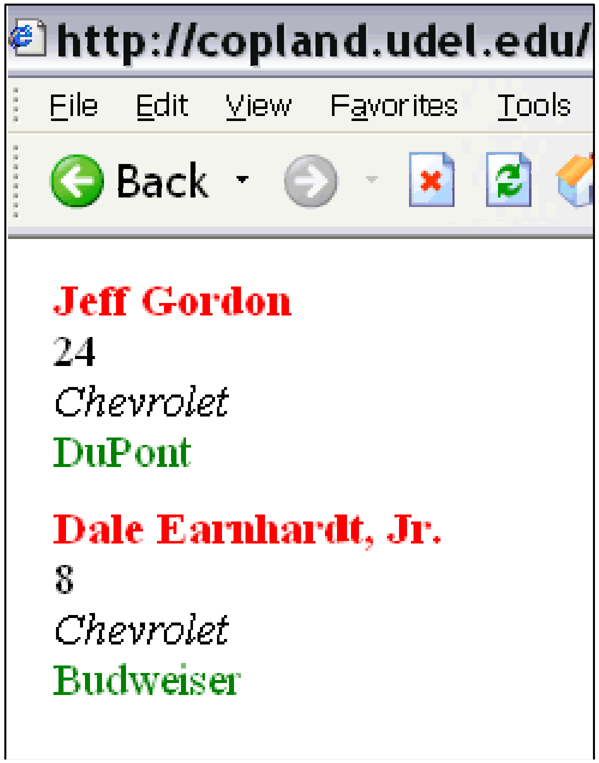
- Validate it against a DTD or Schema
- Format it for a browser using a Cascading Style Sheet (CSS)
- Parse it from Java (C++,Python,Perl, etc...) with the DOM or SAX APIs
  - DOM parses whole document into a tree, then lets you access it
  - SAX is event-based; it provides callbacks for a depth-first traversal
- Transform it into another format (e.g. HTML) using XSL/XSLT
  - often HTML is the target format, but could be LaTeX, MySQL commands, CSV, etc.
- Search it using XPath expressions
  - XPath is sort of a "query language" for XML
  - can be used to specify subset of elements that match some criteria
- Make links to subsets of content in the document using XPointer
  - XPointer provides a way of making a hyperlink to the subset of an XML document specified by an XPath expression.
- Use XLink to make hyperlinks with XLink
  - more sophisticated than can be made with regular HTML.

Preliminary notes on CSS

- "XML is content driven, not presentation driven." [1,p. 66]
- Cascading Style Sheets (CSS) provide rules for formatting style
- CSS version:
  - current version: http://www.w3c.org/TR/CSS2
  - CSS3 is being worked on; drafts available at www.w3c.org

```
<driver>
  <name>Jeff Gordon</name>
  <number>24</number>
  <make>Chevrolet</make>
  <sponsor>DuPont</sponsor>
</driver>
```


CSS

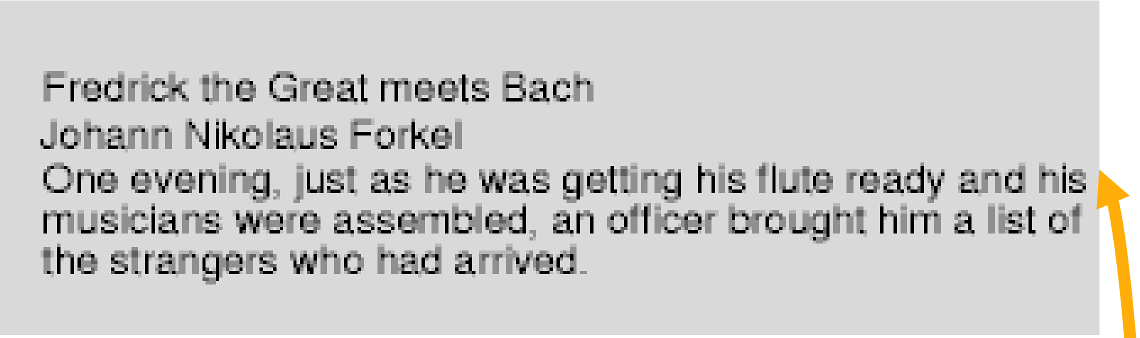


Jeff Gordon
24
Chevrolet
DuPont

Dale Earnhardt, Jr.
8
Chevrolet
Budweiser

Formatting XML with a CSS (part 1)

(taken directly from http://www.w3c.org/TR/CSS2)



Fredrick the Great meets Bach
Johann Nikolaus Forkel
One evening, just as he was getting his flute ready and his musicians were assembled, an officer brought him a list of the strangers who had arrived.

XML fragment, with Processing Instruction (PI) added at top:

```
<?xml:stylesheet type="text/css" href="bach.css"?>
<ARTICLE>
        <HEADLINE>Fredrick the Great meets Bach</HEADLINE>
        <AUTHOR>Johann Nikolaus Forkel</AUTHOR>
        <PARA> One evening, just as he was getting his
              <INSTRUMENT>flute</INSTRUMENT> ready and his musicians were
              assembled, an officer brought him a list of the strangers who had arrived.
        </PARA>
</ARTICLE>
```

Warning: possible typo? See later slide

article1.xml

To display like a document, for each element, declare it as either:

- inline-level (i.e., does not cause line breaks)
- block-level (i.e., causes line breaks).

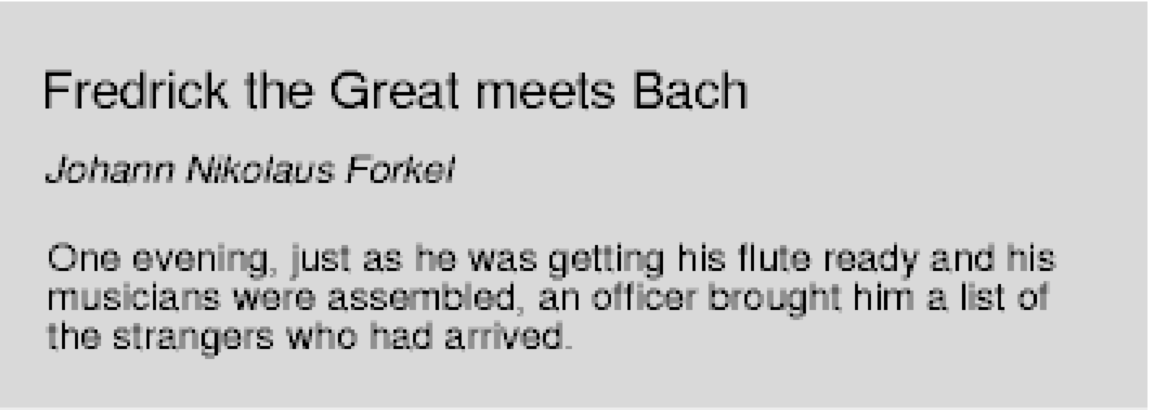
```
INSTRUMENT { display: inline }
ARTICLE, HEADLINE, AUTHOR, PARA { display: block }
```

bach.css

Put both files in same directory, view with a XML/CSS aware browser

Formatting XML with a CSS (part 2)

(taken directly from http://www.w3c.org/TR/CSS2)



Fredrick the Great meets Bach

Johann Nikolaus Forkel

One evening, just as he was getting his flute ready and his musicians were assembled, an officer brought him a list of the strangers who had arrived.

Same XML fragment, but specify a different CSS:

Warning: possible typo? See next slide

```
<?xml:stylesheet type="text/css" href="bach2.css"?>
<ARTICLE>
        <HEADLINE>Fredrick the Great meets Bach</HEADLINE>
        <AUTHOR>Johann Nikolaus Forkel</AUTHOR>
        <PARA> One evening, just as he was getting his
              <INSTRUMENT>flute</INSTRUMENT> ready and his musicians were
              assembled, an officer brought him a list of the strangers who had arrived.
        </PARA>
</ARTICLE>
```

article1.xml

- headline font size larger than then rest of the text
- display the author's name in italic:

```
INSTRUMENT { display: inline }
ARTICLE, HEADLINE, AUTHOR, PARA { display: block }
HEADLINE { font-size: 1.3em }
AUTHOR { font-style: italic }
ARTICLE, HEADLINE, AUTHOR, PARA { margin: 0.5em }
```

bach2.css

An annoying mystery...

- An annoying mystery:
  - www.w3c.org is the official place to go for Web Standards
  - I copied this XML Processing instruction directly from their own tutorial:
    <?xml:stylesheet type="text/css" href="bach.css"?>
  - It works fine in IE, but *refuses to work in Firefox.*
- What's up with that, I think? Is Firefox broken? Could IE be "better"? Say it isn't so!!!
- Further investigation reveals:
  - Firefox claims to adhere "strictly" to w3c.org standards. Their story:
    - If something works in IE or other browsers, but not in Firef
      then the other browser is permissive, while Firefox is strictly interpreting the standard.

Typo?
or alternative
syntax?

  - I looked at other examples on the web, and found the following syntax in many:
    Note hyphen instead of colon:
    <?xml-stylesheet type="text/css" ...
  - Firefox uses the MIME type set by the server rather than the file extension to determine how to present content (which is more in keeping with the standard.)
    - The Web server is the one that sets the MIME type in the HTTP response
    - If you can't configure the server directly, an.htaccess file might help (see next slide).

Example .htaccess file for setting MIME types

```
> ls -al
total 120
drwxr-xr-x   2 pconrad  0376        4096 Apr 19 12:02 .
drwxr-xr-x   3 pconrad  0376        4096 Apr 17 13:34 ..
-rw-r--r--   1 pconrad  0376        1513 Apr 18 20:06 .foo
-rw-r--r--   1 pconrad  0376        1487 Apr 18 19:58 .htaccess
...
> more .htaccess
# CSS
AddType text/css .css

# XHTML
AddType application/xhtml+xml .xhtml

# XML
AddType text/xml .xml

# SVG
AddType image/svg+xml .svg .svgz
AddEncoding x-gzip .svgz

# HTML
# Server Side Includes (SSI)
AddType text/html .shtml

# Active Server Pages
AddType text/html .asp
...
```

Note the use of Unix command
to show hidden files:
ls -al

Compare:

Firefox: "Tools/Page Info"

IE: " File/Properties"

On strauss, without a .htaccess file,
IE worked, but Firefox did not.

Firefox claims to be more "correct" in this behavior,
i.e. working from MIME type in HTTP response header,
not the file extension.

Let's try it with our Drivers file

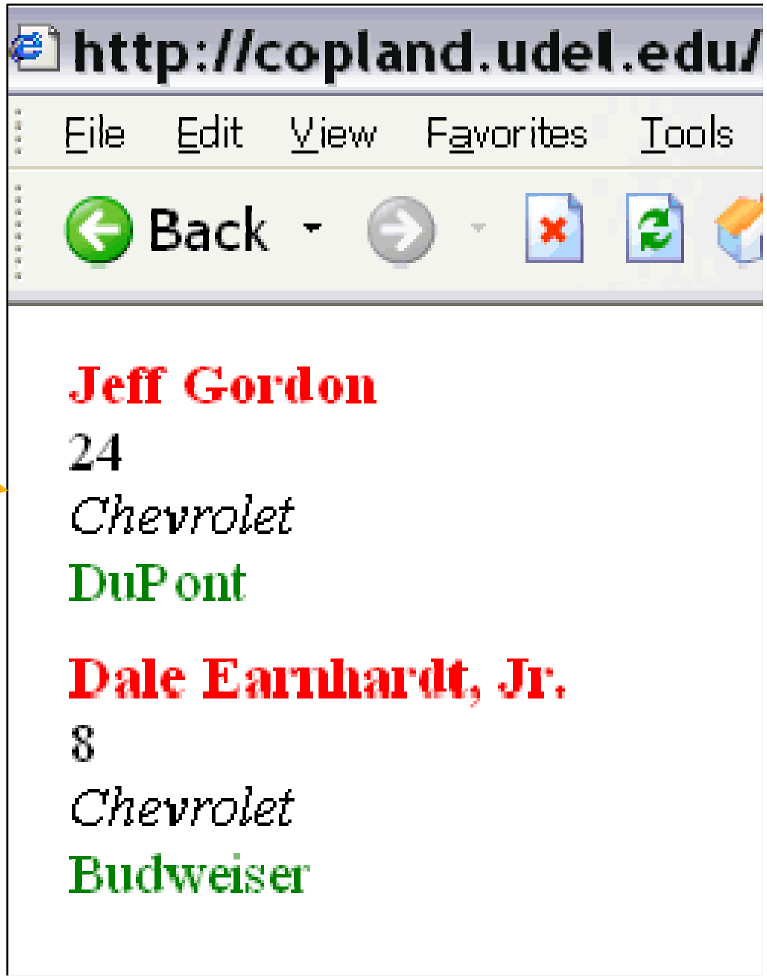
drivers.xml

```
<?xml-stylesheet type="text/css" href="drivers.css"?>
<drivers>
<driver>
   <name>Jeff Gordon</name>
   <number>24</number>
   <make>Chevrolet</make>
   <sponsor>DuPont</sponsor>
</driver>

<driver>
   <name>Dale Earnhardt, Jr.</name>
   <number>8</number>
  <make>Chevrolet</make>
  <sponsor>Budweiser</sponsor>
</driver>
</drivers>
```

drivers.css

```
driver, name, number, sponsor, make { display: block; }
driver { margin: 0.5em; }
name { font-weight: bold; color: red; }
make { font-style: italic; }
sponsor { color: green; }
```



Jeff Gordon
24
Chevrolet
DuPont

Dale Earnhardt, Jr.
8
Chevrolet
Budweiser

- A few practical things I ran into:
  - I put both files in the same directory on my web site.
  - I had to put a top-level element called "<drivers></drivers>" around my list of "<driver>...</driver><driver>...</driver>". The browsers didn't like it if there was more than one top-level element in the document.
  - On strauss, I had to include a .htaccess file to set the MIME types to get Firefox to work.

Homework assignment H04

- Using the same product you used for your H02 and H03, and following the example of the NASCAR drivers file in these slides, create an XML file called prod.xml.
  - Your prod.xml file should embed at least four elements inside each element (e.g. four elements for each NASCAR driver).
- Create a prod.css file so that your product is formatted in some interesting way. Use fonts, colors, and margins.
- Create a directory on strauss called ~userid/public_html/cisc474/H04 and place your prod.xml and prod.css file in this directory. Include also an .htaccess file so that this file can seen with formatting from both IE and Firefox.