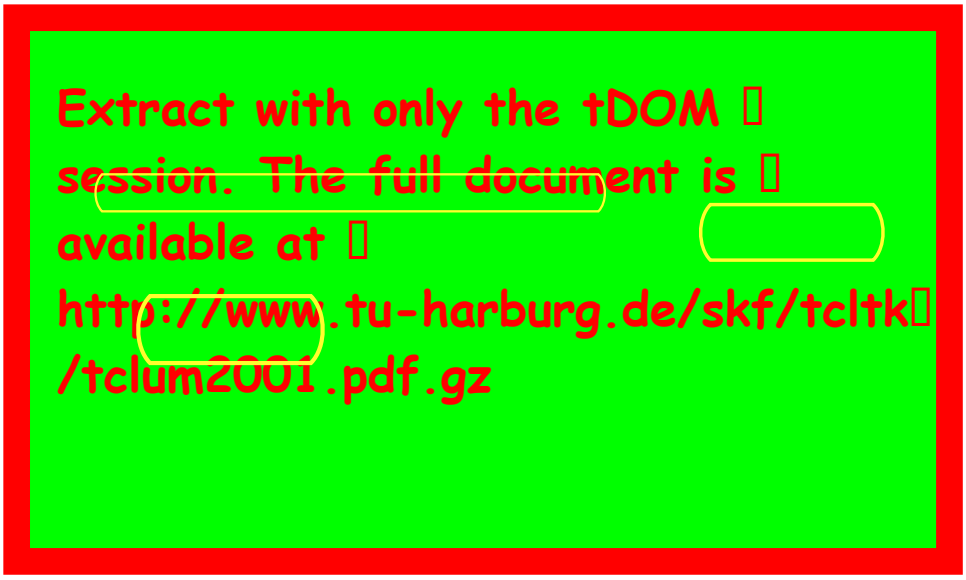
Second European Tcl/Tk User Meeting



Technische Universität Hamburg – Harburg
7. – 8. June 2001

Foreword

These are the proceedings of the Second European Tcl/Tk User Meeting, held the 7th. and 8th. of June in Hamburg.
For more information about this event or proceedings in PDF format see `http://www.tu-harburg.de/skf/tcltk`.

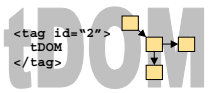
Carsten Zerbst (`mailto:Zerbst@Tu-Harburg.de`)

Contents

6 tDOM

Jochen Löwer (mailto:jochen_loewer@hp.com)

What is tDOM - (new) Features

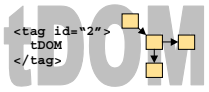


- Tcl package for Tcl 8.x
- two XML parsers (Expat (1.95.1) + SimpleParser)
- enhanced TclExpat (SAX) (originally from Steve Ball / Scriptics)
- DOM implementation (DOM core level 1 + extensions)
- [incr Tcl] / OO - like calling syntax
- fast XPath implementation
- high performance (written in C)
- low memory consumption (enhancements)
- extension namespaces for DOM methods / XPath functions
- free for any use: MPL
- partial XSLT support
- extension system and validiation extension
- HTML parser and element builder
- more DTD information
- thread safeness

Current version: `tDOM-0.52`

`tDOM-0.6x` **in some weeks**

Developers / Contributors

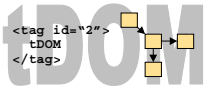


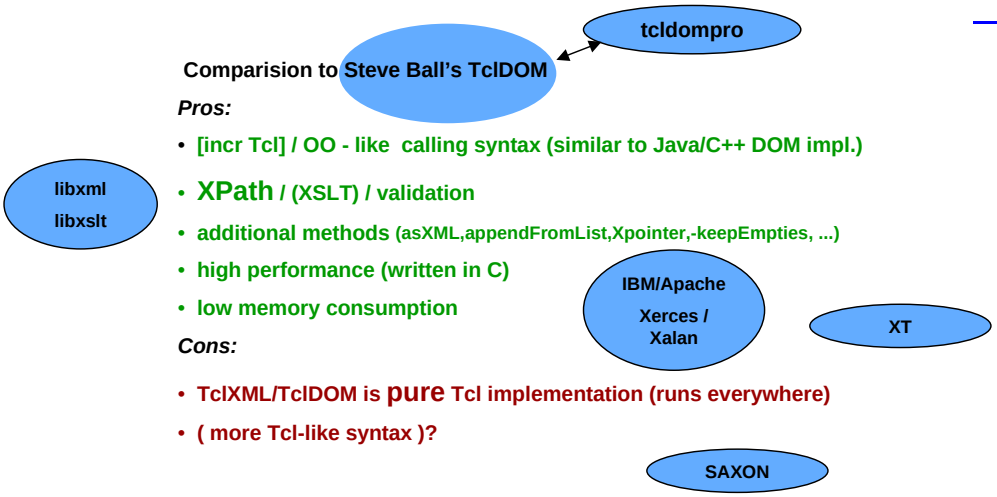
Jochen Loewer	Core (DOM, Tcl binding, Xpath, XSLT)
Rolf Ade	Extension Architecture, Validator, TclExpat enhancements,
Zoran Vasiljevic	Thread Support, Node commands (Tcl Dynamic Pages)

+ various bug reporters

- email based development work
- major releases done by J.Loewer
- use SourceForge for future ? (tdom.sourceforge.net already created, but not used)
- old major site (http://sdf.lonestar.org/~loewerj/tdom.cgi) is currently down!
- Maillinglist on www.egroups.com/group/tdom

Related Work - Pros / Cons





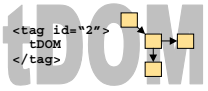
Pros:

- [incr Tcl] / OO - like calling syntax (similar to Java/C++ DOM impl.)
- XPath / (XSLT) / validation
- additional methods (asXML,appendFromList,Xpointer,-keepEmpties, ...)
- high performance (written in C)
- low memory consumption

Cons:

- TclXML/TclDOM is pure Tcl implementation (runs everywhere)
- (more Tcl-like syntax)?

tDOM's usage in the world

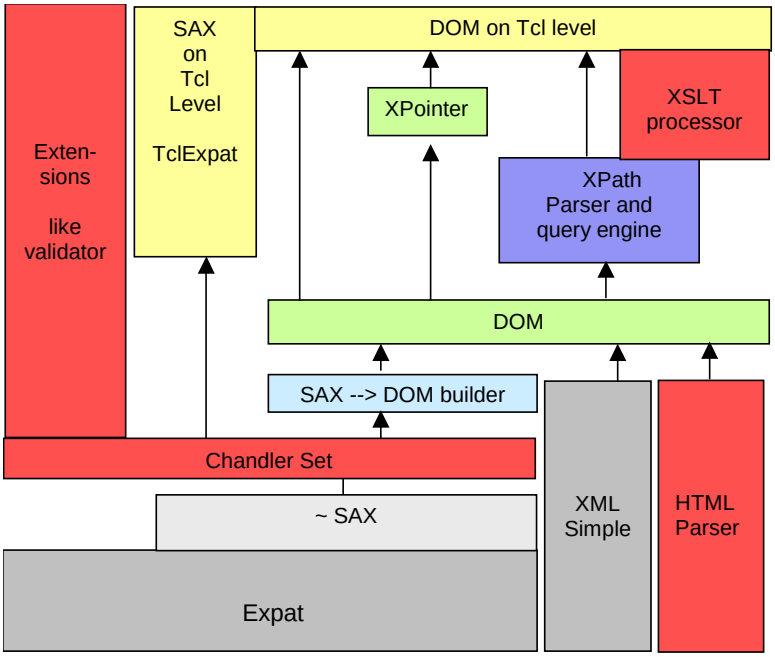


- UML modelling tool (XMI) by a danish company
- BMEcat application (Rolf Ade)
- AOLserver modules (Tcl → DOM → HTML = Tcl Dynamic Pages) Zoran Vasiljevic
- configuration the high-end server (SuperDome)
- external system communication (logistic information HTTP/XML)
- frontend to backend communication
- ? C part used in a WML system/application in Hongkong ?

...

and lots of downloads from various organizations (IBM, Compaq, SUN, Software AG,....)

Architecture



OO-like Syntax / Object Commands

tDOM creates Tcl commands on the fly while traversing the DOM tree, which point to domNode C structuresusing their clientData:

```
domDoc<N>    for DOM document objects
domNode<N>   for all nodes (element,text,comment,PI)
```

No attribute, NamedNodeMap or DocumentFragments objects !

Basic syntax is:

```
$obj method arg1 arg2 …
```

```
% set doc [dom parse $xml]
domDoc1

% set rootNode [$doc documentElement]
domNode1

% domNode2 nodeType
invalid command name "domNode2"

% set child [$rootNode firstChild]
domNode2

% domNode2 nodeType
ELEMENT_NODE
```

domNode2 command has not been created yet

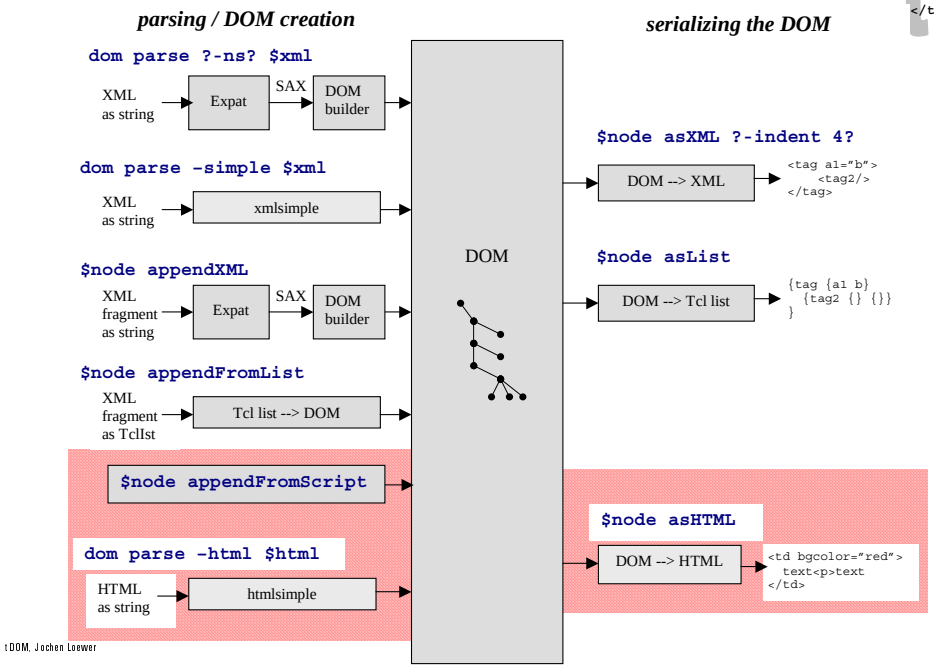
return reference to domNode2 will create a command

Available DOM Methods for Nodes

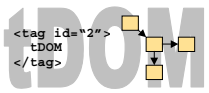
```
domNode<n> method arg1 arg2 ...
```

basic properties	nodeType nodeName nodeValue ownerDocument
navigate	parentNode firstChild lastChild nextSibling previousSibling hasChildNodes childNodes getElementsByTagname
handle attribute	get/setAttribute removeAttribute hasAttribute attributes @<attr>
modify	appendChild instertBefore replaceChild removeChild cloneNode
PIs/text	target data text
Xpointer97 navigation search methods	find child descendant ancestor fsibling psibling
XPath XSLT	selectNodes **xslt**
serialize	asList asXML **asHTML**
add fragments	**toXPath**
optional properties	appendFormList **appendFromScript** appendXML getLine getColumn

Parsing / DOM builder / Serialization



New I/O for Parsers / Serializers



Instead of passing/retrieving a string, a Tcl channel or filename can be used

For XML parsing / DOM building:

```
$expatParserHandle parsechannel <ch>
$expatParserHandle parsefile <f>
$dom parse -channel <ch>
```

For serializers:

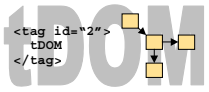
```
$node asXML -channel <ch>
$node asHTML -channel <ch>
```

Advantages:

Avoids additional copy of data in memory

Parse while data gets in and stop before the end

Namespace Support



While DOM building (dom parse) namespaces are parsed and stored in a per document table.

Nodes and attribute nodes contain 8 bit index to document namespace table (→ memory savings).

New methods:

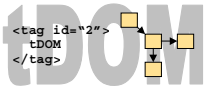
```
$node namespaceURI
$node prefix
$localName
```

not (yet) implemented:

```
$node getAttributeNS
$node setAttributeNS
$node hasAttributeNS
$node getElementsByTagNameNS
```

Right now also the XML serializer doesn't create namespace definitions for elements/attributes (→XSLT issue)

DTD Information



Moved back to standard Expat distribution (1.95.1 on SF) from hacked / improved version (by Scriptics/PerlXML/own hacks)

→ gives callbacks while DTD parsing (element / attribute declarations)

Input DTD:

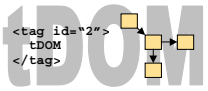
```
<?xml version="1.0" ?>
<!DOCTYPE root [
<!ELEMENT spec (front, body, back?)>
<!ELEMENT div1 (head, (p | list | note)*, div2*)>
]>
```

Callbacks on Tcl level:

```
XmlDecl 1.0 {} {}
StartDocTypeDel root NULL NULL 1
ElemDecl spec {
  SEQ {} {} {
      {NAME {} front {}}
      {NAME {} body {}}
      {NAME ? back {}}
  }
}
ElemDecl div1 {
  SEQ {} {} {
    {NAME {} head {}}
    {CHOICE * {} {
       {NAME {} p {}}
       {NAME {} list {}}
       {NAME {} note {}}
     }}
     {NAME * div2 {}}
  }
}
EndDocTypeDecl
```

could be useful for Validators + Code Generators (generate validate code, generate object/data extraction code)

Thread Support



Initial modification for thread safeness did Zoran Vasiljevic for AOLserver integration last year.

Basically moved global data to thread local storage.

Each thread will have his own object counter, no conflicts, no locks needed,

but no ability to share / hand over DOM documents.

To enable this Zoran made a new implementation in May/June.

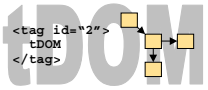
There will be locks on the whole document, which are controlable via two new methods.

Document objects will be passed between threads through object names containing the physical address (doc4f00340 → able to kill whole interpreter if bad address is used!).

There are currently discussion about a safer approach.

→ Thread support should have a compile time switch, so that non-threading tDOM uses gets no performance, robustness and complexity penalty!

Memory Consumption - DOM Allocator



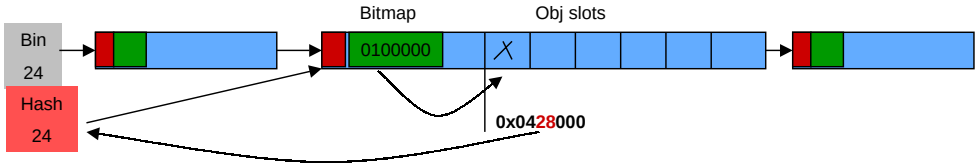
Many memory allocators can have quite large space requirements just for housekeeping information, usually around 8 bytes (linked list pointer + size)

DOM objects consists of many equally sized object → exploit this fact using a specialized allocator, which has a minimal overhead in these cases.

Idea:

- **use large blocks (32K) for perfect fit of equal sized objects**
- **use bitmap vector for used/free tracking (→ 1 bit overhead)**
- **all blocks for object of that size go into one bin**

`-DUSE_NORMAL_ALLOCATOR`
to disable it at compile time

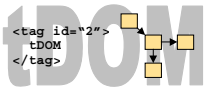


Allocating is fine. Freeing gets complicated:

How to find the right block info structure just by the given memory address?

→ Use hashing of some middle address bits and comparing against begin/end address (cache line conecpt for 1st level CPU caches)

xmlbench Performance Results

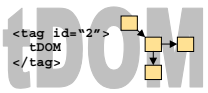


xmlbench suite (www.sosnoski.com/opensrc/xmlbench/results.html)
announced on www.xmlhack.com recently.
Test results made by Rolf this week on Win2000 (PII 333Mhz):

-Runtime(ms)-	much_ado	periodic	xml	
tdom SAX	771	591		
tdom DOM	1382	1101		
tdom DOM-simple	731	440		SAX: 3 times faster
Java SAX	2473	2153		
Crimson DOM	13149	9294		DOM: >9 times faster
JDOM	19458	10895		
dom4j	11557	8292		
Xerces DOM base	14361	10856		
Xerces DOM def	6429	5037		
Electric XML	17886	10095		

- Memory -	much_ado	periodic	xml	
tdom	332	212	284	
Crimson DOM	1101	603	817	3-4 times more momery
JDOM	1545	761	1025	
dom4j	1454	955	1167	
Xerces DOM base	1216	685	903	
Xerces DOM def	1065	738	1188	
Electric XML	1367	745	1089	

HTML parser



- based on simple XML parser
- modifications to parse HTML <= version 4.0 code
- No double quotes / ticks for attribute values
- Get script / style tag content unparsed
- Be able to deal with empty tags

```
<p> <br> <hr> ….
```

Main challenge:

To be able to deal with HTML coding errors !

```
<table> <tr> <td> Row1 Col1
             <td> Row1 Col2
        <tr> <td> Row1 Col1
             <td> Row2 Col2
</table>
<a href=/app.cgi?param=11><font color=white> LINK </a></font></a>
```

Advantage:

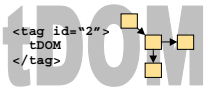
operate on the HTML document using DOM methods,
apply XPath queries
future: HTML -> DOM -> XSLT/Tcl -> WML ?
-> DoComo(?) HTML
(Japanese mobile HTML)

Idea: list of fields which could be autoclosed (under certain conditions)
ignoring some closing tags, which are left over

Heuristics need improvements!

Other sites won't change just because you can't parse them.

Usage of HTML parser

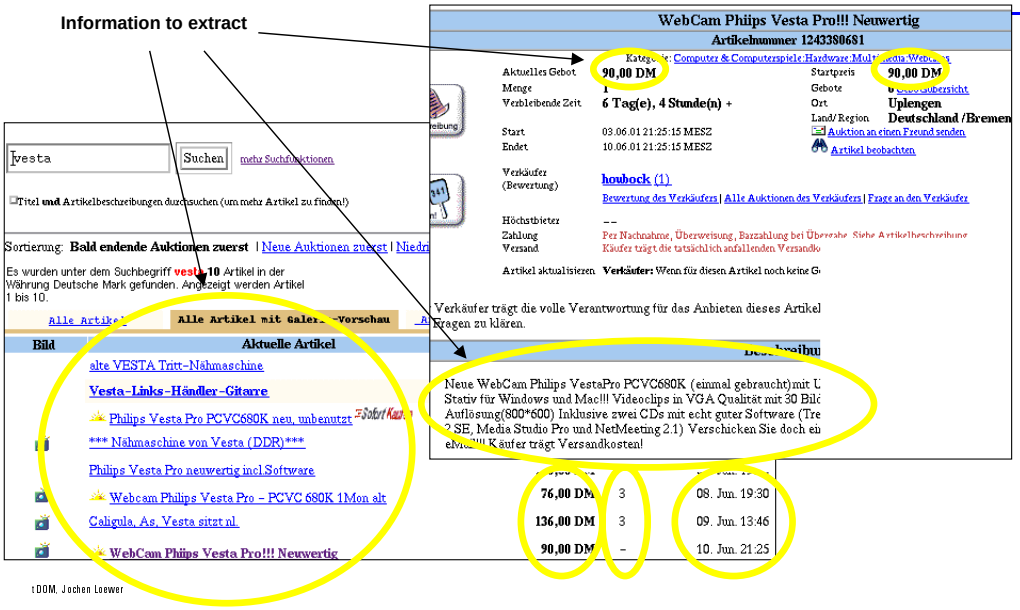


- HTML code analysis (browse for example with XE)
- HTML code condenser (rules to strip non visible white space, then asHTML)
- wrapper to (legacy) web application
- web robots / agents
- web service interfacing (→ WIDL WebMethods)

→ XPath features enable easy powerful scripting in Tcl !

Example: monitor / extract offering from Ebay

Web Extractor (Ebay)



XPath Queries: Example

all descendant element nodes

Predicate / Subfilter

```
//country[name='Germany']/province[population > 5000000]/name
```

filter all elements with nodeName 'country'

get all child nodes

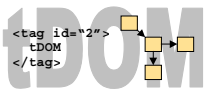
implicit convert of text nodes below 'population' tag into numbers

```
<country>
  <name>Germany</name>
  <province>
    <name>Berlin</name>
    <population>3472009</population>
  </province>
  …
  <province>
    <name>Bayern</name>
    <population>11921944</population>
  </province>
  …
</country>
```

'Join' over reference id values out of different parts in the XML:

```
//mountain[in_country[@ref = string(//country[name='Germany']/@id)]
```

Web Extractor (Ebay) – XPath extract code



down to object

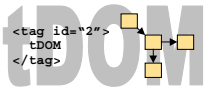
up start of row

```
set doc [dom parse -html $xml]
set root [$doc documentElement]
foreach item [$root selectNodes {//a[contains(@href,"item=")]/ancestor::tr }] {
   set object  [$item selectNodes { string(td[2]) }]
   set price   [$item selectNodes { string(td[3]) }]
   set bids    [$item selectNodes { string(td[4]) }]
   set endtime [$item selectNodes { string(td[5]) }]
}
```

next column to the right

```
set gebot       [$root selectNodes { string(//*[contains(.,'Aktuelles Gebot')]
                                           /following-sibling::td[1])}]
set startpreis  [$root selectNodes { string(//*[contains(.,'Startpreis')]
                                           /following-sibling::td[1])}]
set description [$root selectNodes { string(//blockquote[1])}]
```


Web Extractor (Ebay) – XE examples 1




```
xml(file:/home/jolo/ebay1.html) /
 - html
    + head
    = a  name=top
    + script  language=JavaScript
    C  header for search: find items
    ┤ body  BGCOLOR=#FFFFFF
       + script  SRC=http://include.ebay.com/aw/pics/de/js/cobrand/search_de.js
       + table  BORDER=0 CELLPADDING=0 CELLSPACING=0 WIDTH=600
       + center
       = br
       + form  NAME=forminput ACTION=search.dll METHOD=GET
       + table  border=0 cellPadding=0 cellSpacing=0 width=100%
       + table  width=100%
       + table  border=0 w
       + table  bgColor=#0
       + table  bgColor=#0
       + table  bgColor=#0
       + table  bgColor=#f
       + table  width=100%
       + table  border=0 b
       + table  width=100%
       + table  width=100%
       + table  width=100%
```

```
xml(file:/home/jolo/ebay1.html) //a[contains(@href,"item=")]/ancestor::tr
 - tr
    + td  align=center valign=middle width=11%
    + td  valign=top width=52%
    - td  nowrap=align="right" valign=top width=14%
       - font  size=3
          + b 10,00 DM
    + td  align=center valign=top width=6%
    + td  align=right valign=top width=16%

 - tr
    + td  align=center valign=middle width=11%
    + td  valign=top width=52%
    + td  nowrap=align="right" valign=top width=14%
    + td  align=center valign=top width=6%
    - td  align=right valign=top width=16%
       ├ font  size=3
          T 07. Jun. 07:11
```

Web Extractor (Ebay) – XE examples 2

```
xml(file:/home/jolo/ebay2.html) /
 - html
    + head
    C  header for browse: view item
    C  begin header
    - body  BGCOLOR=#FFFFFF
       + table  BORDER=0 CELLPADDING=0 CELLSPACING=0 WIDTH=600
       + center
       + center
       + center
       = br
       + table  border=0 cellpadding=8 cellspacing=0 width=100%
       + center
       - blockquote
          T
            Neue WebCam Philips VestaPro PCVC680K (einmal gebraucht)mit USE
            Käufer trägt Versandkosten!

       + a  name=ebayphotohosting
```

```
xml(file:/home/jolo/ebay2.html) string( //*
                                          [contains(.,"Aktuelles Gebot")]
                                            /following-sibling::td[1]
                                        )
90,00 DM
```


Handler Sets for Expat / Stacked tDOM

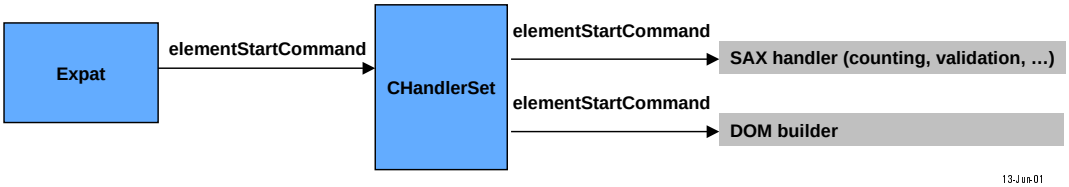
developed by Rolf Ade

For DOM building the Expat callbacks (SAX events) are tight to the DOM object creation functions.
Why not having having the SAX events trigger other actions (element statistics, validation, …) beside the standard DOM builder in parallel?
Having a subsequent parse to accomplish that is not a very clever alternative!

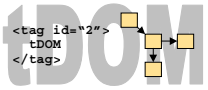
Idea:

Extent Expat to be able to invoke multiple callbacks for an event → CHandlerSets

CHandlerSets provide a stackable modular extension mechanism, which allows to write extensions separate from the main tDOM distribution



tnc Extension: DTD based validation



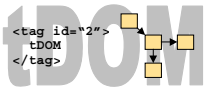
developed by Rolf Ade

The tnc extension package uses the CHandlerSets and allows fast C speed DTD based validation while building up the DOM tree at the same time.

```
package require tdom
package require tnc

roc LoadAndValidate { xml } {
    set parser [expat]
    tnc  $parser enable
    tdom $parser enable
    $parser parse $xml
    set doc [tdom $parser getdoc]
    puts [[$doc documentElement] asXML]
    $parser free
    return $doc
}
LoadAndValidate {<?xml version="1.0"?>
    <!DOCTYPE Test [ <!ELEMENT Test (#PCDATA) > ]><Test></Test>}
LoadAndValidate {<?xml version="1.0"?>
    <!DOCTYPE Test [ <!ELEMENT Test (#PCDATA) > ]><TestFoo></TestFoo>}
```


NodeCmd



developed by Zoran Vasiljevic

Fast C implementation for DOM node creation, which could nest.

Example:

```
% dom createNodeCmd elementNode html::body
% dom createNodeCmd elementNode html::title
% dom createNodeCmd textNode    html::t
```

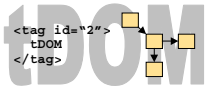
And usage:

```
% set d [dom createDocument html]
% set n [$d documentElement]
% $n appendFromScript {
    html::body {
        html::title { html::t "This is an example" }
    }
% puts [$n asHTML]
<html>
  <body>
    <title>This is an example</title>
  </body>
</html>
```

This is the foundation for Zoran tdomtdp package for AOLserver: Tcl Dynamic Pages

Tcl Dynamic Pages - Examples

developed by Zoran Vasiljevic



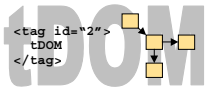
Example:

```
body {
  #
  # Server info
  #
  h2 {t "Server Information"}
  blockquote {
    table {
      foreach item {
        server hostname address pid uptime boottime home config log
        pageroot tcllib nsd argv0 name version label builddate platform
      } {
        tr {
          td -align right -valign top {b {t "$item: "}}
          set itemval [ns_info $item]
          if {$item == "boottime"} {
              set itemval [ns_httptime $itemval]
          }
          td -align left -valign top {t "$itemval\\ "}
        }
      }
    }
  }
}
```

similar to Don Libes' cgi.tcl approach, but builds DOM tree fragments (in C).

At the end the DOM tree is serialized with [$root asHTML]

XSLT processor (in C)



Right now there is no directly Tcl embedded XSLT engine available (Steve Ball did some wrapper to XT/SAXON/... outside as a process)

Implementation started end of July 2000, most code done until October.
Got final approval to also release it as Open Source in December.
Currently ~ 70 KByte C code and a new code in domxpath (LocationPath matcher)
(obj code: 55K for XPATH, 26K for XSLT)

Problem: A lot of templates have to be checked in parallel

```
<xsl:template match="book/author">
…
</xsl:template>
<xsl:template match="book/title">
…
</xsl:template>
```

Eval XPathExpr

AxisChild book
AxisChild author

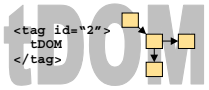
LocPathMatches(node)

IsElement author
ToParent
IsElement book

match contains LocationPath and not XPath expressions. So current XPath implementation doesn't help much and evaluates for the current node downwards.

→ **new LocationPath parser had to be coded**
→ **bottom-up match approach**

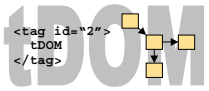
XSLT processor (in C) – Usage




```
$root xslt $xsltDoc transformedDoc   <-- change to document method

proc ApplyTemplate { xml xslt } {
      dom parse -keepEmpties $xml  xmlDoc
      dom parse -keepEmpties $xslt xsltDoc
      [$xmlDoc documentElement] xslt $xsltDoc transformedDoc
      # depending on the output type
      [$transformedDoc documentElement] asXML
      [$transformedDoc documentElement] asHTML
}
```


XSLT processor (in C) – Current State



Other work / competitors:

C: libxml/libxslt from D.Veillard (GNOME)
C++: Sabletron
Java: XT James Clark
Saxon
Xalan (IBM -> Apache XML project)

currently already passed:

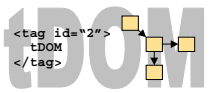
Mozilla XSLT engine tests
a great part of tests from LibXML/LibXSLT (GNOME, D. Veillard)
some other XSLT test from various sources
some of M.Kay's tests
Joe English TMML Tcl man page formatter does not work completely (key/import is missing)

current problems:

- **document fragments for xslt:variables / XPath expression**
- **xslt:number, xslt:sort, ... partial implemented**
- **function format-number (Tcl based implementation initially preferred)**
- **missing:**
  - **element creation with namespaces**
  - **output type determination and handling (only text, may need asText)**
  - **xslt:key**
  - **xslt:import / xslt:include / xslt:apply-import**

→ **not usable in production code, if all features are needed!**

XPath / XSLT extension functions in Tcl

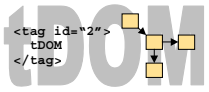


If function is not found in Xpath / XSLT processing, a callback can try look for a Tcl-level implementation:

```
proc ::dom::xpathFunc::format-number { ctxNode pos nodeListType nodeList args } {
    set argLen [llength $args]
    if { ($argLen != 4) && ($argLen != 6) } {
        return -code error "wrong number of args: format-number(node,typeString,?decFormat?)"
    }
    foreach { arg1Typ arg1Value arg2Typ arg2Value } $args break
    set num       [::dom::xpathFuncHelper::coerce2number $arg1Typ $arg1Value]
    set formatStr [::dom::xpathFuncHelper::coerce2string $arg2Typ $arg2Value]
    ...
    return [list string $num]
}
```

Good way to get something implemented and working first on Tcl level.
Later recode in C can be made, if performance matters.

Future / Enhancements



- **finalize thread support**
- **xslt bugs fixes and enhancements**
- **namespace enhancement (c level creation functions, tcl level methods, namespace support in XML serializer)**
- **release tDOM-0.6x soon.**

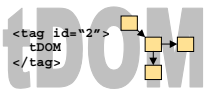
- **PURL for tDOM / sourceforge (?)**
- **XML schema validation**
- **SOAP client/server**
- **UDDI ?**
- **new XML Query proposal by W3C (Software AG,...)**

Poll:

What do want to have?

Use of XML or XML-based RPC technics (SOAP) in future projects?

Thanks



- Thanks for interest
- Call for help / volunteers / testers / users.
- Questions?