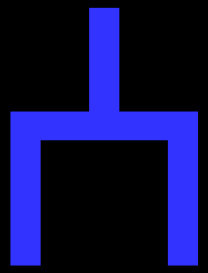
Tehnologii Web

procesarea datelor XML (I)



DOM (*Document Object Model*)

“Regula de aur este că nu există reguli de aur.”

George Bernard Show

Cum putem prelucra documentele XML?

Tipuri de procesări XML

procesare manuală

e.g., expresii regulate ☹

Tipuri de procesări XML

procesare obiectuală

DOM (*Document Object Model*)
non-DOM

Tipuri de procesări XML

procesare condusă de evenimente

SAX (*Simple API for XML*)
XPP (*XML Pull Parsing*)

Tipuri de procesări XML

procesare simplificată

Simple XML

Tipuri de procesări XML

procesare particulară

via API-uri specializate pentru a prelucra
tipuri de documente specifice – *e.g.*, RSS, SOAP, SVG,…

Procesoare (analizoare) XML
fără validare

verifică doar dacă documentul
este bine-formatat (*well formed*)

Expat, libxml, MSXML,...

Procesoare (analizoare) XML
cu validare

verifică dacă documentul este valid,
conform unei metode de validare – *e.g.*, DTD

Apache Xerces, JAXP, libxml, MSXML,...

Modelul DOM

introducere
interfețe DOM
DOM Core
DOM – nivelul 2
DOM – nivelul 3
DOM – nivelul 4
implementări
DOM direct în navigator

dom: intro

Scop:
procesarea obiectuală – standardizată –
a documentelor XML și/sau HTML

dom: caracterizare

Interfață de programare a aplicațiilor (API) abstractă pentru XML/HTML

dom: caracterizare

Interfață de programare a aplicațiilor (API) abstractă pentru XML/HTML

independentă de platformă și limbaj

standardizată de Consorțiul Web

dom: caracterizare

Definește o structură logică arborescentă
a documentelor XML

document = ierarhie a unui set de obiecte

dom: niveluri de specificare

DOM 1 (1998)

http://www.w3.org/TR/REC-DOM-Level-1/

DOM Core pentru XML

DOM HTML pentru procesarea standardizată
a paginilor Web – uzual, la nivel de client (*browser*)

dom: niveluri de specificare

DOM 2 (2001)

http://www.w3.org/TR/REC-DOM-Level-2/

recomandări multiple privind diverse funcționalități:
spații de nume, aplicare de stiluri,
răspuns la evenimente etc.

dom: niveluri de specificare

DOM 3 (2004)

http://www.w3.org/TR/DOM-Level-3-Core/

funcționalități specifice oferite de module
(unele deja standardizate)

XPath, traversare, validare,
încărcare & salvare (asincrone),...

dom: niveluri de specificare

DOM 4 (2014)

http://www.w3.org/TR/dom/
în stadiu de ciornă la Consorțiul Web

http://dom.spec.whatwg.org/
în contextul HTML 5 – *living standard* (18 aprilie 2015)

dom: interfețe

Modalitate abstractă de accesare și de modificare a reprezentării interne a unui document XML

dom: interfețe

Modalitate abstractă de accesare și de modificare a reprezentării interne a unui document XML

datele sunt încapsulate în obiecte, ascunse și/sau protejate de prelucrarea externă directă

dom: interfețe

Nu implică o implementare concreta, particulară:

DOM oferă interfețe
independente de implementare pentru
accesarea/procesarea datelor

dom: interfețe

Interfețele sunt specificate cu
IDL (*Interface Description Language*)

introdus în premieră de CORBA
(*Common Object Request Broker Architecture*)

dom: interfețe – IDL

Definește tipurile de obiecte
prin specificarea interfețelor acestora
(date membre + metode publice)

pur declarativ

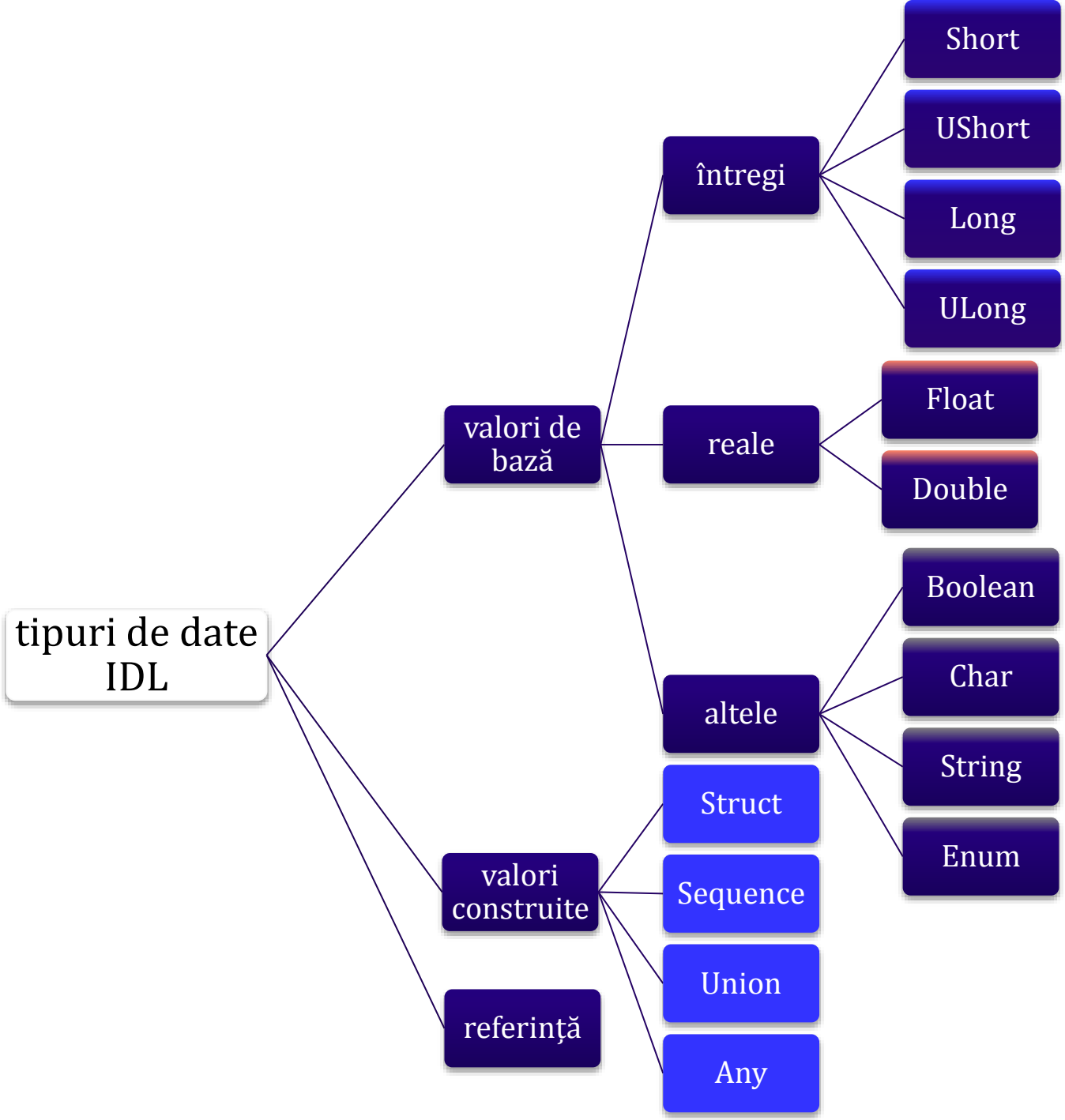
dom: interfețe – IDL

Oferă suport pentru moștenire multiplă prin intermediul interfețelor

specifică module, interfețe, metode, tipuri, atribute, excepții, constante

www.w3.org/TR/WebIDL/

candidate recommendation (W3C, aprilie 2012)


tipuri de date IDL
valori de bază
întregi
Short
UShort
Long
ULong
reale
Float
Double
altele
Boolean
Char
String
Enum
valori construite
Struct
Sequence
Union
Any
referință

dom: interfețe – exemplu

Specificarea interfeței **NodeList**

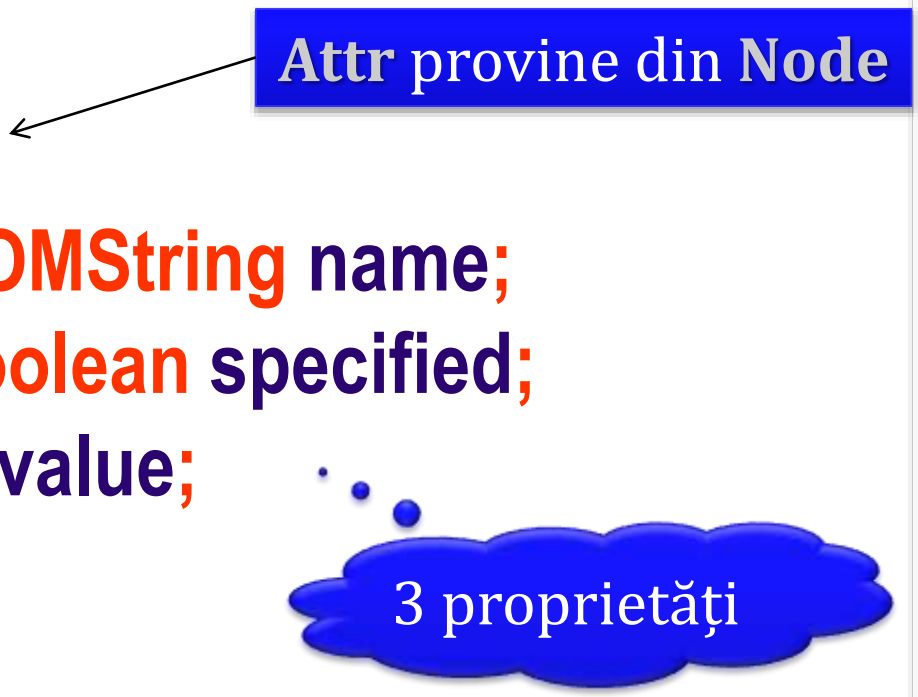
```
interface NodeList {
  Node item (in unsigned long index);
  readonly attribute unsigned long length;
};
```

metodă cu un parametru;
rezultat: o valoare de tip **Node**

proprietate *read-only*
de tip întreg lung fără semn

dom: interfețe – exemplu

Specificarea interfeței **Attr**



dom: core

Un document ≡ ierarhie de **obiecte-nod**
care pot implementa interfețe (specializate)

nodurile posedă descendenți ori sunt noduri frunză

a se (re)vedea
XML Infoset

Markup to test (permalink, save, upload, download, hide):

```
<html><head><title>DOM</title></head><body><article><h1>Tehnologii
Web</h1><img src="image" alt="Sigla materiei"/>Despre DOM...</article></body>
</html>
```

DOM view (hide, refresh):

```
└HTML
  ├HEAD
  │ └TITLE
  │   └#text: DOM
  └BODY
    └ARTICLE
      ├H1
      │ └#text: Tehnologii Web
      ├IMG alt="Sigla materiei" src="image"
      └#text: Despre DOM...
```

un document HTML și arborele DOM corespunzător reprezentat via Live DOM Viewer

http://software.hixie.ch/utilities/js/live-dom-viewer/

dom: core

Accesul la date
– liste de noduri, atribute, valori,… –
se realizează recurgându-se la metodele specifice
fiecărui tip de noduri ale arborelui

dom: core – tipuri de noduri

Document	**Element**, **ProcessingInstruction**, **Comment**, **DocumentType**
Document Fragment	**Element**, **ProcessingInstruction**, **Comment**, **Text**, **CDATASection**,…
Element	**Element**, **Text**, **Comment**, **CDATASection**, **EntityReference**,…
Attr	**Text**, **EntityReference**
Text	– (nod frunză al arborelui DOM)

dom: core

Interfețe fundamentale:

DOMException

gestionează setul de excepții de procesare

dom: core

Interfețe fundamentale:

DOMImplementation

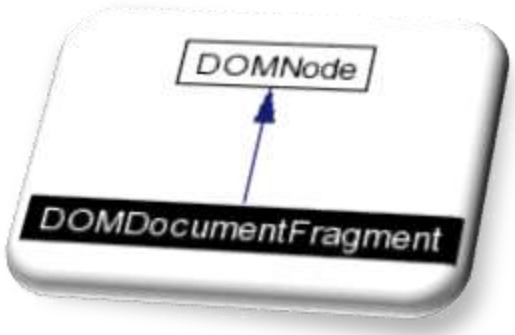
furnizează detalii despre implementarea curentă

dom: core

Interfețe fundamentale:

DocumentFragment : Node

acces la fragmente de arbore



dom: core

Interfețe fundamentale:

Document

oferă acces la document

pentru consultare și/sau modificare

dom: core

Interfețe fundamentale:

- Document
  - proprietăți
    - doctype, implementation, documentElement

dom: core

Interfețe fundamentale:

Document

proprietăți

doctype, implementation, documentElement

dom: core

Interfețe fundamentale:

Document

metode createElement(), createTextNode(), createAttribute(), getElementsByTagName(), getElementById(), createElementNS(), importNode(), getElementsByTagNameNS(), renameNode() etc.

DOM 2

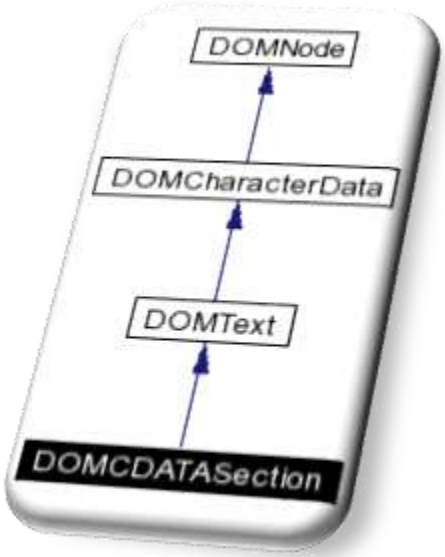
DOM 3

dom: core

Interfețe fundamentale:

Node

acces la nodurile arborelui



dom: core

Interfețe fundamentale:

- Node
  - proprietăți nodeName, nodeValue

dom: core

Interfețe fundamentale:

Node

metode getNodeType(), insertBefore(), appendChild(), replaceChild(), removeChild(), cloneChild(), hasChildNodes(), hasAttributes(), isSameNode()

DOM 2

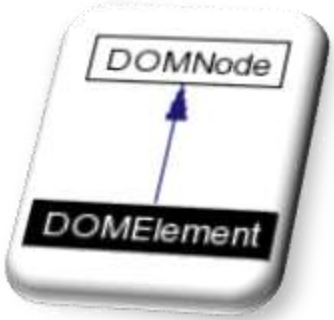
DOM 3

dom: core

Interfețe fundamentale:

Element

oferă acces la elementele XML



dom: core

Interfețe fundamentale:

- Element
  - proprietatea tagName

dom: core

Interfețe fundamentale:

Element

metode getAttribute(), getAttributeNode(), setAttributeNode(), removeAttributeNode(), hasAttribute(), hasAttributeNS(),...

dom: core

Interfețe fundamentale:

Attr : Node

acces la atributele unui element

dom: core

Interfețe fundamentale:

- NodeList
- NamedNodeMap
  - permit accesul la colecții de noduri
  - via indecși ori chei

dom: html

DOM HTML extinde DOM Core

specializarea interfețelor și oferirea de suport obiectual pentru prelucrarea documentelor HTML

standardizează procesarea paginilor Web
(*e.g.*, în cadrul navigatorului)

```
interface HTMLDocument : Document {
  attribute DOMString title;                    // titlul documentului
  readonly attribute DOMString referrer; // adresa resursei ce referă pagina
  readonly attribute DOMString domain; // domeniul de care aparține
  readonly attribute DOMString URL;      // URL-ul absolut al documentului
  attribute HTMLElement body;              // acces la elementul <body>
  readonly attribute HTMLCollection images; // lista tuturor imaginilor
  readonly attribute HTMLCollection links;     // lista tuturor legăturilor
  readonly attribute HTMLCollection forms;   // lista tuturor formularelor

  attribute DOMString cookie;                // acces la cookie-uri
    // emite o excepție dacă e asignată o valoare

  void open ();   // deschide un flux de scriere (alterează DOM-ul curent)
  void close ();  // închide fluxul de scriere și forțează redarea conținutului
  void write (in DOMString text);     // scrie un șir de caract. (e.g., cod HTML)
  void writeln (in DOMString text);  // idem, dar inserează și new line
  NodeList getElementsByName (in DOMString numeElement);
    // furnizează o listă de elemente conform unui nume de tag
};
```

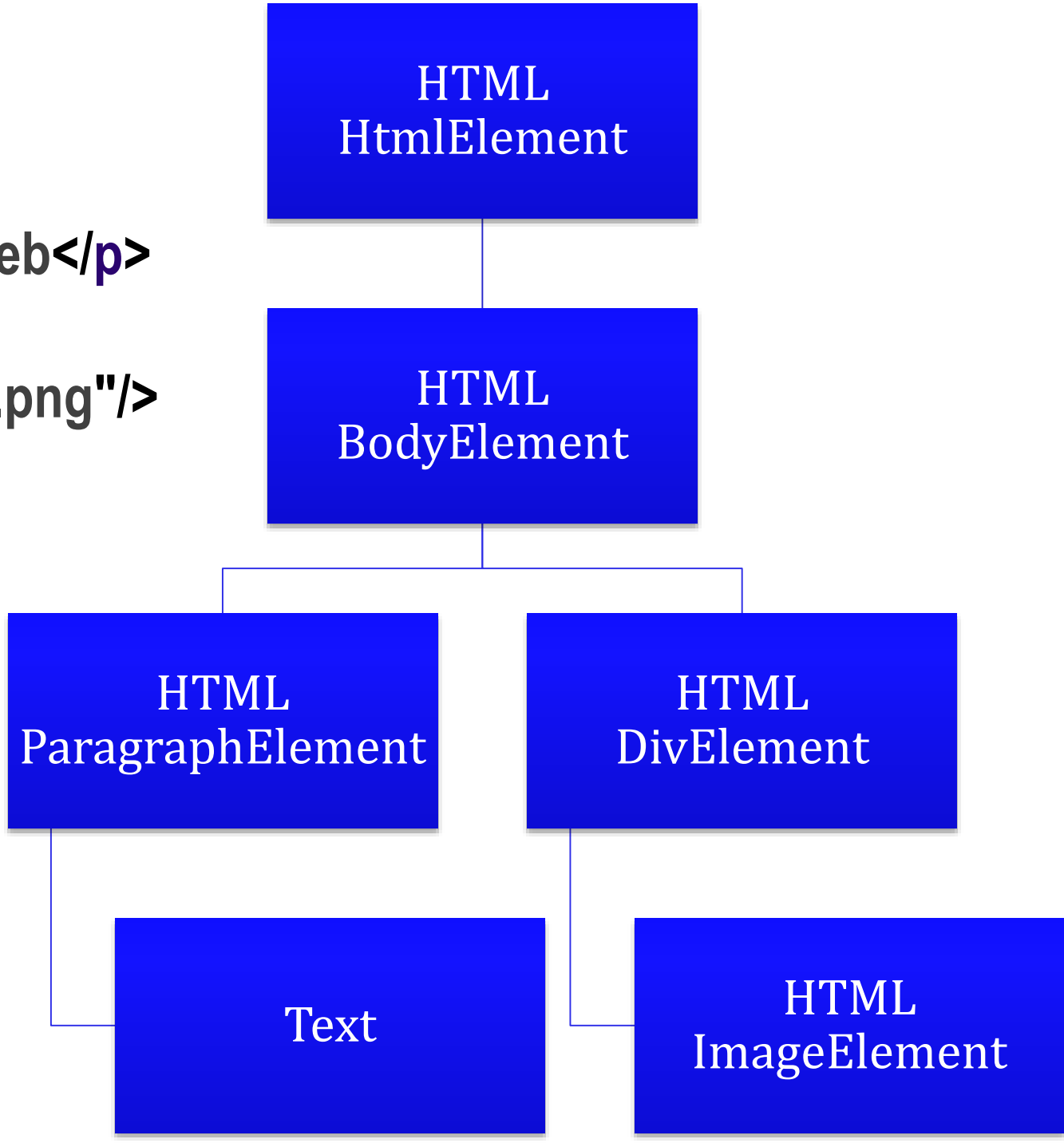
dom: html

Interfața generică **HTMLCollection**

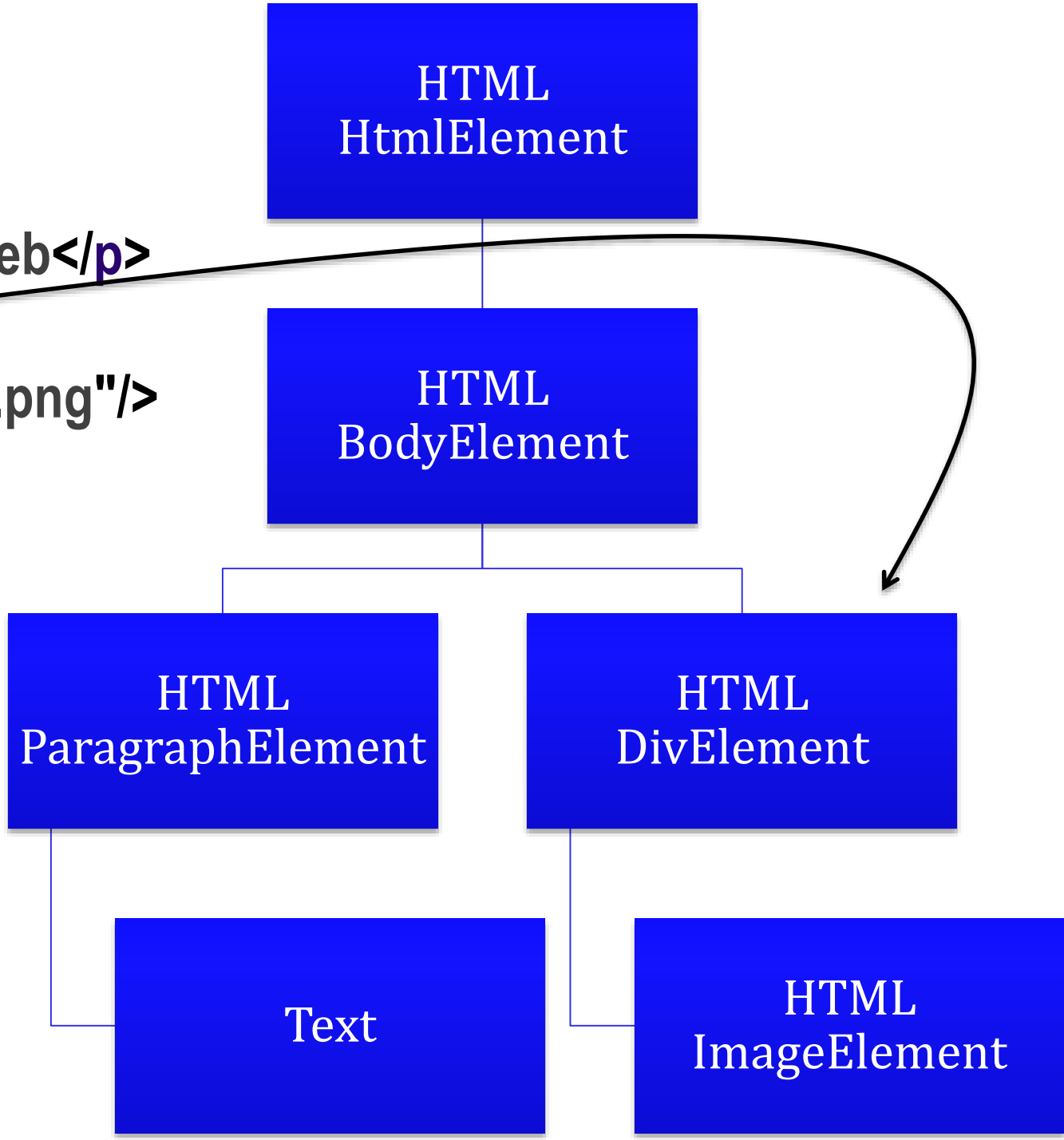
fiecărui element HTML îi corespunde o interfață specifică

HTMLDocument
HTMLDivElement
HTMLImageElement
...

```
<!DOCTYPE html>
<html>
  <body>
    <p>Tehnologii Web</p>
    <div>
      <img src="web.png"/>
    </div>
  </body>
</html>
```



```
<!DOCTYPE html>
<html>
 <body>
  <p>Tehnologii Web</p>
  <div>
   <img src="web.png"/>
  </div>
 </body>
</html>
```



```
// un element HTML generic
interface HTMLElement : Element {
  attribute DOMString      id;                  // identificator asociat elementului
  attribute DOMString      title;               // titlu explicativ
  attribute DOMString      lang;                // limba în care e redactat conținutul
  attribute DOMString      className;           // numele clasei CSS folosite pentru redare
};
// specifică un formular Web
interface HTMLFormElement : HTMLElement {
  readonly attribute HTMLCollection  elements; // elementele HTML incluse în formular
  readonly attribute long            length;    // numărul câmpurilor formularului
  attribute DOMString                action;    // URI-ul resursei ce procesează datele
  attribute DOMString                enctype;   // tipul MIME de codificare a datelor
                                                // (application/x-www-form-urlencoded)
  attribute DOMString                method;    // metoda HTTP folosită (GET sau POST)
  void                               submit();  // trimite date URI-ului definit de 'action'
};
// interfața DOM pentru elementul <img/> (e.g., conținut grafic raster: GIF, JPEG, PNG)
interface HTMLImageElement : HTMLElement {
  attribute DOMString                alt;       // text alternativ descriind conținutul grafic
  attribute DOMString                src;       // URL-ul resursei reprezentând imaginea
};
```

dom: nivelul 2

Extinde funcționalitățile DOM1
crearea unui obiect Document
copierea unui nod dintr-un document în altul

și multe altele...

dom: nivelul 2

Extinde funcționalitățile DOM1

alte facilități:
controlul aplicării foilor de stiluri CSS

tratarea evenimentelor

specificarea filtrelor și iteratorilor
(parcurgeri sofisticate de arbori DOM)

dom: nivelul 2

Suport pentru procesarea foilor de stiluri

StyleSheet

StyleSheetList

MediaList

DocumentStyle

detalii la **http://www.w3.org/TR/DOM-Level-2-Style**

Dr. Sabin Buraga ▲ www.purl.org/net/busaco

dom: nivelul 2

Tratarea evenimentelor

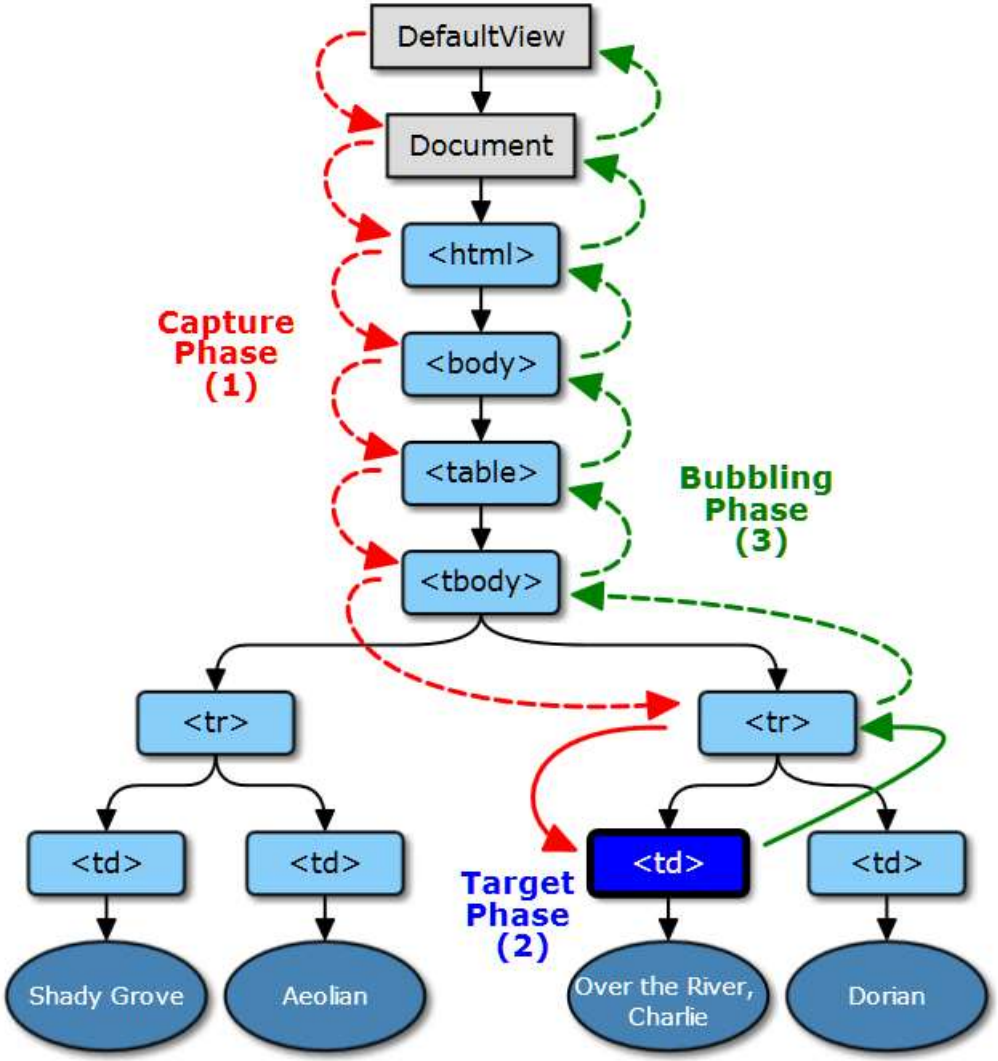
definirea de activități (*callback*-uri) executate
la apariția unui eveniment

eveniment = acțiune produsă în cadrul mediului de execuție în urma căreia programul va putea reacționa

dom: nivelul 2

Tratarea evenimentelor

descrierea arborescentă a fluxului de evenimente
(*cascading vs. bubbling*)



fluxul de evenimente (T. Leithead *et al.*, 2012)

a se studia și W. Page, *An Introduction to DOM Events* (2013)
http://coding.smashingmagazine.com/2013/11/12/an-introduction-to-dom-events/

dom: nivelul 2

Tratarea evenimentelor

conceperea unui set standard de evenimente

e.g., de control al interacțiunii cu utilizatorul și de notificare a modificărilor de structură

detalii la **http://www.w3.org/TR/DOM-Level-2-Events**

dom: nivelul 2

Tratarea evenimentelor

de interfață – *e.g.*, interacțiunea cu utilizatorul:
click, **mousedown**, **mouseup**, **mouseover**, **mousemove**,
keypress, **keydown**, **keyup**, **resize**, **scroll**

dom: nivelul 2

Tratarea evenimentelor

de modificare a structurii documentului:
subtreeModified, **nodeInserted**, **nodeRemoved**, **attrModified**,...

dom: nivelul 2

Tratarea evenimentelor

HTML (la nivel de document Web ori de formular):
load, **unload**, **abort**, **error**, **select**, **submit**, **focus**, **blur**,...

dom: nivelul 2

Tratarea evenimentelor

sunt puse la dispoziție interfețele:

EventTarget

EventListener

Event – UIEvent, MouseEvent, MutationEvent

dom: nivelul 2

Traversarea arborilor DOM

se specifică interfețele opționale

TreeWalker

NodeIterator

Filter

http://www.w3.org/TR/DOM-Level-2-Traversal-Range

Dr. Sabin Buraga ▲ www.purl.org/net/busaco

dom: nivelul 3

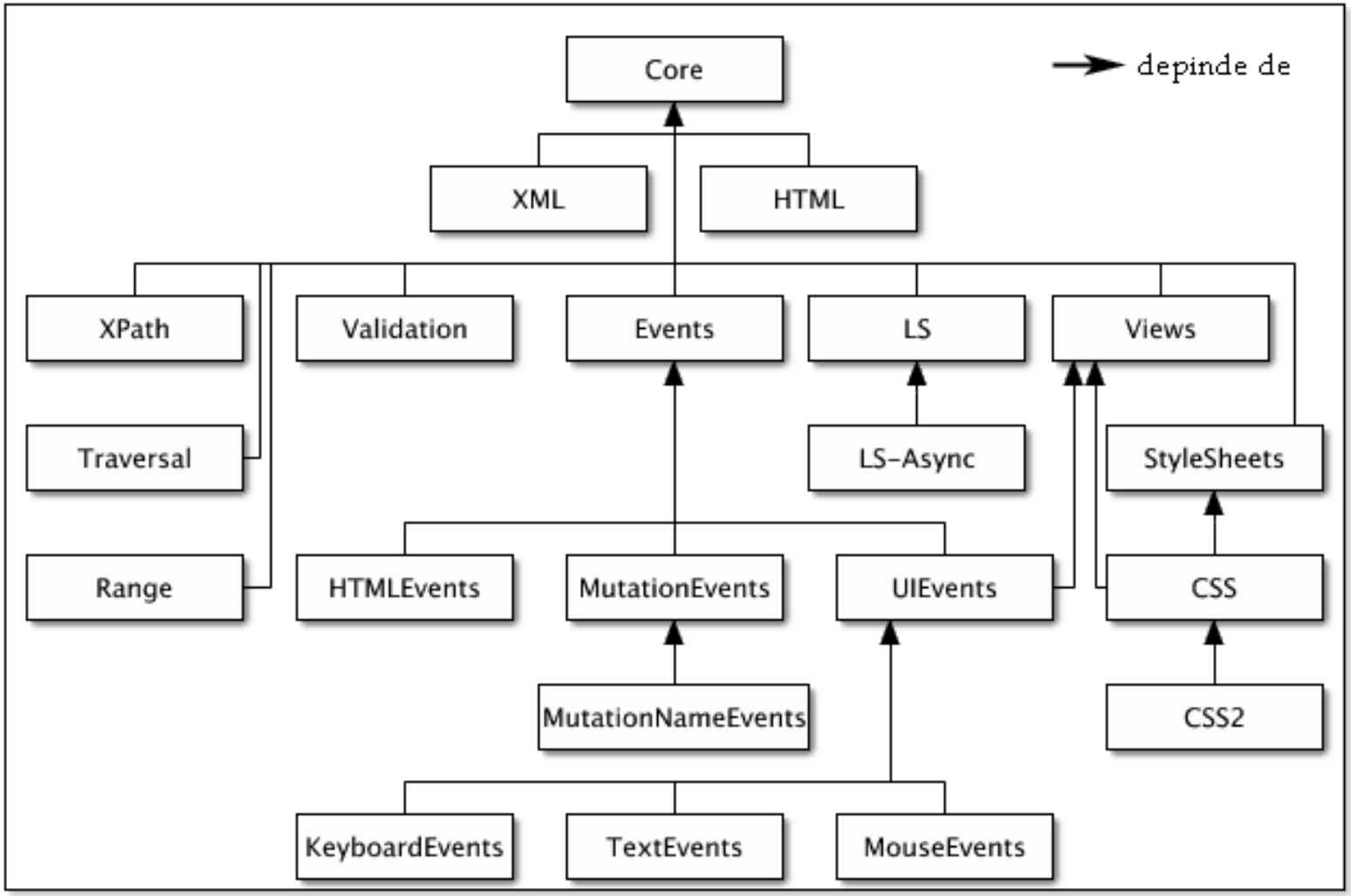
Extinde DOM 2, oferindu-se interfețe pentru manipularea XML via module DOM

un modul pune la dispoziție o facilitate particulară

dom: nivelul 3

Modulul Core include interfețele fundamentale ce trebuie implementate de toate implementările DOM conformându-se standardului

http://www.w3.org/TR/DOM-Level-3-Core



module disponibile: XML, HTML, XPath, Traversal, Range, Validation, Events, Views, Load & Save, Stylesheet,...

dom: nivelul 3

Interfețe noi importante:

DOMStringList, NameList, TypeInfo, UserDataInfo, DOMImplementationList, DOMImplementationSource, DOMLocator, DOMConfiguration

dom: nivelul 3

Interfețe modificate (extinse):

Document, Node, Attr, Element, Text, Entity

dom: nivelul 3

Module DOM 3 standardizate
DOM Load & Save

interfețe puse la dispoziție:
LSParser, LSInput, LSSerializer, LSOutput

http://www.w3.org/TR/DOM-Level-3-LS

dom: nivelul 3

Module DOM 3 standardizate
DOM Validation

oferă funcționalități de creare/editare (automată)
de documente conformându-se
unor scheme de validare

http://www.w3.org/TR/DOM-Level-3-Val

dom: nivelul 4

Unifică DOM3 Core, Element Traversal, Selectors API – level 2, DOM3 Events

apar interfețele ParentNode, ChildNode, Elements,...

suport și pentru specificarea de evenimente proprii via interfața CustomEvent

dom: nivelul 4

Interfața Node este extinsă cu proprietățile firstChild, lastChild, previousSibling, nextSibling și metoda adoptNode ()

dom: nivelul 4

Interfața Element include metodele
getElementsByClassName () și matches ()

dom: nivelul 4

Noi evenimente HTML5 ce pot fi tratate

interacțiune tactilă – www.w3.org/TR/touch-events/
touchstart, touchend, touchmove, touchcancel
+
conectivitatea la rețea: online, offline
diverse operațiuni: redo, undo, drag, drop,...
starea dispozitivului – deviceorientation, devicemotion

...și altele

dom: nivelul 4

Selectors API

acces la diverse date via selectorii CSS cu metodele

query() queryAll() querySelector() querySelectorAll()

```
// toate elementele <li> selectate via CSS (date de tip NodeList)
var elemente = document.querySelectorAll ("ul.resurse>li");
for (var i = 0; i < elemente.length; i++) {
  prelucreaza (elemente.item (i));                    // procesăm fiecare nod
}
```

exemplificare – folosim consola oferită de *browser*:

document.querySelectorAll ("section[id^=\"week\"]:nth-child(even) > h2");

dom: implementări

HXT (*Haskell XML Toolbox*): procesări DOM în Haskell
www.haskell.org/haskellwiki/HXT

JAXP – parte integrantă din J2SE (javax.xml.*)

JDOM – API special pentru Java: **http://www.jdom.org/**

JSXML – bibliotecă JavaScript: **http://jsxml.net/**

QDOM – parte din Qt (C++)

dom: implementări

libxml – API oferit inițial de GNOME: **http://xmlsoft.org/**
baza unor biblioteci pentru C++, Perl, PHP, Python, Ruby,…
folosit și de libxslt

MSDOM – procesări XML pe client/server în C++
sau JScript (IE, IIS+ASP,…); parte din MSXML SDK

Xerces DOM API – platformă XML (C++/Java):
http://xml.apache.org/

dom: implementări

XmlDocument – clasă .NET Framework (C# *et al.*)

xml.dom – modul Node.js: **https://github.com/jindw/xmldom**

XML::DOM – modul Perl pentru DOM,
bazat pe Expat (XML::Parser)

xml.dom – modul din Python, parte a PyXML

avansat

dom: api-uri particulare (exemple)

BBC API – acces la programele transmise (disponibile în formatele XML, JSON, RDF, iCal):
http://www.bbc.co.uk/programmes/developers

Google APIs – suită de API-uri (Java, JS, PHP, Python, Objective-C, .NET) privind serviciile Google publice:
https://developers.google.com/apis-explorer/

node-xmpp – biblioteca Node.js (JavaScript) pentru XMPP
https://npmjs.org/package/node-xmpp

avansat

dom: api-uri particulare (exemple)

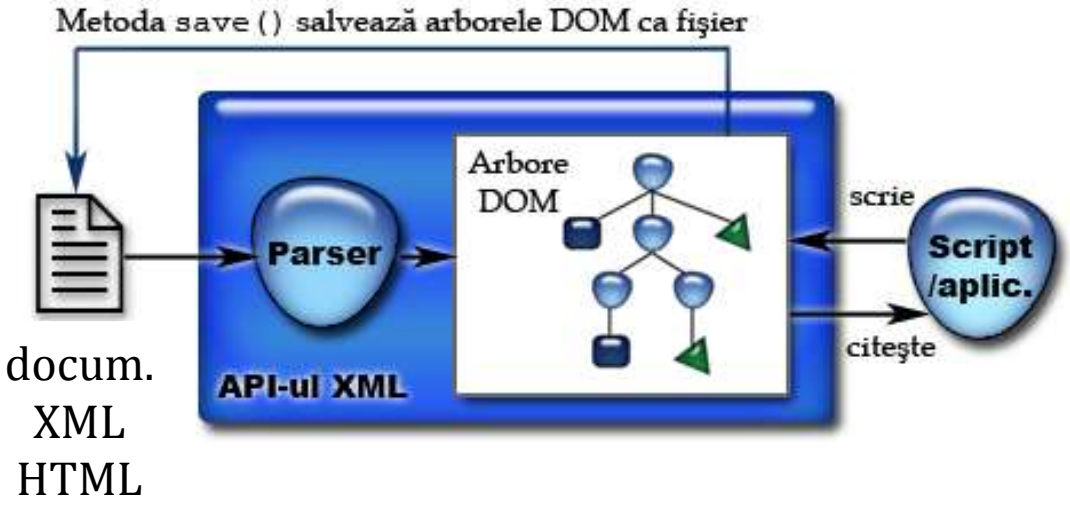
Open Data Protocol – inițiativă Microsoft (implementări în Java, JS, PHP, Ruby, .NET): **http://www.odata.org/**

SVG DOM API – procesarea documentelor SVG în Java (Apache Batik): **http://xmlgraphics.apache.org/batik**

TwiMLT (*Twilio Markup Language*) API – acces la serviciile de telefonie Twilio via biblioteci C++, Erlang, Perl, PHP, Python, Ruby, Scala: **http://www.twilio.com/docs**

dom: implementări

Procesarea documentelor XML/HTML



```php
try {
  $doc = new DomDocument;   // instanțiem un obiect DOM
  $doc->load ("projects.xml");  // încărcăm documentul XML
  // afișăm informații privitoare la proiecte: titlul & clasa (dacă există)
  $projs = $doc->getElementsByTagName("project");
  foreach ($projs as $proj) {     // preluăm nodurile-element <title>
    $titles = $proj->getElementsByTagName("title");
    foreach ($titles as $title) {
      echo "Proiect: " . $title->nodeValue;
    }
    // verificăm dacă există specificată clasa proiectului
    if ($proj->hasAttribute("class")) {
      echo " de clasa " . $proj->getAttribute("class");
    }
  }
} catch (Exception $e) {
   die ("Din păcate, a survenit o excepție.");
}
```

procesări DOM
în limbajul PHP

```
try:
   doc = urllib.urlopen('projects.xml')  # încărcăm documentul
   dom = parse(doc)                      # îl procesăm…
   # parcurgem elementele <project> și oferim informații despre fiecare
   for proiect in dom.getElementsByTagName('project'):
      print 'Proiectul ' + proiect.childNodes[1].firstChild.nodeValue + \
            ' este de clasă: ' + proiect.getAttribute('class')
except Exception:
   print 'Din păcate, a survenit o excepție.'
```

procesări DOM în limbajul Python

http://docs.python.org/2/library/xml.dom

```
# program Perl care contorizează proiectele de clasă 'S'

# instanțiem procesorul XML
$parser = new XML::DOM::Parser;
# procesăm documentul
$doc = $parser->parsefile ('projects.xml');
$proiecte = $doc->getElementsByTagName ('project');
$nr_proiecte = $proiecte->getLength;
$proiecte_clasaS = 0;                          # inițial, 0 proiecte de clasă 'S'
for (my $i = 0; $i < $nr_proiecte; $i++) { # baleiăm toate proiectele
   $proiect = $proiecte->item ($i);          # preluăm clasa proiectului
   $clasa = $proiect->getAttributeNode ('class')->getValue;
   if ($clasa eq 'S') {
     $proiecte_clasaS++;
   }
}
print "\nSunt $proiecte_clasaS proiecte de clasă S.\n";
$doc->dispose ();                              # eliberam memoria
```

procesări DOM
în limbajul Perl

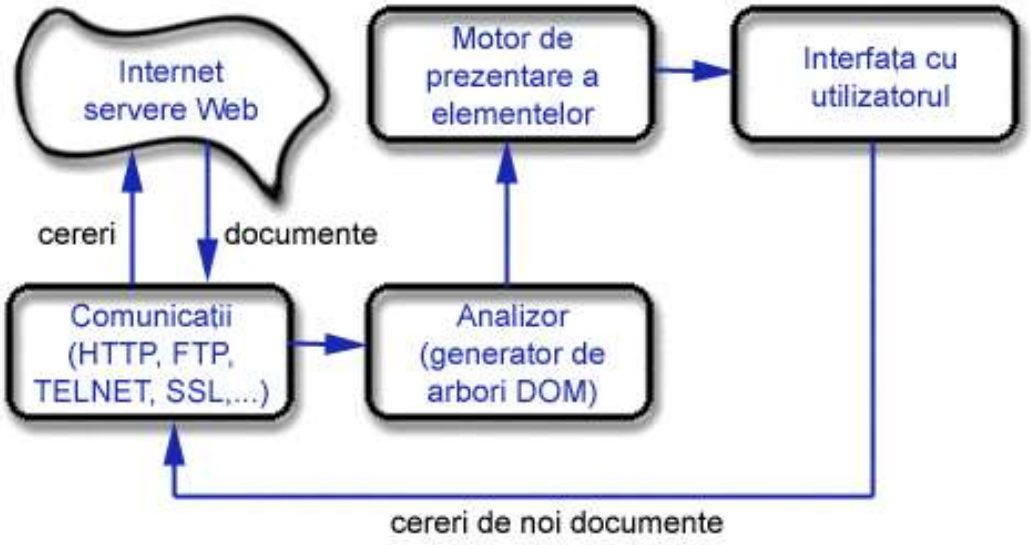
```
try {
  doc = new XmlDocument(); // instanțiem un document XML
  doc.Load("projects.xml");   // pentru a fi încărcat
  // afișăm informații privitoare la proiecte: titlu și clasă
  XmlNodeList projs = doc.GetElementsByTagName("project");
  foreach (XmlElement proj in projs) {
    // selectăm nodurile <title> via o expresie XPath
    XmlNodeList titles = proj.SelectNodes("./title");
    foreach (XmlElement title in titles) {
      Console.Write("Proiect: {0} ", title.InnerXml);
    }
    if (proj.HasAttribute("class") == true) { // există clasa specificată?
       Console.WriteLine("de clasa '{0}'.", proj.GetAttribute("class"));
    }
} catch ( Exception e ) {
   // din păcate, a survenit o excepție
}
```

procesări DOM
în C# (.NET)

dom: demo

dom: browser

Prelucrarea documentelor XML în *browser*



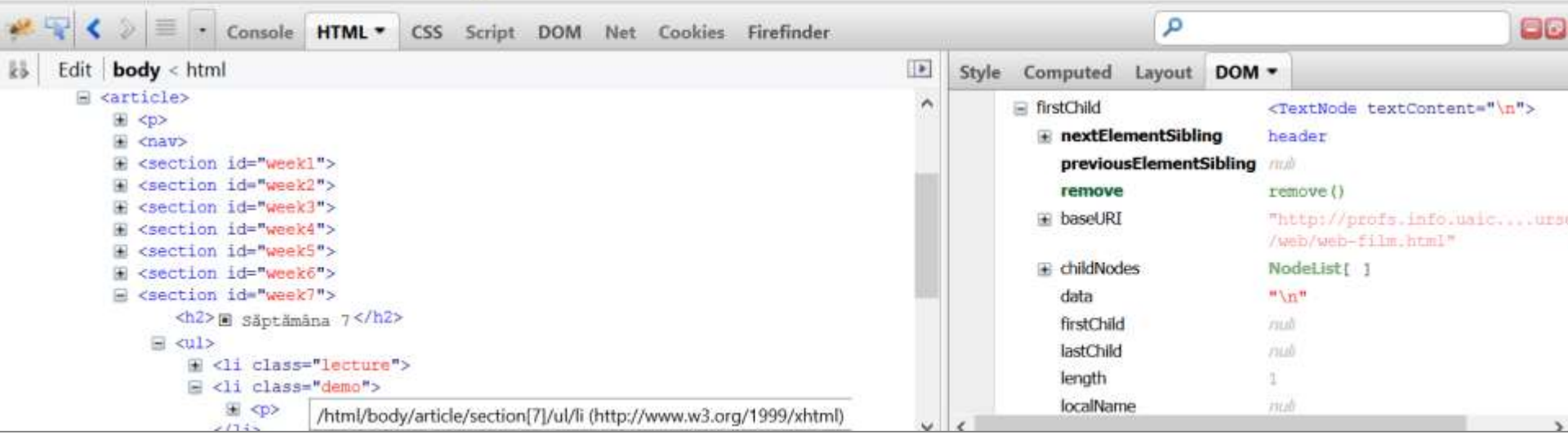
detalii la „Dezvoltarea aplicațiilor Web la nivel de client”

http://profs.info.uaic.ro/~busaco/teach/courses/cliw/web-film.html

- Curs: Modelarea datelor. Extragerea datelor cu XPath. Validări XML via DTD concurs: Upstream Challenge

  date semi-structurate, XML, model, XPath, validare, DTD, exemple

- Demonstraţii: Exemple de validări folosind DTD (arhivă `zip`, 6K)
- Suplimente: Transformări XSL – Exemple XSLT, EXSLT, XPath 2, XSLT 2 (arhivă `zip`, 52K), Validări XML cu XML Schema şi RELAX NG – Exemple XML Schema & RELAX NG (arhivă `zip`, 18K), Baze de date XML. XQuery
- Resurse: FirePath, Firefinder, Declaraţii DTD recomandate, <XMLPlayground/>, Validarea online a documentelor XML, <oXygen /> XML Editor
- Laborator: Programare Web la nivel de server



inspectarea arborelui DOM corespunzător unui document HTML via extensia Firebug: getfirebug.com

dom: browser

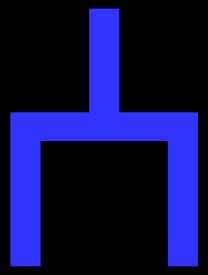
Vizualizarea/procesarea documentelor HTML și XML – fără validare – se realizează via DOM în JavaScript (ECMAScript) interpretat de navigatorul Web

dom: browser

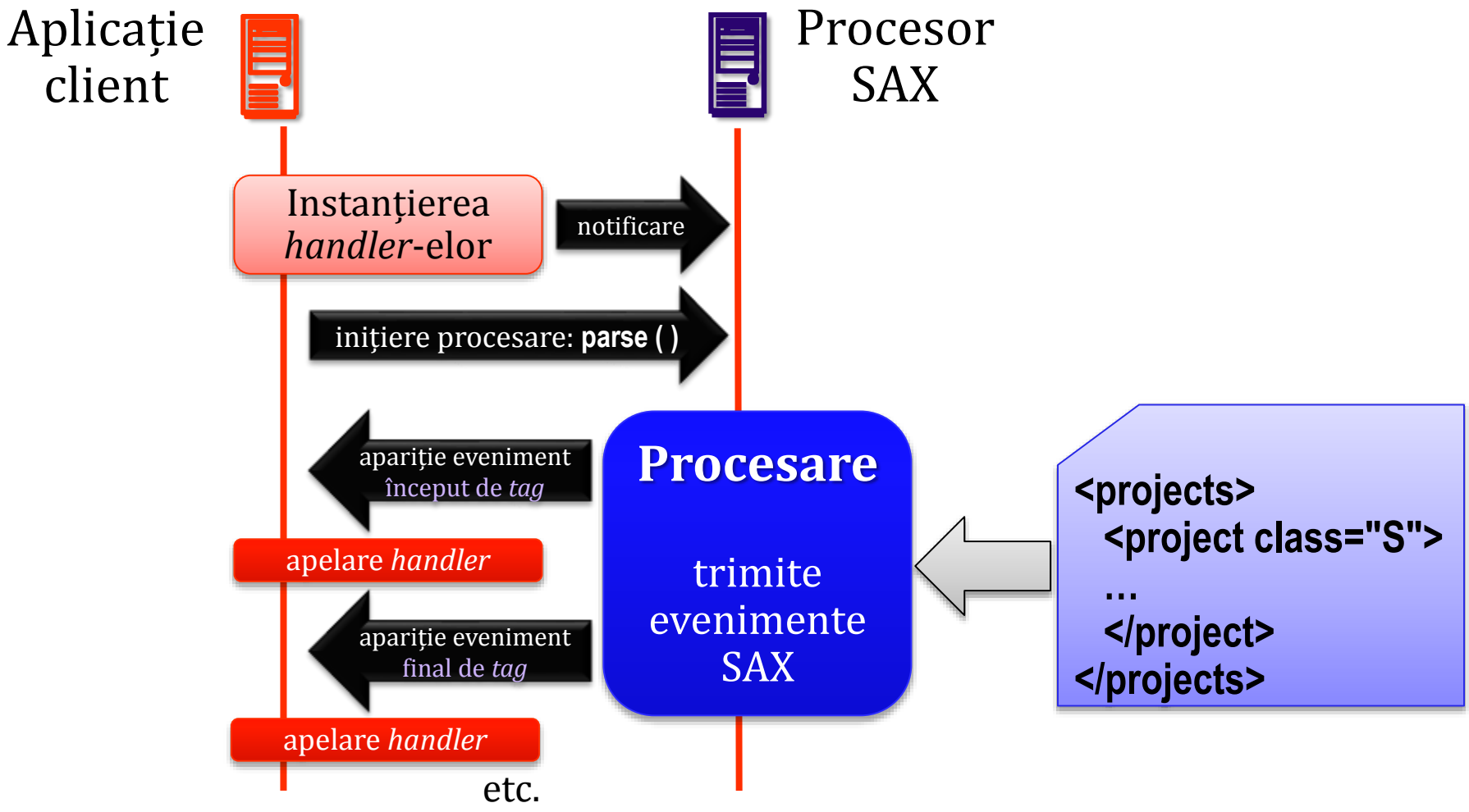
Se oferă suport și pentru transfer asincron de date între client (*browser*) și server Web

AJAX – *Asynchronous JavaScript And XML*
via obiectul **XMLHttpRequest**

rezumat



modelul DOM:
caracterizare, niveluri de specificare, exemple



episodul viitor:

procesări XML via SAX + prelucrări simplificate