Conținut şi design în programarea web. CSS. DHTML.

Ciprian Dobre
ciprian.dobre@cs.pub.ro

Obiective

- În cadrul cursului prezentăm o noțiuni introductive legate de stil, Cascading Style Sheets şi formatarea documentelor şi paginilor web

Sumar

- XML
- XTHML
- Introducere în CSS
- Greşeli de stil în HTML

XML

eXtensible Markup Language

HTML şi XML, I

XML = eXtensible Markup Language

HTML este folosit pentru formatarea textului a.î. Acesta să fie afişar utilizatorilor

XML este folosit pentru formatarea datelor a.î. să fie procesate de calculatoare

HTML descrie atât structura (e.g. <p>, <h2>,) cât şi aparența (e.g.
, <font>, <i>)

XML describe doar conținut sau “sens”

HTML foloseşte un set fix, nemodificabil de tag-uri

În XML se folosesc propriile tag-uri

HTML şi XML, II

- HTML şi XML arată similar deoarece ambele sunt limbaje SGML (SGML = Standard Generalized Markup Language)
  - Atât HTML cât şi XML folosesc elemente încadrate de tag-uri (e.g. <body>This is an element</body>)
  - Ambele folosesc atribute de tag-uri (e.g., <font face="Verdana" size="+1" color="red">)
  - Ambele folosesc entități (<, >, &, ", ')
- Mai exact,
  - HTML este definit în SGML
  - XML este un subset (restrâns) al SGML

HTML şi XML, III

- HTML este adresat oamenilor
  - HTML describe pagini web
  - Nu doriți să vedeți mesaje de eroare despre paginile pe care le vizitați
  - Browserele ignoră şi/sau corectează atâtea erori HTML cât se poate
- XML este folosit de computere
  - XML describe date
  - Regulile sunt stricte şi nu sunt permise erori
    - Din acest punct de vedere XML seamană cu un limbaj de programare
  - Versiunile curente ale majorității browserelor pot afişa XML
    - Totuşi suportul browserelor pentru XML e destul de redus

Tehnologii înrudite XML

- DTD (Document Type Definition)şi XML Schemas sunt folosite pentru definirea tag-urilor legale XML şi a atributelor acestora pentru scopuri particulare
- CSS (Cascading Style Sheets) descrie cum sunt afişate HTML şi XML în browser
- XSLT (eXtensible Stylesheet Language Transformations) şi XPath sunt folosite pentru translatarea de la o formă de XML la o alta
- DOM (Document Object Model), SAX (Simple API for XML, and JAXP (Java API for XML Processing) sunt API-uri pentru parsare XML

Exemplu de document XML

```
<?xml version="1.0"?>
<weatherReport>
   <date>7/14/97</date>
   <city>North Place</city>, <state>NX</state>
   <country>USA</country>
   High Temp: <high scale="F">103</high>
   Low Temp: <low scale="F">70</low>
   Morning: <morning>Partly cloudy, Hazy</morning>
   Afternoon: <afternoon>Sunny &amp;amp; hot</afternoon>
   Evening: <evening>Clear and Cooler</evening>
</weatherReport>
```

Sursa: **XML: A Primer**, de Simon St. Laurent

Structura

- Un document XML poate începe cu una sau mai multe instrucțiuni de procesare (PIs) sau directive:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/css"
href="ss.css"?>
```

- După directivă trebuie să existe exact un *singur* tag, numit elementul *root*, ce conține restul documentului XML:

```
<weatherReport>
    ...
</weatherReport>
```

Elementele de bază XML

- Un document XML mai este construit din:
  - elemente: high în <high scale="F">103</high>
  - tag-uri, în perechi: <high scale="F">103</high>
  - atribute: <high scale="F">103</high>
  - entități: <afternoon>Sunny & hot</afternoon>
  - date de tip caractere ce pot fi:
    - parsate (procesate ca XML)—modalitatea default
    - neparsate (toate caractere sunt de sine stătătoare)

Elemente şi atribute

- Atributele şi elementele sunt oarecum interschimbabile
- Exemplu folosind doar elemente:

```
<name>
   <first>David</first>
   <last>Matuszek</last>
</name>
```

- Exemplu folosind atribute:

```
<name first="David" last="Matuszek"></name>
```

- Atributele conțin adesea şi metadate, precum ID-uri unice
- Generic, browserele afişează doar elemente (valori închise de tag-uri), nu tag-uri şi atribute

XML bine format

- Orice elemente trebuie să aibă *atât* un tag de start, cât şi un tag de end, e.g. <name> … </name>
  - Dar elementele empty pot fi abreviate: <break />.
  - Tag-urile XML sunt case sensitive
  - Tag-urile XML nu pot începe cu literele xml
- Elementele trebiue să fie corect imbricate, e.g. *nu* bold and italic
- Orice document XML trebuie să aibă unul şi numai un element root
- Valorile atributelor trebuie să fie încadrate de ghilimele sau apostroafe, e.g. <time unit="days">
- Datele caracter nu pot conține < sau &

Reguli XML

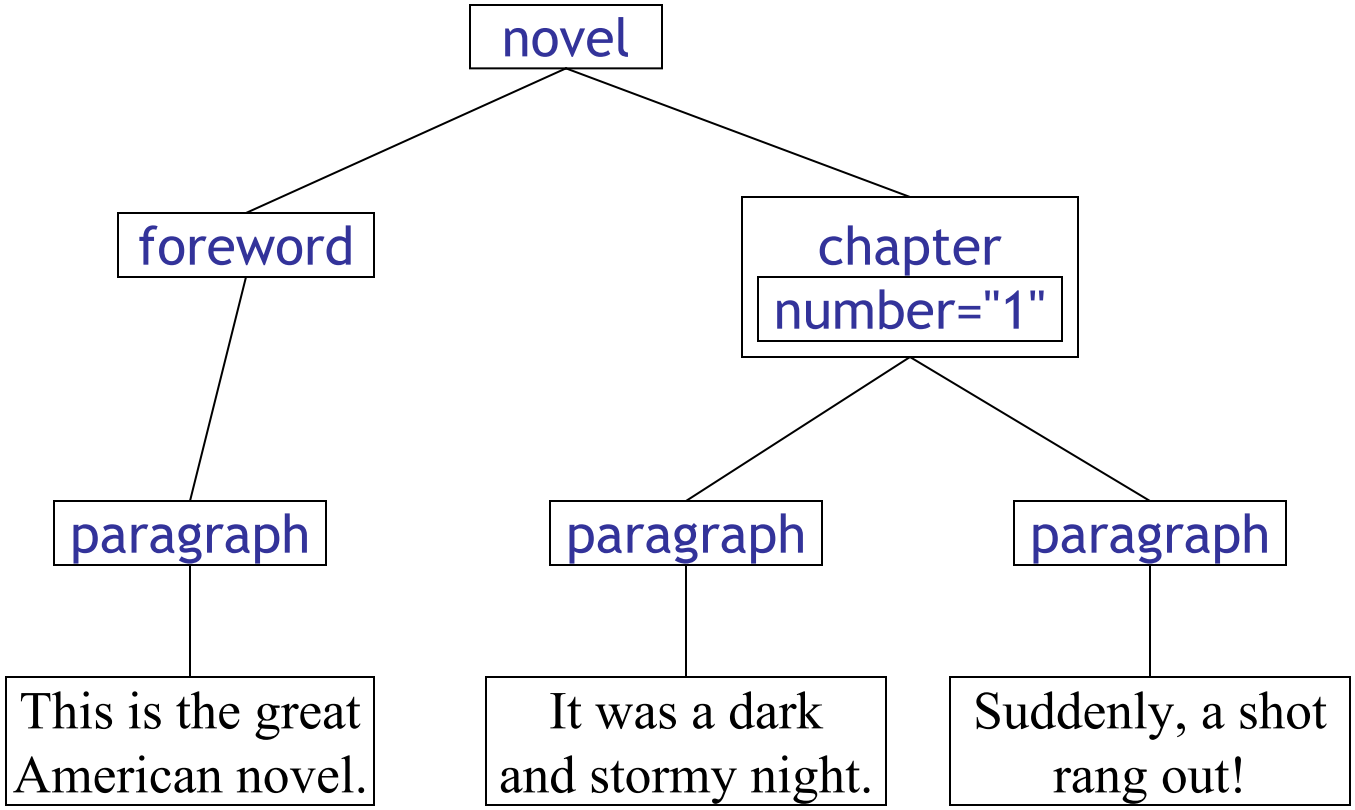
- Începe cu <?xml version="1"?>
- XML este case sensitive
- Trebuie să aveți un singur element root ce cuprinde tot documentul XML
- Orice element trebuie să aibă un tg de închidere
- Elementele trebuie corect imbricate
- Valorile atributelor trebuie să fie încadrate de ghilimele sau apostroafe
- Există cinci entități predefinite

Un exemplu de document bine structurat

```
<novel>
  <foreword>
    <paragraph> This is the great American
novel.
    </paragraph>
</foreword>
  <chapter number="1">
    <paragraph>It was a dark and stormy night.
    </paragraph>
    <paragraph>Suddenly, a shot rang out!
    </paragraph>
  </chapter>
</novel>
```

XML ca un arbore

- Un document XML reprezintă o ierarhie, o ierarhie este un tree



XML Valid

- Puteți crea propriile tag-uri şi atribute XML, *dar...*
  - ...orice program ce *foloseşte* XML-ul trebuie să ştie la ce să se aştepte!
- Un DTD (Document Type Definition) defineşte ce tag-uri sunt legale şi unde pot acestea apărea în cadrul XML
- Un document XML *nu necesită* un DTD
- XML este bine structurat dacă respecta regulile amintite anterior
- În plus, un XML este valid dacă declară un DTD şi este conform cu respectivul DTD
- Un DTD poate fi inclus în XML, dar tipic se foloseşte ca document separat
- Erorile dintr-un document XML vor *opri* programele XML
- Alternative la DTD-uri sunt XML Schemas şi RELAX NG

Vizualizarea XML

- XML este proiectat pentru a fi procesat de către programele software, nu pentru a afişa date
- Totuşi aproape toate browserele pot afişa documente XML
  - Ele nu vor fi afişate în aceeaşi manieră
  - Nu vor fi afişate deloc dacă conțin erori
- Reminder:

  HTML is designed to be *viewed*,

  XML is designed to be *used*

Standarde extinse de documente

- Puteți să vă definiți propriile tag-uri XML, dar există seturi deja disponibile:
  - XHTML: HTML redefinit pentru XML
  - SMIL: Synchronized Multimedia Integration Language
  - MathML: Mathematical Markup Language
  - SVG: Scalable Vector Graphics
  - DrawML: Drawing MetaLanguage
  - ICE: Information and Content Exchange
  - ebXML: Electronic Business with XML
  - cxml: Commerce XML
  - CBL: Common Business Library

XHTML

http://www.w3schools.com/xhtml/

HTML 4

- HTML 4 extinde HTML cu mecanisme adecvate pentru*
  - style sheets,
  - scripting,
  - frames,
  - embedding objects,
  - improved support for right to left and mixed direction text
  - richer tables
  - enhancements to forms, offering improved accessibility for people with disabilities.

*http://www.w3.org/TR/1999/REC-html401-19991224/intro/intro.html#h-2.3

Ce este XHTML?

- XHTML = Extensible Hypertext Markup Language
  - XHTML urmăreşte să înlocuiască HTML
  - XHTML este aproape identic cu HTML 4.01
  - XHTML este o versiune mai strictă şi mai curată a HTML
- XML (Extensible Markup Language) este un limbaj de markup proiectat pentru descrierea de *date*
  - XHTML este HTML redefinit ca o aplicație XML
  - XHTML este o “punte” între HTML şi XML

Problema cu HTML

- HTML a fost conceput ca o modalitate de a descrie *structura* unui document, cu tag-uri pentru a indica headere, paragrafe, etc.
- Apare nevoia de control asupra *aparenței* documentelor, motiv pentru care HTML a fost îmbogățit cu tag-uri pentru controlul font-urilor, aliniatelor, etc.
- Rezultatul este un limbaj de markup ce face ambele, dar nu este prea bun la nici unul

HTML vs. XML

XML arată mult ca HTML, dar--

HTML foloseşte un set fix de tag-uri	În XML puteți folosi propriile tag-uri (şi defini ce reprezintă ele într-un document separat)
HTML este proiectat pentru afişarea datelor în formă umană	XML este proiectat pentru descrierea datelor pentru computere
Browser-ele sunt foarte tolerante cu erorile în HTML	Documentele XML trebuie sp fie bine formate (corect sintactic)
Toate browserele pot afişa HTML	Toate browserele moderne afişează XML, dar în diverse moduri

Tag-uri XML

- Tags
  - Reprezintă Metainformații incluse în text
    - Similare ca formă cu tag-urile HTML
    - Diferența între tag-urile HTML şi XML : cele HTML conțin informații de reprezentare a datelor (ex: <B>), în timp ce tag-urile XML conțin informații de structurare şi semantică a datelor
  - Tag-urile XML sunt case-sensitive
  - Pot conține text sau alte tag-uri
  - Fiecare trebuie să aibă un tag de sfârşit:
  - <tag> </tag>
  - O pereche de tag-uri fără conținut se poate scrie ca şi <tag />
- Tag Attributes
  - Defineşte perechi name-value în interiorul unui tag
    - <dot x="72" y="13" />

Codificarea documentelor XML

- Un document XML începe cu o declarație:
- <?xml version='1.0' encoding='utf-8'?>
- Forma arborescentă:
  - Există exact un element rădăcină
  - Alte elemente sunt încuibate
  - Prin element se înțelege o secvență cuprinsă între 2 tag-uri pereche
- <person>
- <firstname>Ion</firstname>
- <lastname>Popescu</lastname>
- <age>30</age>
- <ssn>2711130345678</ssn>
- </person>

Documente XML: Well-formed

- "well-formed": un document corect din punctul de vedere al regulilor sintactice generale XML
  - are exact un element rădăcină
  - fiecare element are un tag de sfârşit
  - elementele sunt încuibate corect
  - valorile atributelor sunt între ghilimele

Documente XML: Valide
XML Schema

- "Valid": un document care respectă anumite reguli impuse structurii
- Metode de specificare formală a structurii unui document (unei clase de documente) XML:
  - XML DTD (Data Type Definition):
  - XML Schema (XSD): un limbaj ce impune constrângeri asupra structurii documentelor XML
- Pentru o clasă de documente XML se pot impune reguli privitoare la:
  - Ce tag-uri sunt permise, în ce ordine pot sa apară, de câte ori, ce atribute pot să aibă, de ce tipuri, etc.
- Parserele XML cu validare verifică respectarea constrângerilor impuse de o schemă specificată
- XML Schema Tutorial: http://www.w3schools.com/schema/default.asp

Exemplu 1 - XML Schema

- Pentru reprezentarea unui set de puncte în plan se stabilesc următoarele reguli:
  - Elementul rădăcină este dots
- <xs:element name="dots">
  - Acesta poate conține un număr oarecare de elemente de tip dot
    - Este un tip complex pentru că conține alte elemente
- <xs:complexType>
  - Conține o secvență de alte elemente
- <xs:sequence>
  - Fiecare element dot are 2 atribute, x şi y, cu valori întregi
- <xs:attribute name="x" type="xs:integer" />

Exemplu 1 XML Schema

dots.xsd

```
<?xml version="1.0"?>
<xs:schema xmlns:xs=http://www.w3.org/2001/XMLSchema>
<xs:element name="dots">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="dot" maxOccurs="unbounded">
        <xs:complexType>
              <xs:attribute name="x" type="xs:integer" use="required"/>
              <xs:attribute name="y" type="xs:integer" use="required"/>
      </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Exemplu: document XML cu schema

dots.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<dots xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="dots.xsd">>
<dot x="32" y="100" />
<dot x="17" y="14" />
<dot x="18" y="58" > </dot>
</dots>
```

De la HTML la XHTML, I

- Elementele XHTML trebuie să fie corect imbricate

  `<b><i>bold and italic</b></i>` este *greşit*

- Documentele XHTML trebuie să fie bine formate

```
<html>
<head> ... </head>
<body> ... </body>
</html>
```

- Numele de tag-uri trebuie să fie lowercase
- *Toate* elementele XHTML trebuie să fie închise
  - Dacă un tag HTML nu este un container el se închide:
    `<br />`, `<hr />`, `<img src="smile.gif" />`

De la HTML la XHTML, II

- Numele de atribute trebuie să fie *lower case*
  - Exemplu: <table width="100%">
- Valorile atributelor trebuie să fie încadrate de “
  - Example: <table width="100%">
- Minimizarea atributelor este interzisă
  - Exemplu: <frame noresize="noresize">, nu poate fi abreviată la <frame noresize>
- Atributul id înlocuieşte atributul name
  - Greşit:
  - Corect:
  - Cel mai bine:

SGML şi DTD-uri

- SGML = Standard Generalized Markup Language
- HTML, XHTML, XML şi multe alte limbaje de markup sunt definite în SGML
- Un DTD, sau “Document Type Definition” descrie sintaxa ce trebuie folosită pentru un anumit document
- Există trei tipuri diferite de DTD-uri pentru XHTML—puteți lucra cu oricare
  - Ele sunt *publice* pe web
  - Documentul XHTML trebuie să înceapă cu o referință la unul dintre aceste DTD-uri

Declarația DOCTYPE, I

- Orice document XHTML trebuie să înceapă cu una dintre declarațiile the DOCTYPE (DTD-uri):
  - <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
  - <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
  - <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">

Declarația DOCTYPE, II

- Cele trei DTD-uri sunt:
  - Strict
    - Folosit pentru markup curat, fără informații de afişare (fără informații legate de font, color sau size)
    - Folosit cu CSS (Cascading Style Sheets) dacă se doreşte definirea felului în care arată documentul
  - Transitional
    - Folosit cu HTML standard şi/sau cu CSS
    - Permite elemente HTML depricate
  - Frameset
    - Folosit dacă documentul include frame-uri HTML

Un exemplu XHTML

- <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

```
<html>
    <head>
        <title>A simple document</title>
    </head>
    <body>
        <p>A simple paragraph.</p>
    </body>
</html>
```

Instrumente

- Dave Raggett's HTML TIDY http://www.w3.org/People/Raggett/tidy/ este un instrument free UNIX pentru verificarea şi curățarea paginilor HTML
- W3C HTML Validation Tool http://validator.w3.org/ este un formular HTML pentru verificarea (dar nu şi corectarea) documentelor HTML şi XHTML

Vocabular

- SGML: Standard Generalized Markup Language
  - HTML: Hypertext Markup Language
  - XHTML: eXtensible Hypertext Markup Language
  - XML: eXtensible Markup Language
- DTD: Document Type Definition

Quiz

- Care este diferența dintre document XML bine format şi document XML valid?
- Ce este un DTD?
- Pe ce standard HTML este bazat XHTML?
- Enumerați patru diferențe între HTML şi XHTML.

CSS

Problema cu HTML

- HTML a fost de la inceput conceput pentru a descrie *continutul* unui document
- Autorii de pagini web nu aveau nevoie sa descrie layout-ul -- browserul avea grija de acest aspect
- Aceasta reprezinta o abordare inginereasca potrivita, insa nu satisface stilistii si “artistii”
  - Chiar oameni ce aveau nevoie sa spuna mai multe simteau nevoia de un control mai bun al modului de aparitie al propriilor pagini web
- Ca rezultat, HTML a inceput sa incorporeze din ce in ce mai multe tag-uri ce sunt folosite pentru controlul *afisarii*
  - Continul si modul de afisare al informatilor au devenit din ce in ce mai mult interconectate
  - Browsere diferite afiseaza lucruri in moduri diferite, ceea ce reprezinta o problema reala atunci cand modul de prezentare al informatiilor e chiar mai important decat informatia propriu-zisa

Cascading Style Sheets

- Un Cascading Style Sheet (CSS) descrie modul de prezentare a unei pagini HTML intr-un document separat
- CSS are urmatoarele avantaje:
  - Permite separarea continutului de prezentare
  - Permite definirea prezentarii si a layout-ului tuturor paginilor dintr-un site web intr-un singur loc
  - Poate fi folosit atat pentru pagini HTML cat si XML
- CSS are si un dezavantaj:
  - Unele browsere nu il suporta in totalitate

Sintaxa CSS

- Sintaxa CSS este simpla – este doar un fisier continand o lista de selectori (pentru alegerea tag-urilor) si descriptori (pentru a specifica ce dorim sa facem cu respectivele tag-uri):
  - Exemplu: h1 {color: green; font-family: Verdana} specifica ca orice este inclus in tag-uri h1 (HTML heading level 1) trebuie afisat folosind font de tip Verdana si colorat in verde
- Un fisier CSS reprezinta doar o lista de astfel de perechi selector/descriptor
  - Selectorii pot fi simple tag-uri HTML sau XML, dar CSS permite de asemenea definirea altor moduri de combinare a tag-urilor
  - Descriptorii sunt definiti chiar in CSS

Sintaxa CSS

- Sintaxa generala ese:

  selector { ***property***: ***value*** }

  – sau

  selector, . . . , ***selector*** {
  property: ***value***;
  . . .
  property: ***value***
  }

  – unde

  - *selector* reprezinta tag-ul ce este afectat de stil (selectorul este case-sensitive daca si numai daca limbajul de descriere a documentului este case-sensitive)
  - *property* si *value* descriu modul de afisare al respetivului tag
  - Spatiile dupa virgule si punct si virgule sunt optionale
  - Un punct si virgula trebuie sa fie folosit *intre* perechi property:value, dar dupa ultima pereche punct si virgula devine caracter optional

Exemplu de CSS

```
/* This is a comment */
h1,h2,h3 {font-family: Arial, sans-serif;}   /* use 1st available font */
p, table, li, address {                      /* apply to all these tags */
  font-family: "Courier New";                /* quote values containing spaces */
  margin-left: 15pt;                         /* specify indentation */
}
p, li, th, td {font-size: 80%;}              /* 80% of size in containing element */
th {background-color:#FAEBD7}  /* colors can be specified in hex */
body { background-color: #ffffff;}
h1,h2,h3,hr {color:saddlebrown;} /* adds to what we said before */
a:link    {color:darkred}                    /* an unvisited link */
a:visited {color:darkred}                    /* a link that has been visited */
a:active  {color:red}                        /* a link now being visited */
a:hover   {color:red}                        /* when the mouse hovers over it */
```

Selectori

- Un tag XML sau HTML poate fi folosit ca un simplu element selector:

  ```
  body { background-color: #ffffff }
  ```

- Putem insa folosi selectori multipli:

  ```
  em, i {color: red}
  ```

  Putem repeta selectorii:

  ```
  h1, h2, h3 {font-family: Verdana; color: red}
  h1, h3 {font-weight: bold; color: pink}
  ```

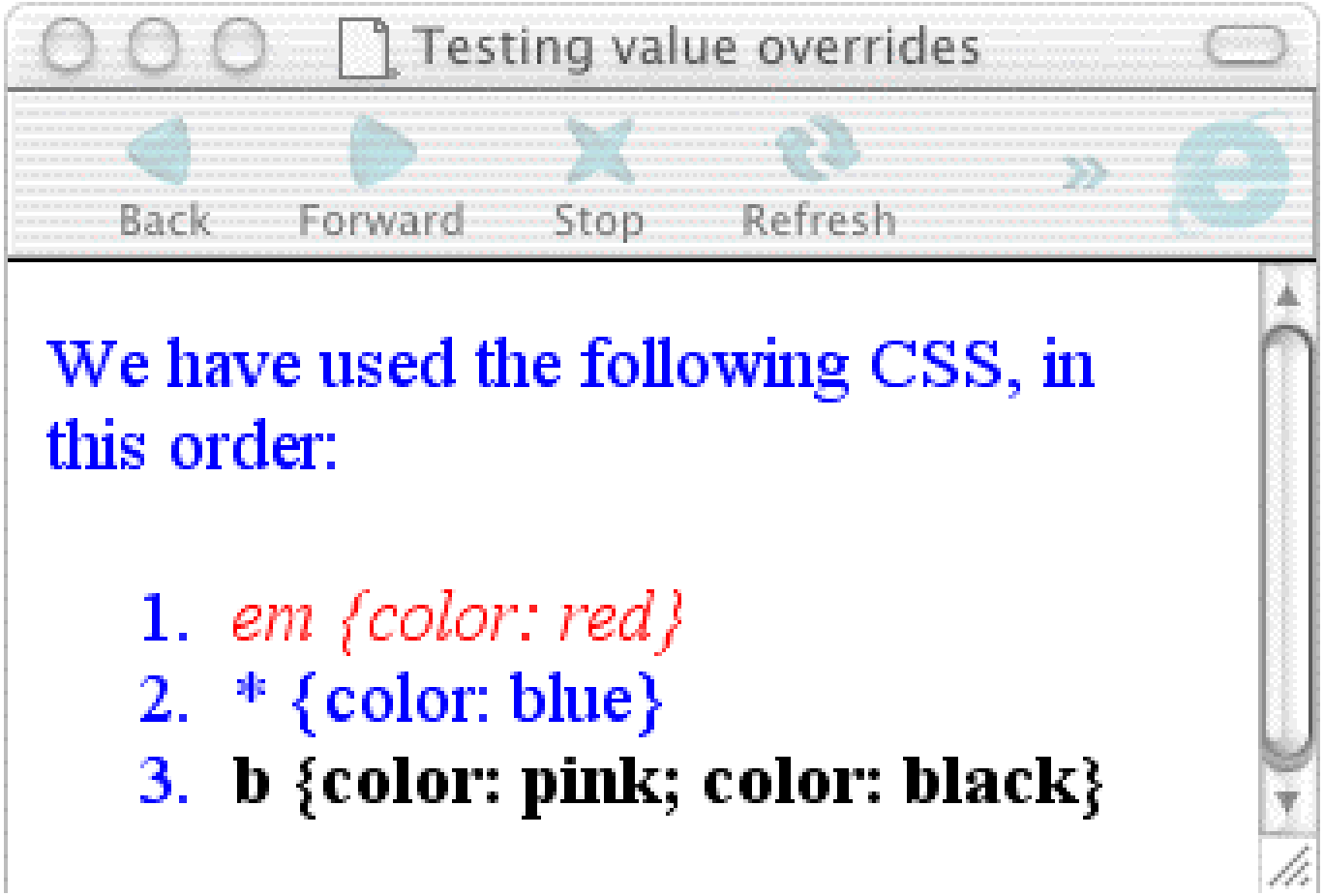
  - Atunci cand valorile nu coincid, ultima declaratie suprascrie declaratiile anterioare

- Selectorul universal * se aplica oricarui si tuturor elementelor:

  ```
  * {color: blue}
  ```

  - Atunci cand valorile nu coincid, selectorii mai specifici suprascriu comportamentul selectorilor mai generali (deci elementele em vor fi in continuare rosii)

Exemplu de suprascriere



Selectori

Un selector descendent alege un tag avand un ancestor corespunzator:

- p code { color: brown }
- selecteaza code daca este folosit in interiorul unui paragraf

- Un selector copil > alege un tag avand un parinte corespunzator:

  h3 > em { font-weight: bold }

  selecteaza un em doar daca parintele imediat este h3

- Un selector adiacent alege un element ce imediat urmeaza altuia:

  b + i { font-size: 8pt }
  Exemplu: I'm bold and I'm italic
  Rezultatul va arata astfel: **I'm bold and** *I'm italic*

Selectori

- Un selector simplu de atribut permite alegerea elementelor ce au un anumit atribut, indiferent de valoarea acestuia:
  - Sintaxa: ***element*[*attribute*] { ... }**
  - Exemplu: table[border] { ... }
- Un selector de atribut valoare permite alegerea elementelor ce au un anumit atribut avand o anumita valoare:
  - Sintaxa: ***element*[*attribute*="*value*"] { ... }**
  - Exemplu: table[border="0"] { ... }

Valori

- Sintaxa unei reguli CSS este: ***selector***, . . . , ***selector*** { ***property***: ***value***; . . . ***property***: ***value*** }
- Valoarea este orice apare intre doua puncte si punct si virgula (sau acolada)
- Exemplu: * {font-family: Trebuchet, Verdana, sans-serif;}
  - Aceasta inseamna ca trebuie folosit font Trebuchet pentru orice, daca este disponibil; altfel, se foloseste font, daca este disponibil; altfel foloseste orice font sans serif pe care browserul il foloseste
- section {border: thin solid blue;}
  - Aceasta inseamna ca trebuie pusa o bordura in jurul elementelor section; bordura trebuie sa fie subtire *si* solida *si* albastra

Atributul class

- Atributul class permite existenta mai multor stiluri diferite pentru acelasi element
  - In fisierul de “style sheet”:
    ```
    p.important {font-size: 24pt; color: red}
    p.fineprint {font-size: 8pt}
    ```
  - In HTML:
    ```
    <p class="important">The end is nigh!</p>
    <p class="fineprint">Offer ends 1/1/97.</p>
    ```
- Pentru definirea unui selector ce se aplica oricarui element ce are o anumita clasa se omite numele tag-ului (dar se pastreaza punctul):

  ```
  .fineprint {font-size: 8pt}
  ```

Atributul id

- Atributul id este definit similar atributului class, dar foloseste # in loc de .
  - In style sheet:
    p#important {font-style: italic} sau
    # important {font-style: italic}
  - In HTML:
    <p id="important">
- class si id pot fi ambele folosite si nu e obligatoriu sa aiba nume diferite:
  <p class="important" id="important">

div si span

- div si span sunt elemente HTML al caror unic scop este acela de a mentine informatie CSS
- div asigura existenta unei linii noi inainte si dupa (similar unui paragraf); span nu
- Exemplu:
  - CSS: div {background-color: #66FFFF}
    span.color {color: red}
  - HTML: This div is treated like a paragraph, but this span is not.

Folosirea de style sheet-uri

- Sunt trei moduri de folosire a unui CSS:
  - Style sheet extern
    - Cea mai puternic abordare
    - Se aplica atat pentru HTML cat si pentru XML
    - Toate elementele CSS pot fi folosite
  - Style sheet embedded
    - Se aplica doar pentru HTML, *nu* si pentru XML
    - Toate elementele CSS pot fi folosite
  - Styles inline
    - Se aplica doar pentru HTML, *nu* si pentru XML
    - Forma limitata de sintaxa CSS

External style sheets

- In HTML, in interiorul elementului <head>:
  <link REL="STYLESHEET" TYPE="text/css" HREF="***Style Sheet URL***">

- In prologul unui document XML:
  <?xml-stylesheet href="***Style Sheet URL***" type="text/css"?>

- Nota: "text/css" reprezinta tipul MIME

Embedded style sheets

- In HTML, in interiorul elementului <head>:

```
<style TYPE="text/css">
    <!--
    CSS Style Sheet
    -->
</style>
```

- Nota: Incapsularea style sheet-ului intr-un comentariu reprezinta un artificiu de ascundere a informatiei de browsere mai vechi ce nu inteleg sintaxa CSS

Inline style sheets

- Atributul STYLE poate fi adaugat oricarui element HTML:

  <*html-tag* STYLE="*property*: *value*"> sau

  <*html-tag* STYLE="*property*: *value*;
  property: *value*; ...; *property*: *value*">

- Avantaj:
  - Folositor daca dorim doar o mica modificare de stil
- Dezavantaje:
  - Mix de informatii de afisaj in HTML
  - Ascunde si ingreuneaza vibilitatea codului HTML
  - Nu se pot folosi toate elementele CSS

Ordinea de cascadare

- Stilurile vor fi aplicate unui HTML in urmatoarea ordine:
  1. Stilul implicit al browser-ului
  2. External style sheet
  3. Internal style sheet (in interiorul tag-ului <head>)
  4. Inline style (in interiorul altor elemente, cele mai din afara mai intai)
- Cand elementele de stil ajung sa fie in conflict, cel mai “apropiat” (mai recent aplicat) stil castiga

Exemplu de ordine de cascadare

- External style sheet:

```
h3 {   color: red;
       text-align: left;
       font-size: 8pt
}
```

- Internal style sheet:

```
h3 {   text-align: right;
       font-size: 20pt
}
```

- Atributele rezultante:

```
color: red;
text-align: right;
font-size: 20pt
```

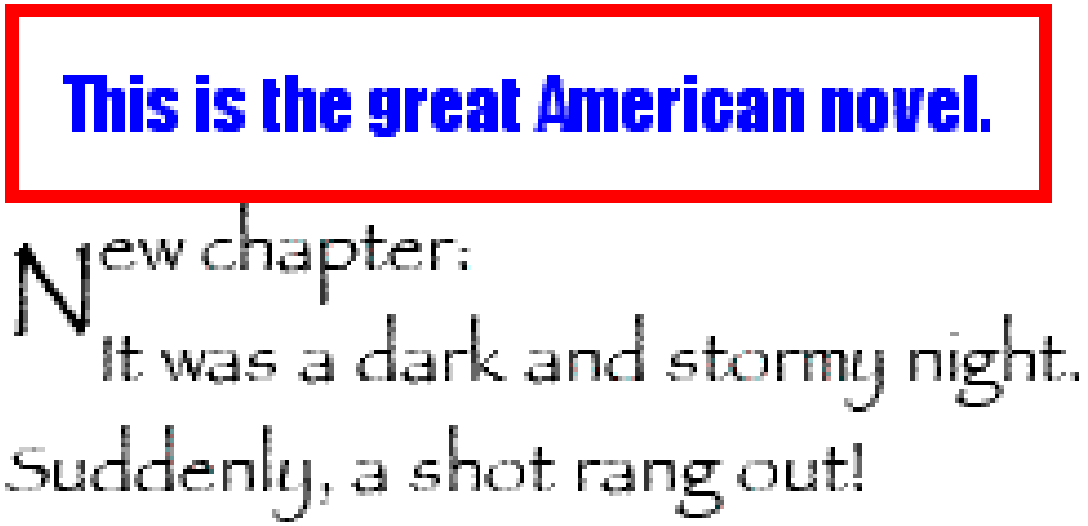
Un exemplu XML

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE novel SYSTEM "novel.dtd">
<?xml-stylesheet href="styles.css" type="text/css"?>
<novel>
  <foreword>
    <paragraph>This is the great American novel.</paragraph>
  </foreword>
  <chapter>
    <paragraph>It was a dark and stormy night.</paragraph>
    <paragraph>Suddenly, a shot rang out!</paragraph>
  </chapter>
</novel>
```

Exemplu: CSS

```
chapter {font-family: "Papyrus", fantasy}
foreword > paragraph {border: solid red; padding: 10px}
novel > foreword {font-family: Impact; color: blue}
chapter {display: block}
chapter:first-letter {font-size: 200%; float: left}
paragraph {display: block}
chapter:before {content: "New chapter: "}
```

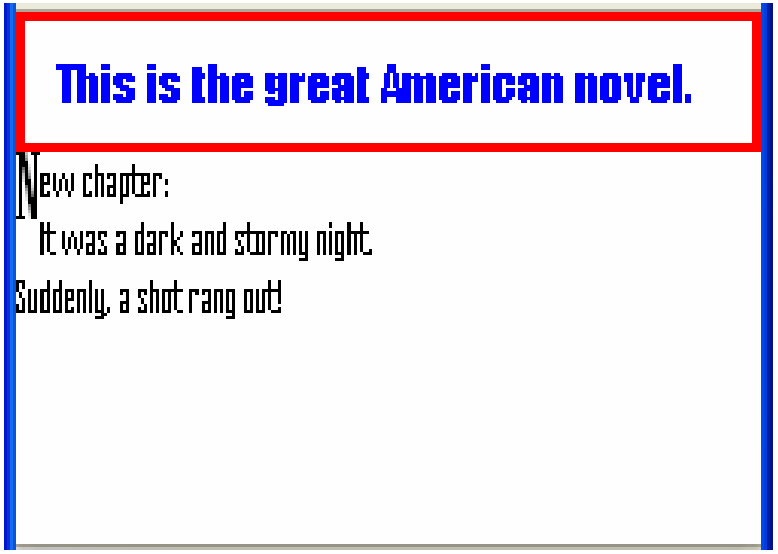
Rezultatul



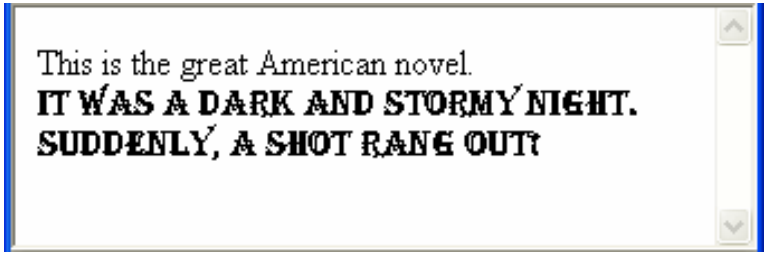
- Acesta este rezultat afisat de Netscape – alte browsere afiseaza rezultate diferite (nu prea bune)

Mai multe rezultate

- Firefox 2.0.0.6



- IE 6.0.2900.2180



Câteva proprietăți şi valori pentru font

- font-family:
  - inherit (la fel ca fontul elementului parent)
  - Verdana, "Courier New", ... (daca fontul este instalat pe statia client)
  - serif | sans-serif | cursive | fantasy | monospace (Generic: browser-ul decide ce font sa foloseasca)
- font-size:
  - inherit | smaller | larger | xx-small | x-small | small | medium | large | x-large | xx-large | ***12***pt
- font-weight:
  - normal | bold |bolder | lighter | 100 | 200 | ... | 700
- font-style:
  - normal | italic | oblique

Forma simplificată a proprietăților

- Adesea mai multe proprietăți pot fi combinate:

  h2 { font-weight: bold; font-variant: small-caps; font-size: 12pt; line-height: 14pt; font-family: sans-serif }

  Poate fi scrisă sub forma:

  h2 { font: bold small-caps 12pt/14pt sans-serif }

Culori si lungimi

- color: si background-color:
  - aqua | black | blue | fuchsia | gray | green | lime | maroon | navy | olive | purple | red | silver | teal | white | #FF0000 | #F00 | rgb(255, 0, 0) | ***Additional browser-specific names (not recommended)***
- Acestea sunt elemente folosite pentru unitatile de masura:
  - em, ex, px, %
    - Dimensiunea fontului, inaltimea pe x, pixeli, procent din dimensiunea mostenita
  - in, cm, mm, pt, pc
    - inci, centimetri, milimetri, puncte (1/72 dintr-un inch), picas (1 pica = 12 puncte), relative la valoarea mostenita

Cateva proprietati si valori pentru text

- text-align:
  - left | right | center | justify
- text-decoration:
  - none | underline | overline | line-through
- text-transform:
  - none | capitalize | uppercase | lowercase
- text-indent
  - ***length*** | ***10%*** (indentarea primei linii a textului)
- white-space:
  - normal | pre | nowrap

Pseudo-clase

- Pseudo-clasele sunt elemente ale caror stare (si mod de aparitie) poate varia cu timpul
- Sintaxa: ***element*:*pseudo-class*** {...}
  - :link
    - Un link ce nu a fost vizitat
  - :visited
    - Un link ce a fost vizitat
  - :active
    - Un link pe care tocmai se executa un click
  - :hover
    - Un link peste care este pozitionat cursorul mouse-ului
- Pseudo-clasele sunt permise oriune in cadrul selectorilor CSS

Alegerea numelor

- CSS este proiectat pentru a *separa continut de stil*
  - Prin urmare, numele ce sunt folosite in HTML sau (mai ales) in XML trebuie sa descrie *continut, nu stil*
- Exemplu:
  - Sa presupunem ca definiti span.huge {font-size: 36pt} si folositi in cadrul mai multor documente
  - Ulterior descoperiti ca utilizatorii dezagreaza stilul acesta, deci modificati CSS-ul in span.huge {font-color: red}
  - Numele este incorect ales; incercati sa il redefiniti in toate documentele?
  - Daca ati fi ales de la inceput span.important {font-size: 36pt}, propriile documente ar fi fost mai clase si mai usor de intretinut

CSS

Aplicații la XML

O abordare diferită

- CSS este la fel pentru XML ca şi pentru HTML, *dar--*
  - HTML deja include elemente necesare pentru layout (aranjarea elementelor în pagină)
  - XML *nu* conține informații de layout, deci de sine stătător va fi afişat ca un singur mare chunk de text

- Când scriem CSS pentru XML, tipic primul lucru de care ne îngrijorăm este aranjarea textului în cadrul paginii

- Nici un browser curent nu suportă încă corect CSS, mai ales atunci când este folosit împreună cu XML, deci:
  - Ar trebui să vă asigurați că oricine vizualizează pagina creată foloseşte acelaşi browser, *sau*
  - Ar trebui să testați întotdeauna paginile în toate(majoritatea) browserelor importante

Proprietatea *display*

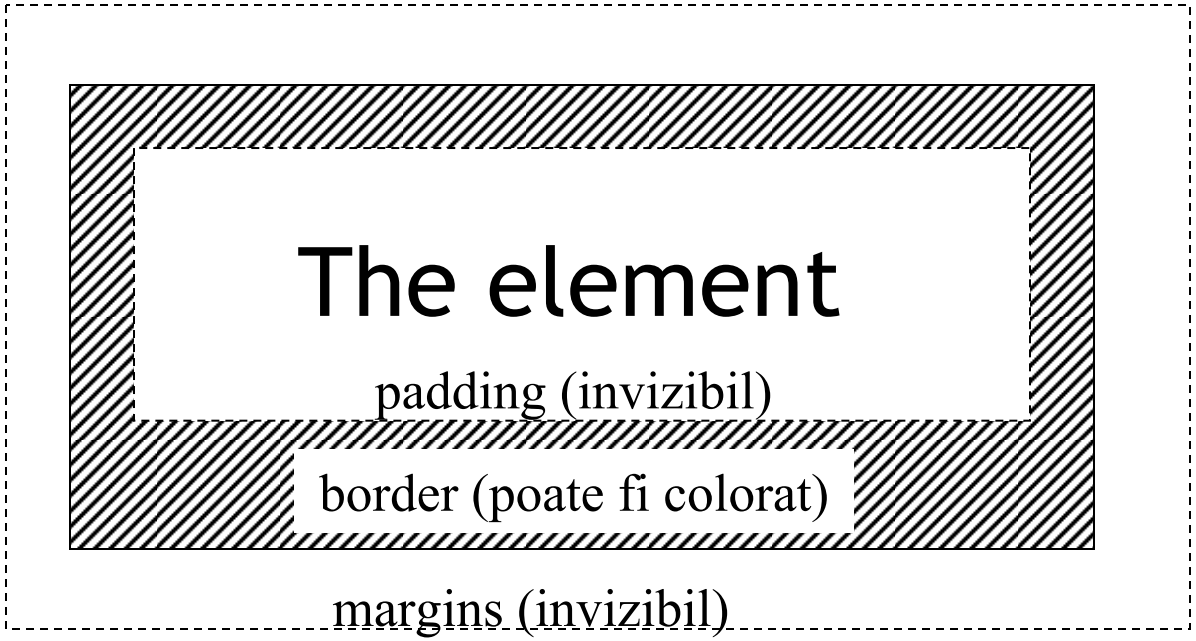
- Orice element XML formatat printr-o comandă CSS este considerată a fi într-un “box” invizibil
- Dreptunghiul ce conține un element XML poate avea unul dintre cele trei proprietăți display:
  - display: block
    - Elementul va porni pe o linie nouă şi de asemenea şi elementul imediat următorul (un *paragraf* HTML de exemplu)
  - display: inline
    - (implicit) Elementul nu va porni pe o linie nouă sau cauza ca următorul element să pornească pe o linie nouă (*bold* de exemplu)
  - display: none
    - Elementul este invizibil/ascuns şi nu va fi afişat

Unități CSS

- Pentru multe dintre valorile CSS rămase vom avea nevoie să specificăm măsurători de mărime
  - Acestea sunt folosite pentru specificarea dimensiunilor:
    - em lumgimea literei ‘m’
    - ex înălțimea literei ‘x’
    - px pixele (uzual 72 per inch, dar depinde de monitor)
    - % procent din dimensiunea moştenită
  - Acestea sunt de asemenea folosite pentri specificarea dimensiunilor, dar nu au sens decât dacă cunoaşteți rezoluția ecranului:
    - in inci
    - cm centimetri
    - mm milimetri
    - pt puncte (72pt = 1in)
    - pc picaşi (1pc = 12pt)
- Notă: puteți folosi fracții zecimale, precum 1.5cm

Boxes

- Dreptunghiul invizibil conține un element XML de stil ce are trei zone speciale:



Padding

- Padding-ul reprezintă spațiul extra din jurul unui element, dar din interiorul border-ului
- Pentru setarea padding-ului, folosiți oricare sau toate dintre:
  - padding-top: ***size***
  - padding-bottom: ***size***
  - padding-left: ***size***
  - padding-right: ***size***
  - padding: ***size***
- ***size*** este dat în unitățile descrise anterior
- Exemplu: sidebar {padding: 1em; padding-bottom: 5mm}

Bordere

- Puteți seta atributele border-ului cu oricare sau toate dintre : border-top:, border-bottom:, border-left:, border-right:, şi border:
- Atributele sunt:
  - *Grosimea* marginii: thin, medium (default), thick, sau o dimensiune specifică (e.g. 3px)
  - *Stilul* marginii: none, dotted, dashed, solid, double, groove, ridge, inset, sau outset
  - *Culoarea* marginii: una dinte cele 16 nume de culori predefinite sau o valoare hex cu ***#rrggbb*** sau rgb(***r***, ***g***, ***b***) sau rgb(***r%***, ***g%***, ***b%***)
- Exemplu: section {border-top: thin solid blue;}
- Notă: stilurile speciale (precum groove) nu sunt atât de *cool* pe cât sunare not as cool as they sound

Margini

- Marginile sunt spațiul suplimentar rămas în afara border-ului
- Setarea marginilor este analoagă setării padding-ului
- Pentru setarea marginilor, folosiți oricare sau toate dintre :
  - margin-top: ***size***
  - margin-bottom: ***size***
  - margin-left: ***size***
  - margin-right: ***size***
  - margin: ***size***

Interacțiuni cu box şi display

- With display:none, contents are invisible and margin, border, and padding settings have no effect
- With display:block, margin, border, and padding settings work about as you would expect
- With display:inline (which is the default if you don't specify otherwise):
  - Margin, border, and padding settings for left and right work about as you would expect
  - Margin, border, and padding settings for top and bottom *do not add extra space* above and below the line containing the element
    - This means that inline boxes will overlap other text

Elemente de redimensionare

- Orice element are o *dimensiune* şi o *poziție naturală* în care este afişat
- Se pot seta înălțimea şi lungimea elementelor display:block cu:
  - height: ***size***
  - width: ***size***
- *Nu* se pot seta înălțimea şi lungimea elementelor inline (dar se pot seta marginile stânga dreapta)
- În HTML se pot seta înălțimea şi lungimea imaginilor şi a tabelelor (tag-uri img şi table)

position:absolute; left: 0in; top: 0in

position:absolute; right: 0in; top: 0in

Setarea poziției absolute

- Pentru mutarea unui element într-o poziție absolută în cadrul paginii
  - position: absolute (necesar!) *şi încă* unul sau mai multe dintre
  - left: ***size*** sau right: ***size***
  - top: ***size*** sau bottom: ***size***
- Noțiuni:
  - ***size*** poate fi pozitiv sau negativ
  - top: ***size*** pune un element la ***size*** unități de marginea de sus a paginii
  - bottom: ***size*** pune un element la ***size*** unități de marginea de jos a paginii
  - left: ***size*** pune un element la ***size*** unități de marginea stânga a paginii
  - right: ***size*** pune un element la ***size*** unități de marginea dreapta a paginii

position:absolute; left: 0in; bottom: 0in

position:absolute; right: 0in; bottom: 0in

Setarea poziției relative

- Pentru mutarea relativă a unui element se folosesc
  - position: relative (necesar!) *şi încă* una sau mai multe dintre
  - left: ***size*** sau right: ***size***
  - top: ***size*** sau bottom: ***size***

- Noțiuni:
  - ***size*** poate fi pozitiv sau negativ
  - pentru mutare *stânga*, se poate face fie left negativ fie right pozitiv
  - pentru mutare *dreapta*, se poate face fie right negativ fie left pozitiv
  - pentru mutare *sus*, se poate face fie top negativ fie bottom pozitiv
  - pentru mutare *jos*, se poate face fie bottom negativ fie top pozitiv

Pseudo-elemente

- Pseudo-elementele descriu “elemente” ce nu stau între tag-uri în domentul XML
- Sintaxă: ***element:pseudo-class*** {…}
  - first-letter primul caracter dintr-un element la nivel de bloc
  - first-line prima linie dintr-un element la nivel de bloc (depinde de dimensiunea curentă a ferestrei browser)

```
p:first-line {color:#0000ff;font-variant:small-caps}
<p>Some text that ends up on two or more lines</p>
```

The output could be something like this:

```
Some text that ends
up on two or more lines
```

- În special folositoare pentru XML:
  - before adaugă material înainte de un element
    - Exemplu: author:before {content: "by "}
  - after adaugă material după un element

Principii de design

Principii de design

- În **The Non-Designer's Design Book:** Design and Typographic Principles for the Visual Novice, Robin Williams discută aceste patru principii:
  - Proximitate: Elementele înrudite ar trebui să fie grupate împreună
  - Aliniere: Nimic nu ar trebui plasat în pagină arbitrar -- orice element ar trebui să aibă o conexiune vizuală cu altceva din pagină
  - Repetiție: Unele aspectele ale design-ului ar trebui să fie repetate în pagină
  - Contrast: Dacă două itemuri nu sunt exact la fel ele ar trebui să fie reprezentate diferit – *chiar* diferit.

Proximitate

- Proximitate – apropriere – reprezintă instrumentul util pentru organizarea lucrurilor în cadrul paginii
  - Dacă lucrurile sunt apropiate ele apar ca fiind înrudite
  - Prin urmare:
    - Dacă lucrurile sunt înrudite, ele ar trebui să stea apropiat
    - Dacă lucrurile nu sunt înrudite, ele nu ar trebui să stea apropiat
  - Evitarea spațierii egale a tuturor lucrurilor
  - Nu se plasează lucruri în colțuri sau singure în mijlocul paginii
  - Evitarea introducerii prea multor grupuri în cadrul paginii
  - Headerele trebuie să arate ca headere iar lucrurile ce nu sunt headere ar trebui să nu arate ca headere

Alinierea

- Alinierea – “lining thing up”
  - Alinierea bună ajută la unificarea şi organizarea paginii
  - Se doreşte evitarea impresiei de “scattered all over”

Alinierea stânga tinde
să apară în mod natural
în paginile Web

Alinierea dreaptă nu
este în general de prea
mare ajutor

Alinierea pe centru tinde să fie
plictisitoare şi este mai ales urâtă
când linii sunt toate cam de aceeaşi
lungime

  - Încercați să evitați mai mult de un singur tip de aliniament în cadrul unei pagini

Repetiția

- Scopul repetiției este:
  - De a unifica paginia sau grupul de pagini
  - De a adăuga interes vizual
    - Puține lucruri arată mai plictisitor decât pagini lungi, continue de text
    - Lucrurile ce arată plictisitor cel mai adesea nu primesc o a doua privire

- Repetiția e legată de consistență
  - Deja ați folosit-o probabil pentru consistența fonturilor, a headerelor, etc.
  - Unele elemente vizuale (elemente de fundal, icon-uri, bordere, reguli orizontale) ar trebui să se repete în cadrul unei pagini Web sau al unui grup de pagini Web înrudite
  - Dacă paginile sunt grupate, ele ar trebui să *apară* ca fiind grupate
  - Totuşi nu folosiți prea multă repetiție pentru că poate deveni de la un punct supărătoare

Constrastul

- Contrastul apare atunci când două elemente sunt clar diferite
- Se poate crea contrast prin folosirea dimensiunilor diferite
- Se poate crea constrast prin folosirea de diverse fonturi
- Se pot folosi linii subțiri şi groase
- Se pot folosi linii orizontale şi verticale
- Se pot folosi culori contrastante: culori reci şi calde
- Se poate folosi text spațiat larg sau strâns spațiat
- Nu e aşa mare contrastul între tipul de 12 puncte şi cel de 14 puncte

Repetiție!

- Constratul apare atunci când două elemente sunt *clar* diferite
- Se poate crea contrast prin:
  - Folosirea de diverse dimensiuni de tip
  - Folosirea de diverse fonturi
  - Folosirea liniilor subțiri şi groase
  - Folosirea liniilor orizontale şi verticale
  - Folosirea culorilor contrastante: culori reci şi calse
  - Folosirea de text spațiat diferit

- Nu fiți plângăreți – faceți lucrurile să arate ***cu adevărat*** diferit
  - Nu e prea mare contrastul între font de 12 şi font de 14!

Tipuri de fonturi

- Serif Fonts
- Sans serif fonts -- no serifs
- Monospaced fonts -- all characters are the same width
- *Display fonts - not intended for lots of text*
- EVEN IN A GOOD FONT, LARGE AMOUNTS OF TEXT IN ALL CAPITALS IS DIFFICULT TO READ

Alte câteva principii

- Stabilirea unei ierarhii vizuale
  - Oamenii prima dată văd grafica şi pe urmă textul
  - Balansarea, organizarea şi contrastul vizual sunt vitale
- Direcționara ochilor cititorului
  - Oamenii scanează textul de la stânga la dreapta, de sus în jos
  - Doar primii patru inchi cei mai de sus ar putea fi vizibili
  - Folosiți nuanțe pastel pentru fundal sau elemente minore
- Atenție la elemente ce distrag atenția
  - Ilustrațiile bogate şi (în special) grafica animată sau textul animat distrag atenția utilizatorului de la conținut
  - Dacă totul este accentuat, nimic nu este accentuat
- Asigurarea consistenței
  - Nu se lasă lucrurile împrăştiate în toată pagina
  - Lăsați stilul propriu să “evolueze” pe măsură ce îmbunătățiți pagina

Stabilirea unui look consistent

- Orice pagină din cadrul site-ului ar trebui să împartă unele elemente de stil cu toate celelalte pagini
  - Utilizatorul ar trebui să ştie, fără să se gândească la asta, că navighează încă în cadrul aceluiaşi site
  - Se foloseşte acelaşi logo sau acelaşi set de butoane de navigare în fiecare pagină
  - Se foloseşte o schemă de culori şi un set de fonturi consistente
- Paginile nu trebuie să arate toate identic, dar ele trebuie să aibă acelaşi stil
- Style sheet-urile CSS pot fi de mare ajutor în definirea stilului consistent
  - Dar trebuie teste făcute pe o varietate de browsers

Corectitudinea stilului HTML

Ghiduri de stil

- Există multe ghiduri de stil HTML pe Web
- Unul dintre cele mai bune este cel de la Yale, http://info.med.yale.edu/caim/manual/
- Un altul este cel de la W3C, http://www.w3.org/Provider/Style/
- Unul dintre cele mai plăcute site-uri este http://www.webpagesthatsuck.com/
  - Motto: “Where you learn good Web design by looking at bad Web design”
- O carte adecvată subiectului:
  - **Don't Make Me Think:** A Common Sense Approach to Web Usability, de Steve Krug, Roger Black

Cunoaşteți pe cine încercați să impresionați

- Care este audiența țintă?
  - Publicul general (Surferi Web)
    - Precum o copertă de revistă, pagina home ar trebui să atragă oamenii
      - Folosiți grafică de calitate şi fraze scrise cu bold
      - *In cele mai puține cuvinte posibil*, spuneți vizitatorilor ce aveți de oferit
      - Toate link-urile ar trebui să conducă direct la paginile site-ului
  - Vizitori ocazionali
    - Navigarea ar trebui să fie simplă şi intuitivă
    - Se folosesc pagini de sumar, icoane ajutătoare, FAQs şi organizare cât mai simplă
  - Experți şi vizitatori frecvenți
    - Furnizarea de informație bine organizată rapid cu un deranj minim
    - Evitarea graficii pretențioase, pagini greu de încărcat
    - Furnizarea de “site maps” şi funcții de căutare
  - Utilizatori internaționali
    - Folosirea limbajului standard, uşor de înțeles
    - Furnizarea de pagini în diverse limbi (suport internațional)
    - Evitarea formatelor de timp şi dată specific regionale, pecum 10-12-2002

Cunoaşteţi ce încercaţi să faceţi

- O pagină fără un scop este precum o prezentare orală fără o topică
- Încercaţi să vindeţi ceva?
  - Odorizante de cameră: folosiţi scene exterioare (natură) cât mai frumoase
  - Deodorant: oameni frumoşi (atât femei *cât şi* bărbaţi)
  - Computere: fotografii atractive şi specificaţii tehnice
  - Pe voi înşivă: vedeţi orice carte legată de scrierea unui CV
- Încercaţi să transmiteţi informaţie?
  - Folosiţi o organizare cât mai clară, chiar pe baza unui cuprins (table of contents)
  - Pentru multe topici este necesară şi o secţiune FAQ

Formate media

- Cărțile au existat chiar înainte de Biblia lui Gutenberg din 1456
- Câteva dintre “standardele de interfață” pentru cărți:
  - Cărțile au coperți
  - Cărțile sunt legate de-a lungul marginii stânga
  - Titlul apare pe cotor sau pe copertă
  - Cărțile au o pagină de titlu
  - Paginile unei cărți sunt numerate
    - Paginile impare sunt în dreapta
    - Primele pagini sunt numerotate cu simboluri romane
  - Cărțile au un cuprins şi un index

Paginile Web nu sunt cărți

- Standardele evoluează rapid, dar nu sunt “finalizate”
- Web-ul aduce noi capabilități:
  - Publicarea este *ieftină*, orice o poate face
  - Hiperlinkurile permit acces neliniar la informație
  - Motoarele de căutare permit găsirea mai uşoară a informației
    - Înainte oamenii lucrau cu sute de bookmarkuri; astăzi toată lumea foloseşte Google
- Web-ul aduce şi noi situații:
  - Utilizatorii se pot “pierde în hiperspațiu”
    - Sunt necesare instrumente adecvate de navigare
  - Utilizatorii navighează pe Web rapid
    - Aveți la dispoziție *zece secunde* pentru a vă face auzit mesajul

Câteva sugestii foarte generale

- Stilul bun de scris *a fost*, *este* şi *va fi întotdeauna* important
  - Tot ce ați învățat vreodată despre scrierea de compuneri bune se aplică
- Folosiți un spell checker
  - Astăzi aproape orice aplicație software include un spell checker
- Faceți ca fiecare pagină să iasă în evidență individual
  - Nu ştiți cum ajunge cineva să acceseze respectiva pagină

Întrebări

- Cititorul ar trebui să poată descoperi:
  - Cine a scris pagina?
    - Găsiți o pagină despre cancerul la plămâni. Ea a fost scrisă de (a) American Medical Association, (b) cineva care lucrează pentru Philip Morris, sau (c) un instalator din Feteşti?
  - Care este topicul abordat în pagină?
    - Dacă nu aveți nimic de spus, mai bine nu-l spuneți
    - Folosiți un tilu clar, scurt – poate deveni un bookmark
  - Când a fost pagina scrisă/actualizată?
    - Descoperiți o pagină despre un nou medicament disponibil “luna viitoare”
    - O altă pagină descrie “ultima versiune” a unui anumit software
  - Unde este pagina?
    - Cine a scris pagina? Cine a sponsorizat scrierea paginii?
    - Dacă tipăresc pagina o voi mai putea regăsi pe Web din nou vreodată în viitor?

Construirea de ajutor de navigație

- Când cineva vă accesează site-ul:
  - Ce e cel mai probabil să caute?
  - Cât de sofisticați sunt utilizatorii?
  - Aproape nimeni nu întreprinde suficientă testare a uşurinței de navigare
- O problemă clasică: descoperiți o pagină interesantă dar nu ştiți ce pagini se află “împrejurul” ei
  - Sunt paginile organizate într-o manieră simplă şi consistentă?
  - Poate vizitatorul să găsească calea către pagina home?
  - Poate utilizatorul să spună dacă se mai află sau nu pe acelaşi site?
  - Barele de butoane sunt utile, dar nu omiteți legăturile text
  - Evitați paginile dead-end (acelea fără nici un link)

Ajutați vizitatorii să găsească pagini în site

- Dacă aveți multe pagini puteți folosi meniuri pe nivele...
  - Priviți un cuprins dintr-o carte; de obicei există secțiuni principale şi subsecțiuni
- ...dar utilizatorilor nu le plac multe meniuri mici
  - Studiile arată că utilizatorii preferă meniuri dense cu multe alternative în detrimentul meniurilor mici, gen one-step-at-a-time
  - “Site maps” au devenit tot mai populare
- Nu toată lumea încarcă grafica implicit
  - Hărțile grafice sunt frumoase, dar păstrați totuşi şi legături text
  - Gândiți-vă să faceți paginile disponibile şi persoanelor cu disabilități
- Luați în considerare adăugarea unui motor de căutare în cadrul site-ului
- Nu permiteți părăsirea accidentală a site-ului
  - Faceți distincție între link-urile locale şi link-urile ce dus vizitatorii în afara site-ului
  - Dați paginilor un “look” consistent

Puneți lucrurile importante la suprafață

- Paginile Web sunt de obicei mai mari decât fereastra în care acestea sunt vizualizate
  - Primul lucru pe care vizitatorii îl văd este partea de sus a paginilor Web
  - Mulți vizitatori nu vor face niciodată scroll down
- Partea de sus a unei pagini ar trebui să spună vizitatorilor tot ce trebuie aceştia să cunoască despre pagina respectivă
  - Dacă un vizitator este pierdut în site s-ar putea să nu remarce legăturile puse în josul paginii
  - Adesea, legăturile puse în susul *şi* josul paginii sunt o idee bună
    - Mai ales în cazul unui şir liniar de pagini, în care link-urile posibile sunt Previous, Next şi (poate) Home sau Table of Contents

Organizați-vă paginile

- *Orice* organizare este mai bună decât *nici o* organizare
- O ierarhie (arbore) de obicei funcționează cel mai bine
  - Puneți conceptele cele mai importante şi mai generale în apropierea părții de sus
  - Paginile de mai jos sunt în general mai specifice
- Arborii nu ar trebui să fie prea adânci
  - Utilizatorii nu apreciază să traverseze multe pagini pentru găsirii acelei pagini de interes
- Arborii nu ar trebui să fie nici întinşi pe orizontală
  - Utilizatorilor nu le place să traverseze o listă lungă de legături pentru găsirea aceleia de care au nevoie
- Desenați o poză a organizării site-ului!

Alte organizări

- Grid:

	HTML	XML	XSLT
Lecture	http://...	http://...	http://...
Links	http://...	http://...	http://...
Assignment	http://...	http://...	http://...

- Liniar:
  - Paginile puse în ordinea în care trebuie citite, cu legături **Previous** şi Next
  - Organizarea cea mai des întâlnită în tutoriale

Grafica pe pagina principală

- Pagina principală (“home”) reprezintă punctul de pornire pentru vizitatorii site-ului
  - Grafica frumoasă face pagina să arate mai bine
  - Grafica complexă face pagina să se încarce mai greu
- Care este audiența țintă?
  - Clienți potențiali
    - Modul de apariție este important...
    - ...dar majoritatea utilizatorilor nu vor aştepta mai mult de 7 sau 8 secunde pentru ca pagina să se încarce
  - Clienți existenți, studenți, angajați
    - Obținerea informației rapid este de mare importanță

Site-uri în continuă adaptare

- Multe site-uri trebuie actualizate frecvent
  - Folosiți o imagine sau text New! – poate ajuta la indicarea modificărilor survenite
    - Dar cât de mult rămâne un element “nou”?
    - Datele ataşate elementelor sunt mai informative
  - Dacă informația este împrăştiată de-a lungul mai multor pagini s-ar putea să fie mai bine folosirea unei pagini centrale What’s New
  - Ex.: se pot pune anunțuri datate deasupra şi adăuga materiale în josul paginii

FAQ

- Pentru multe site-uri o pagină FAQ (Frequently Asked Questions) page, cu răspunsuri, se poate dovedi deosebit de ajutătoare
  - O pagină FAQ este mai ales ajutătoare începătorilor din domeniu
  - http://www.faqs.org/
- Cea mai mare problemă cu paginile FAQ este că multe dintre ele sunt “false”
  - O companie pune o pagină FAQ despre propriile produse ce, în mod evident, nu răspunde la întrebări venind din partea unor utilizatori reali
    - “Care este cel mai mare beneficiu al utilizării şamponului XYZ?”
  - Nu vă mințiți clienții!

Standarde de proiectare

- O companie ar trebui să păstreze standarde de design pentru paginile Web ale companiei
  - Ele dovedesc identitatea companiei
  - Probleme:
    - Grupuri şi indivizi şi-au stabilit propriile standarde şi nu sunt dispuşi la schimbare
    - Oamenii nepotriviți pot fi puşi să definească standardele de design
      - Ar putea să cunoască puțin sau nimic despre standardele deja existente
      - Pot decide folosirea de “prea multe” standarde – lucruri ce arată bine individual, dar care nu merg bine împreună
      - Ar putea dura la nesfârşit finalizarea încât standardele să devină “any day now”
      - Ei au propriile idealuri şi ignoră sau uită *utilizatorul*

Tipuri de formate grafice

- Există *trei* formate grafice ce pot fi folosite pe Web:
  - GIF (Graphics Interchange Format)
  - JPG sau JPEG (Joint Photographic Experts Group)
  - PNG (Portable Network Graphics)

Formatul GIF

- Cel mai comun format
  - Puteți specifica câte culori se folosesc (2, 4, 8, 16, etc.)
    - Cu cât mai puține culori cu atât mai mic va fi fişierul rezultat
  - Fişiere fără pierderi de calitate – puteți recrea exact imaginea originală
  - GIF-urile pot fi animate
  - GIF-urile pot fi interlaced
    - Proprietate ce permite afişarea rapidă a imaginilor
  - GIF-urile suportă transparență
    - Folositoare pentru conturarea de margini arbitrare

Formatul JPG

- Firşierele JPEG furnizează o schemă de compresie superioară atunci când există gradienți de culoare în cadrul imaginii
  - Adecvat deci pentru fotografii
  - JPEG-uri au pierderi – unele informații sunt pierdute în cadrul compresiei iar informația este interpolată (trucată) atunci când se recrează imaginea
  - Se poate seta rata de compresie – cu cât mai multă compresie cu atât fişierul rezultat va fi mai mic, dar şi mai multă informație va fi pierdută

PNG Graphics

- PNG has three main advantages:
  - Alpha channels: you can have *variable* (partial) transparency
  - Gamma correction, so you can get the same colors on different platforms
  - Two-dimensional interlacing
- PNG also provides:
  - Better compression than GIF
  - A less convenient way to do animations
  - No legal hassles

Formatul PNG

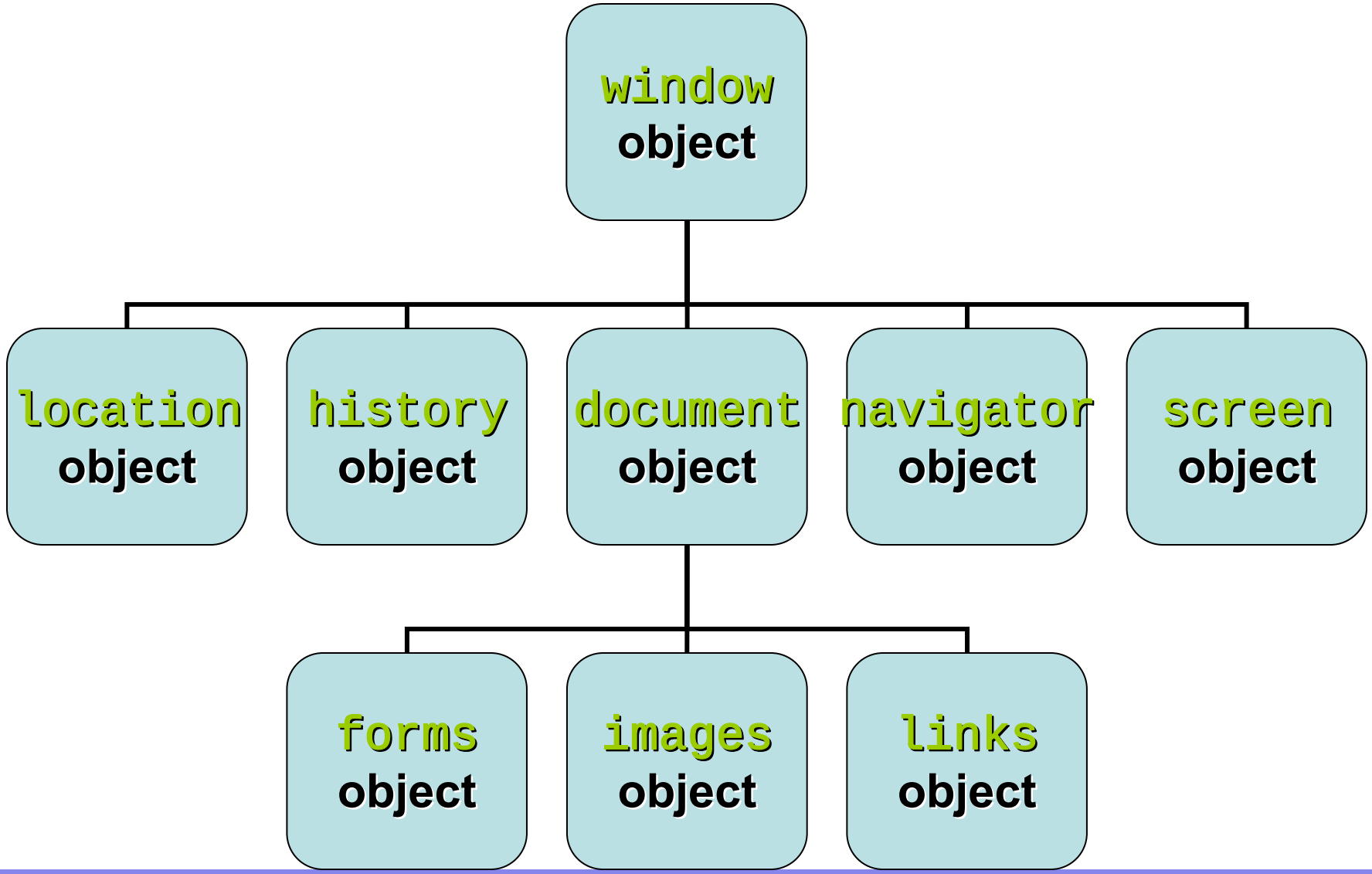
- Fişierele PNG au câteva avantaje:
  - *Alpha channels*: puteți avea transparență *variabilă* (parțială)
  - *Gamma correction*, puteți obține aceleaşi culoari pe diverse platforme
  - *Two-dimensional interlacing*
- PNG mai furnizează:
  - O mai bună compresie decât GIF
  - O manieră mai puțin convenientă de a face animații

Introducere în DHTML: Lucrul cu Obiecte Browser

Alphabet Soup: DHTML, BOM, DOM

- DHTML se referă la ideea de generare de conținut Web în mod dinamic. Se bazează pe datele furnizate de către utilizator.
- BOM (Browser Object Model) descrie modul în care programăm diversele obiecte disponibile spre folosire într-un browser.
- DOM (Document Object Model) se referă la un standard W3C de programare a unei pagini web (documentul). Ignoră obiectele browser non-standarde.

Obiecte Browser Standarde



Obiectul window

- Obiectul window reprezintă cadrul sau fereastra, sau browserul, ce conține o pagină web.
- Exemple de folosire:

```
window.alert()
window.prompt()
window.confirm()
```

Atributul `defaultStatus`

- Putem actualiza atributul **`window.defaultStatus`** pentru a modifica textul status-bar-ului ferestrei browser-ului
  Forma generală:
  `window.defaultStatus = `*`new string`*`;`

Exemplu 1


```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
      strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Status Bar Update</title>

    <script language = "JavaScript 2.1"
          type="text/JavaScript">
      <!--
      function main()
      {
        var strLinkText = new String();

        strLinkText = "Welcome to the wonderful world of DHTML!";

        window.defaultStatus = strLinkText;
      }
      // -->
    </script>
  </head>
  <body onLoad="JavaScript:main();">
    <h1 style="text-align:center">Status Bar Update</h1>
    <hr size="2"
       width="85%" />

    <p>
    Check the Status Bar below.
    </p>
  </body>
</html>
```

Metoda `setInterval()`

- Permite repetarea automată a unui apel de funcție la un număr specificat de milisecunde. Întoarce un obiect “interval”, de care metoda window.clearInterval() are nevoie pentru oprirea repetiției:

```
object = setInterval(“function()”, millisecond delay)
```

Obiectul 2

- Obiectul Date permite lucruri precum obținerea datei ultimei modificări, calendare, ceasuri, etc.
- Metode utile Date includ

```
getDate()
getDay()
getFullYear()
getMonth()
getHours()
getMinutes()
getSeconds()
```

Exemplul 2


```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
      strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Status Bar Clock</title>

    <script language = "JavaScript 2.1"
          type="text/JavaScript">
      <!--

      var tInterval = new Object();

      function showClock()        {

        var d = new Date();
        var dHour = d.getHours();
        var dMinutes = d.getMinutes();
        var dSeconds = d.getSeconds();
        var strAmPmFlag = new String("");
        var strNewTime = new String();

        if(dHour < 12)           {
          strAmPmFlag = " AM";
        }else{
          strAmPmFlag = " PM";
        }

      if(dHour > 12)           {
          dHour -= 12;
        }else if(dHour == 0){
          dHour = 12;
        }//end dHour if
```


```
    if(dMinutes < 10)           {
          dMinutes = "0" + dMinutes;
       }//end include leading zero if for minutes

       if(dSeconds < 10)           {
          dSeconds = "0" + dSeconds;
       }//end include leading zero if for seconds

       strNewTime  = dHour;
       strNewTime += ":";
       strNewTime += dMinutes;
       strNewTime += ":";
       strNewTime += dSeconds;
       strNewTime += strAmPmFlag;

       window.defaultStatus = strNewTime;

    }//end showClock

    function startClock()        {
       tInterval = window.setInterval("showClock()",1000);
    }//end startClock

    function stopClock()        {
       window.clearInterval(tInterval);
       window.defaultStatus = "";
    }//end startClock


    // -->
  </script>
</head>
```


```
<body>
    <h1 style="text-align:center">Status Bar Clock</h1>
    <hr size="2"
       width="85%" />

    <p>
     <a href="#"
       onClick="javascript:startClock();">Start Status Bar Clock</a>
     <br />
     <a href="#"
       onClick="javascript:stopClock();">Stop Status Bar Clock</a>

    </p>
  </body>
</html>
```

Obiectul history

- Obiectul **`history`** permite scrierea de cod care să navigheze istoria unui browser web.
- Include metoda **`history.go()`** ce primeşte o valoare întreagă pentru a indica dacă este necesar mersul înapoi (valoare negativă) sau înainte (valoare pozitivă).
- Metodele **`history.back()`** şi **`history.forward()`** permit navigarea incrementală.

Exemplul 3


```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Programming the History Object</title>

    <script language = "JavaScript 2.1"
          type="text/JavaScript">
      <!--

      function NavHistory(bolGoBack,intHowMany)        {

        if(intHowMany != null)          {
          window.history.go(intHowMany);
        }else if(bolGoBack){
          window.history.back();
        }else{
          window.history.forward();
        }//end intHowMany if

      }//end NavHistory()


      // -->
    </script>
  </head>
```


```
<body>
    <h1 style="text-align:center">Programming the History Object</h1>
    <hr size="2"
       width="85%" />

    <p>
     <a href="#"
        onClick="javascript:NavHistory(true);">Go Back One Page in History</a>
     <br />
     <a href="#"
        onClick="javascript:NavHistory(false);">Go Forward One Page in History</a>
     <br />
     <a href="#"
        onClick="javascript:NavHistory(false,-2);">Go Back Two Pages in History</a>
    </p>
  </body>
</html>
```

Obiectul location

- Obiectul **location** permite scrierea de cod care să modifice URL-ul paginii curent, realizând efectiv navigarea către o altă pagină web.
- Putem face asta în două feluri:
  - Actualizând atributul **window.location.href**
  - Prin invocarea metodei **window.location.replace()**
- A doua abordare duce şi la actualizarea paginii anterioare în istoria browserului.

Exemplul 4


```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
      strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Programming the Location Object</title>

    <script language = "JavaScript 2.1"
          type="text/JavaScript">
      <!--

      function VisitRandom()        {
        var arrLinks = new Array(10);
        var intRandomIndex = new Number();

        intRandomIndex = (Math.floor(Math.random()*10));

        arrLinks[0] = "http://www.yahoo.com/";
        arrLinks[1] = "http://www.webmonkey.com/";
        arrLinks[2] = "http://www.wikipedia.org/";
        arrLinks[3] = "http://www.npr.org/";
        arrLinks[4] = "http://www.nytimes.com/";
        arrLinks[5] = "http://www.google.com/";
        arrLinks[6] = "http://www.wdvl.com/";
        arrLinks[7] = "http://www.iupui.edu/";
        arrLinks[8] = "http://espn.go.com/";
        arrLinks[9] = "http://sourceforge.net/";

        window.location.href = arrLinks[intRandomIndex];

      }//end VisitRandom
```


```
// -->
    </script>
  </head>

<body>
    <h1 style="text-align:center">Programming the Location Object</h1>
    <hr size="2"
       width="85%" />

    <p>
       <form id = "navForm">
          <input type = "button"
                id = "btnChangeLocation"
                value = "Go to a Random Page"
                onClick = "JavaScript:VisitRandom();">
       </form>
    </p>
  </body>
</html>
```

Obiectul navigator

- Obiectul **navigator** permite detectarea de informații importante privind aplicația rulată de utilizator pentru vizualizarea paginii web.
- Atributele obiectului **navigator** pot întoarce informații despre numele şi versiunea browserului (**navigator.appName**), sau numele şi versiunea OS (**navigator.userAgent**).
- Folositor pentru scrierea de scripturi de detecție a browser-ului.

Exemplul 5


```
<head>
    <title>Detecting Browser Information</title>

    <script language = "JavaScript 2.1"
          type="text/JavaScript">          <!--

      function GetBrowserInfo()         {
        var strUserBrowser = new String();
        var strUserOS = new String();
        var strOutputMsg = new String();

        strUserBrowser = window.navigator.appName;
        strUserOS = window.navigator.userAgent;

        strOutputMsg  = "**********************\n";
        strOutputMsg += "SYSTEM INFORMATION\n";
        strOutputMsg += "Browser: ";
        strOutputMsg += strUserBrowser;
        strOutputMsg += "\nOperating System: ";
        strOutputMsg += strUserOS;
        strOutputMsg += "\n**********************";

        window.alert(strOutputMsg);

      }//end GetBrowserInfo()

      // -->
    </script>
  </head>
  <body onLoad="javascript:GetBrowserInfo();">
    <h1 style="text-align:center">Detecting Browser Information</h1>
    <hr size="2"
       width="85%" />
  </body>
</html>
```

Obiectul `screen`

- Obiectul **`screen`** include un grup de atribute folositoare:
  `window.screen.width`
  `window.screen.height`
  `window.screen.colorDepth`

Exemplul 6


```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Detecting Screen Information</title>

    <script language = "JavaScript 2.1"
          type="text/JavaScript">
      <!--

      function GetScreenInfo()  {
        var strUserScreen = new String();
        var strOutputMsg = new String();
        var intScreenWidth = new Number();
        var intScreenHeight = new Number();

        intUserScreen = window.screen.colorDepth;
        intScreenWidth = window.screen.width;
        intScreenHeight = window.screen.height;

        strOutputMsg  = "Your current monitor settings support a ";
        strOutputMsg += "color depth of ";
        strOutputMsg += intUserScreen.toString();
        strOutputMsg += " bits.\n";
        strOutputMsg += "The size of your screen is ";
        strOutputMsg += intScreenWidth.toString();
        strOutputMsg += " pixels wide by ";
        strOutputMsg += intScreenHeight.toString();
        strOutputMsg += " pixels tall.";

        window.alert(strOutputMsg);

      }//end GetScreenInfo()
```

```
      // -->
    </script>
  </head>
  <body onLoad="javascript:GetScreenInfo();">
    <h1 style="text-align:center">Detecting Screen Information</h1>
    <hr size="2"
       width="85%" />


  </body>
</html>
```

Obiectul `event`

- Obiectul **`event`** permite construcția de handleri de evenimente customizați.
- Două atribute pe care le vom folosi sunt:
  `event.clientX`
  `event.clientY`
- Cele două atribute redau poziția pe X şi Y a cursorului mouse pentru evenimente precum **`onClick`**…

Exemplul 7


```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
      strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Detecting Mouse Position using the Event Object</title>

    <script language = "JavaScript 2.1"
          type="text/JavaScript">
      <!--

      function GetXYPosition(e)
      {
        var intOutputMsg = new String("");
        var intMouseX = new Number(0);
        var intMouseY = new Number(0);

        intMouseX = e.clientX;
        intMouseY = e.clientY;

        strOutputMsg  = "Your current mouse position is \n";
        strOutputMsg += "X Position: ";
        strOutputMsg += intMouseX.toString();
        strOutputMsg += "\n";
        strOutputMsg += "Y Position: ";
        strOutputMsg += intMouseY.toString();
        window.alert(strOutputMsg);

      }//end GetScreenInfo()


      // -->
    </script>
  </head>
```

```
<body onClick="javascript:GetXYPosition(event);">
    <h1 style="text-align:center">
Detecting Mouse Position using the Event Object</h1>
    <hr size="2"
       width="85%" />


  </body>
</html>
```

Obiectul `document`

- Obiectul `document` reprezintă pagina curentă afişată.
- Include un număr de atribute folositoare precum:

```
window.document.forms
window.document.links
window.document.images
```

Quiz

- Scrieti o aplicație ce afişează în status bar coordonatele cursorului mouse-ului pentru un eveniment onClick asupra unui element html.

Ce este DOM?

- Document Object Model (DOM) – model de programare pentru reprezentarea obiectelor conținute într-un document web.
- Există mai multe nivele diferite ale Standardului DOM - W3C.

- Level 0: Nu e chiar un nivel standard. Se referă la modelele pentru dezvoltare a vendorilor de browsere a gestiunii documentelor anterioare standardelor.
- Level 1: Prima recomandare de Standard DOM a W3C. Include două părți: core (acoperă XML & HTML) şi o secțiune doar pentru HTML.
- Level 2: Include adăugarea de evenimente şi style sheets. Suportat de versiunile curente ale majorității browserelor populare.
- Level 3: Include adăugări de elemente adresate documentelor XML.

Vederi Liniare vs. Ierarhice

- Nivelul 0 al DOM abordează o vedere oarecum liniară a obiectelor dintr-un document web:

```
strUserName =
window.document.frmMain.txtUser.
value
```

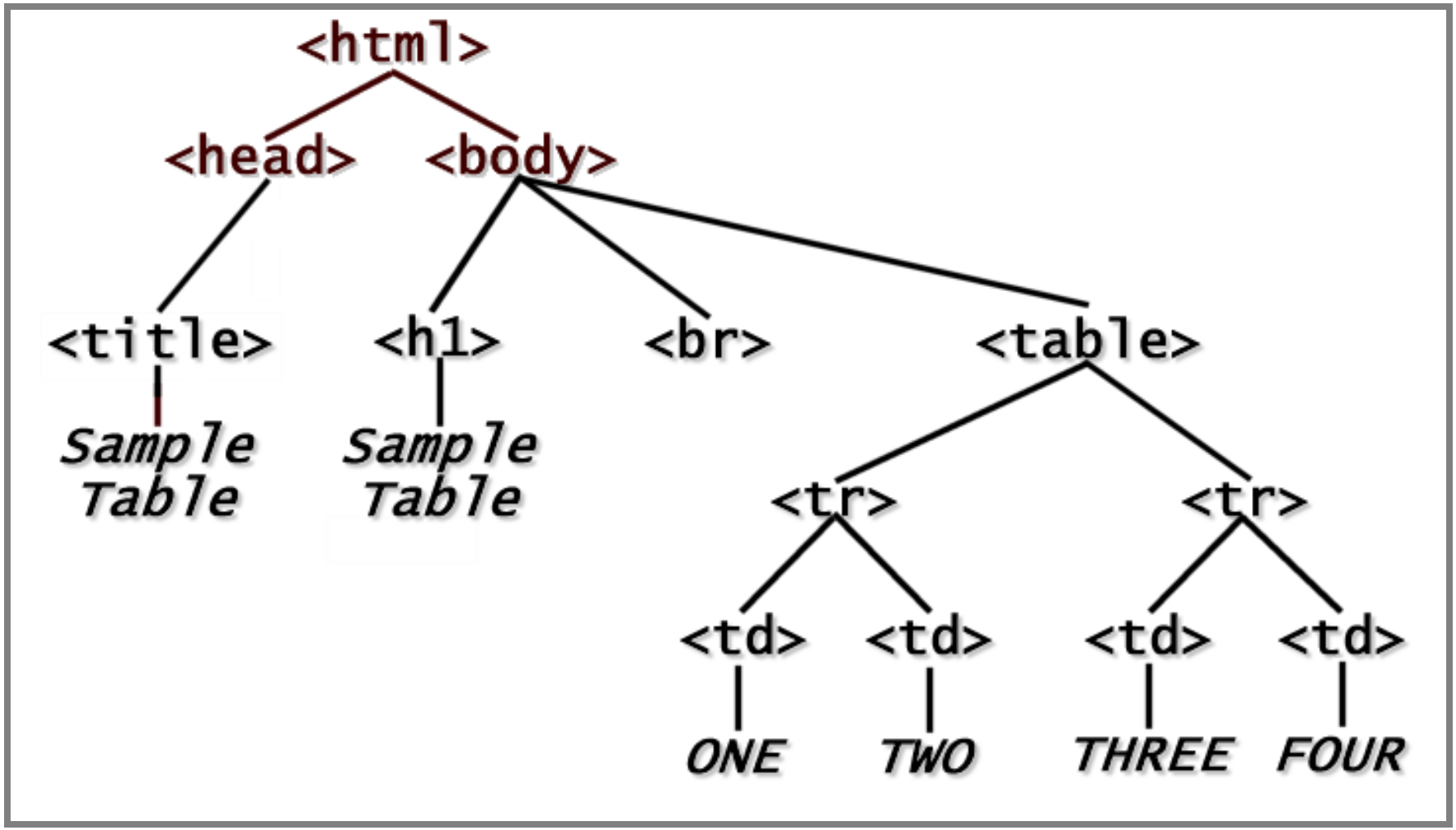
Vederi Liniare vs. Ierarhice

- Celelalte nivele folosesc o vedere ierarhică.
- În loc de identificarea unor nume de tag-uri HTML specifice, vederea funcționează similar modului de parcurgere a unor *noduri* într-un arbore.
- Fiecare nod are potențialul de a fi *parent*, *sibling* sau *child* pentru celelalte noduri din document.

Exemplul 1


```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
      strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Sample Table</title>
  </head>
  <body>
    <h1 style="text-align:center">Sample Table</h1>
    <hr size="2"
       width="85%" />
    <table border=1>
      <tr>
        <td>
          ONE
        </td>
        <td>
          TWO
        </td>
      </tr>
      <tr>
        <td>
          THREE
        </td>
        <td>
          FOUR
        </td>
      </tr>
    </table>
  </body>
</html>
```


<html>
<head>
<body>
<title>
<h1>

<table>
Sample Table
Sample Table
<tr>
<tr>
<td>
<td>
<td>
<td>
ONE
TWO
THREE
FOUR

Pregătirea paginilor pentru DOM

- Toate paginile *trebuie* să fie documente XHTML bine-formate.
- Toate paginile *trebuie* să includă un DOCTYPE valid.
- *Trebuie* să includeți tot textul în elementele XHTML valide.
- Elementele relevante sunt identificate folosind atributul `id`.

Atributul `id`

- `id` este atributul XHTML ce furnizează un identificator intern al unui element XHTML (tag).
- Pentru a putea manipula/citi elemente atributele `id` trebuie să fie unice.

Exemplul 2


```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
    strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Sample Table</title>
  </head>
  <body>
    <h1 style="text-align:center"
       id="H1PageTitle">Sample Table</h1>
    <hr size="2"
       width="85%"
       id="hr1" />
    <table border=1
         id="SampleTable">
     <tr id="SampleTableRow1">
      <td id="SampleTableTD1">
       ONE
      </td>
      <td id="SampleTableTD2">
       TWO
      </td>
     </tr>
     <tr id="SampleTableRow2">
      <td id="SampleTableTD3">
       THREE
      </td>
      <td id="SampleTableTD4">
       FOUR
      </td>
     </tr>
    </table>
  </body>
</html>
```

Referințe

- Unele exemple din această prezentare au fost preluate din tutorialul **W3Schools** de la http://www.w3schools.com/css/css_syntax.asp
- Un foarte bun tutorial online este şi cel al lui Dave Raggett, **Adding a Touch of Style,** disponibil la http://www.w3.org/MarkUp/Guide/Style
- **Index DOT Css** este de asemenea o sursă utilă de informare pentru CSS: http://www.blooberry.com/indexdot/css/index.html