Technologies de l'Internet

Partie 5 : DOM, JavaScript

Iulian Ober

iulian.ober@irit.fr

DHTML : Introduction

- Objectif : créer des pages dynamiques
  - objets qui apparaissent/disparaissent/changent (ex. menus)
  - validation de données de formulaires
  - gestion des "cookies"
  - ...
- DHTML n'est pas un langage / norme mais une combinaison de technologies: (X)HTML, CSS, DOM, JavaScript,...

Modèle DOM

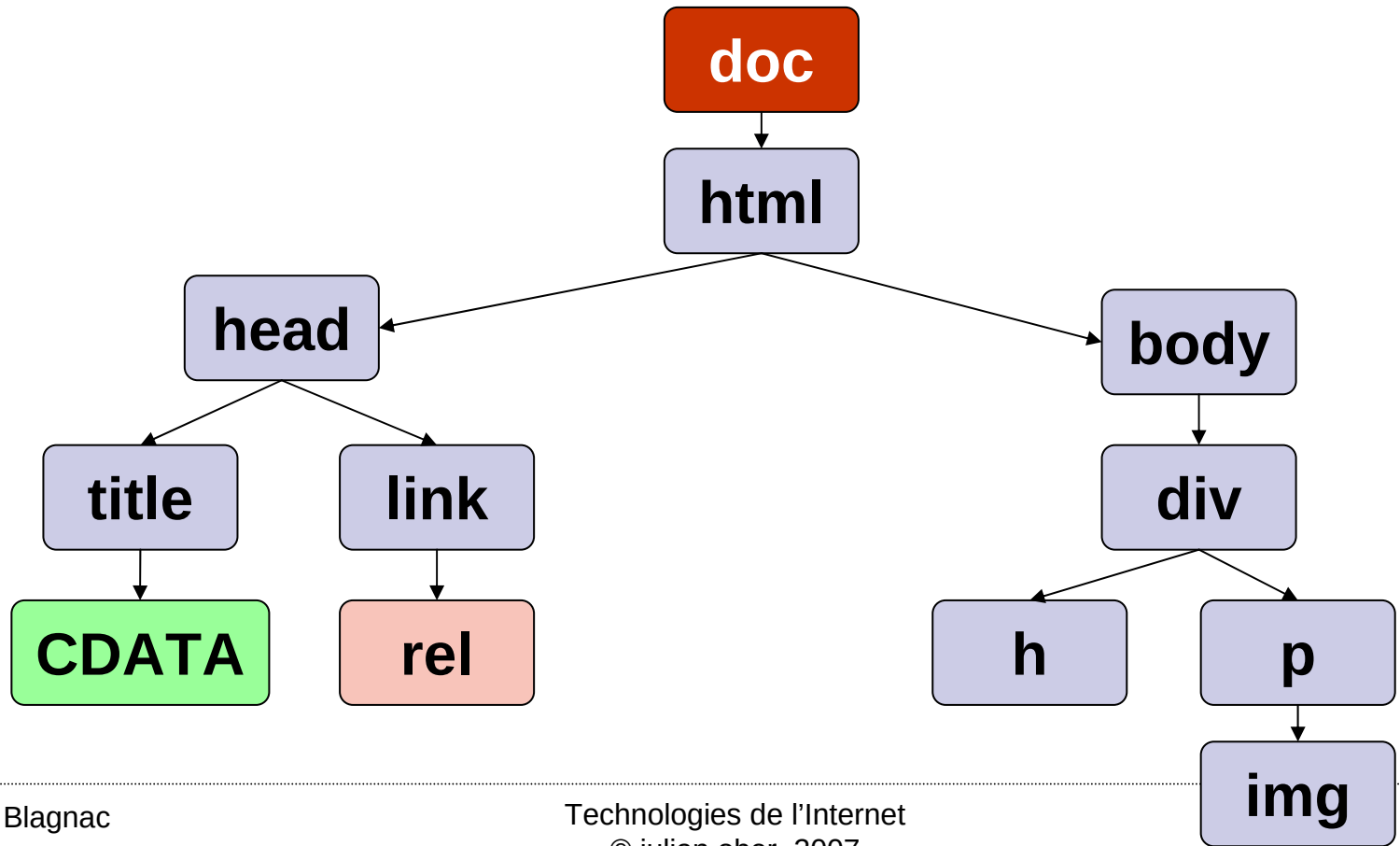
- représentation des éléments d'une page HTML sous forme d'objets, avec propriétés, méthodes, ...
  - page HTML ↔ arbre DOM
  - norme W3C, indépendante du navigateur

- modifier les objets dynamiquement permet de modifier l'aspect de la page

  ⇒ besoin d'un langage (≠ DOM) exécuté par le client (navigateur) pour modifier les objets (ex. JavaScript)

Arbre DOM

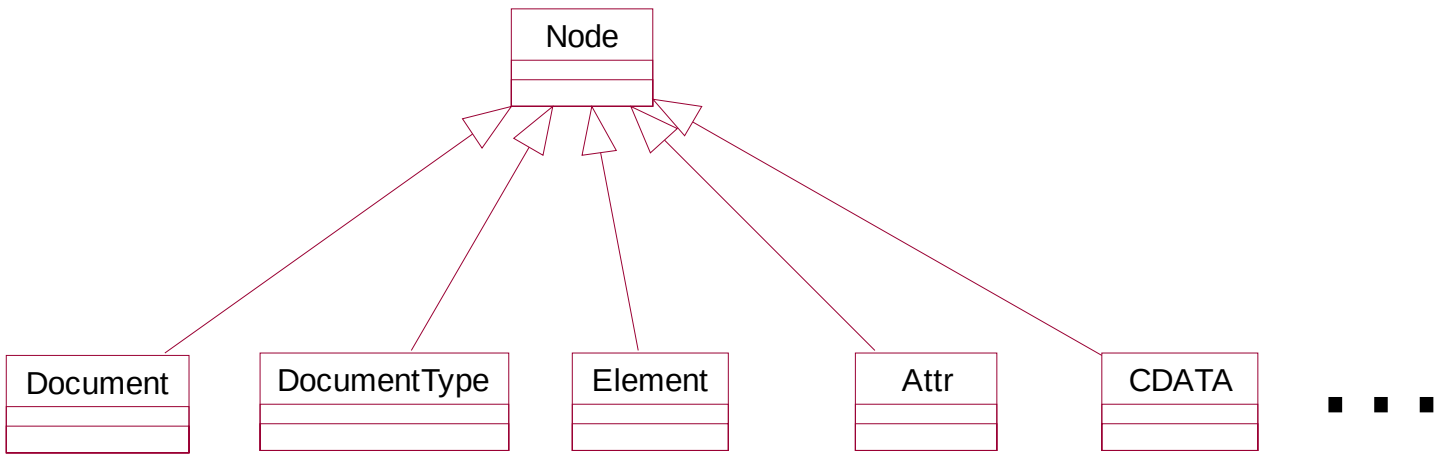
- initialement construit par le navigateur par analyse syntaxique de la page HTML



Types d'objets : niveau générique (XML)

- Node – interface générique pour tous les types de nœuds de l'arbre
  (ex. Document, Element, Attribut, CDATA, …)
- NodeList – liste de nœuds
  (ex. node.childNodes)
- NamedNodeMap – dictionnaire de nœuds en accès par le nom
  (ex. node.attributes, node.)

Types d'objets : niveau générique (XML)



Navigation dans l'arbre

- Document
  - Element getElementById(id)
  - Element getDocumentElement()
  - NodeList getElementsByTagName(tagname)
  - …
- Node
  - Node getFirstChild()
  - Node getNextSibling()
  - Document getOwnerDocument()
  - …
- Element
  - Attr getAttributeNode(name)
  - NodeList getElementsByTagName(tagname)
  - …
- …

Exemple

```
<html>
<head>
<script type="text/javascript">
    i = document.getElementById("myHeader");
    j = document.firstChild.lastChild;
    // devrait être : j = document.getFirstChild().getFirstChild().getNextSibling();
</script>
</head>
<body>

<h1 id="myHeader" >This is a header</h1>

</body>
</html>
```

DOM HTML

Un type d'objet pour chaque élément HTML

- body
- form
- table, tablerow, tablecell
- paragraph
- anchor
- image
- …

Propriétés ↔ attributs de l'élément HTML

- **ex. image :** align, border, ismap, src, …

DOM HTML – styles

- **chaque objet Element a un objet Style associé qui contient l'affectation de style individuelle spécifique à l'objet**

- **on peut changer les propriétés de style:**
  document.getElementById("*id*").style.*property*="*value*"
  - **property : propriété CSS**
    ex. CSS background-image → **DOM** backgroundImage

Exemple

```
<html>
<head>
<script type="text/javascript">
function f() {
    i = document.getElementById("myHeader");
    i.style.backgroundColor = "red";

    link = document.createElement("a");
    linkText = document.createTextNode(" edit");
    link.setAttribute("href", "url.html");
        link.appendChild(linkText);
    i.appendChild(iLink);
}
</script>
</head>
<body>
<h1 id="myHeader" onClick="f()">This is a header</h1>
</body>
</html>
```

JavaScript - introduction

- langage de programmation pour les pages HTML
  - N.B. JavaScript ≠ Java !
- interprété *par le navigateur*
- exemples d'utilisation :
  - changer le texte d'une page
  - réagir à des événements (chargement de la page, clicks, déplacement de souris, ...)
  - valider les données d'un formulaire
  - détecter le type de navigateur du visiteur
  - créer des cookies, ...

Où mettre les scripts

- **dans les entêtes:**

```
<head>
    <script type="text/javascript">
        ...
    </script>
</head>
```

⇒ **interprété au chargement de la page**

⇒ **en principe, définition de fonctions**

- **dans un fichier séparé:**

```
<script src="xxx.js" type="text/javascript"></script>
```

- **dans le corps:**

```
<body>
    <script type="text/javascript">
        document.write("<a href=\"url.html\">a link</a>");
    </script>
</body>
```

⇒ **peut générer du contenu**

Variables

- **créées à l'affectation:**

  var str = "exemple" **ou**

  str = "exemple"

- **nom sensitif à la case**
- **portée**
  - **locales à une fonction**
  - **globales, visibles dans toutes les fonctions de la page**
- **pas de définition de type (→ objet)**

Instructions de contrôle

```
if (time < 10) {
        document.write("Good morning!")
} else {
        document.write("Good day!")
}
```

```
switch (theDay)
{
case 0:
  document.write("It's finally Sunday")
  break
default:
  document.write("looking forward…")
}
```

Instructions de contrôle

```
for (var=0;var<=5;var=var+1)
{
        // code
        if (var=3) break;
}
```

```
while (var<=endvalue)
{
        // code
}
```

Tableaux, plus d'instructions

- **type Array :**

```
var semaine=new Array()
semaine[0]="Dimanche"
semaine[1]="Lundi"
semaine[2]="Mardi"
```

- **instruction for..in :**

```
for (j in semaine) {
      document.write(j+"<br/>");
}
```

Opérateurs

les mêmes qu'en C/C++/Java !

+,-,*,/,%...
++, --
=, +=, *=, %=,...
==,===
&&, ||, !
_?_:_
...

Objets

JavaScript est orienté objet

- types
  - prédéfinis
  - nouveaux (utilisateur)
- accès aux propriétés :
  - exemple (pour txt="toto"):
    txt.length
- accès aux méthodes :
  - exemple
    document.write(txt.toUpperCase());

Types d'objets prédéfinis

- **String**
  - +, indexOf(), match(),…
  - bold(), small()…
- **Date**
  - var myDate=new Date() **// initialisé à la date courante**
  - getDay(), getMonth(), setMonth()…
- **Boolean**
- **Math**
  - min(x,y), abs(x), log(x), …

Fonctions

```
function prod(a,b)
{
        alert("calcul du produit")
        x=a*b
        return x
}
```

Types utilisateur

Définition

```
function getArea(){
 return (this.radius*this.radius*3.14);
}


function Circle(radius){
 this.radius = radius;
 this.getArea = getArea;
}
```

Utilisation

```
var bigCircle = new Circle(100);

alert(bigCircle.getArea());
```

Définition alternative (sans type)

```
var person = new Object()
person.firstname="John"
person.lastname="Doe"
```

Variables prédéfinies relatives au navigateur

- **Window – propriétés & méthodes de la fenêtre de navigation**
  - name, status, location, closed, …
  - alert(), close(), print(), scrollBy(), resizeBy(),…
  - **objet par défaut pour les appels aux fonctions**

```
function f() {
    alert("message1");
    window.alert("message2");
}

f()             // appel à f()
window.f()      // idem
```

Variables prédéfinies relatives au navigateur

- **Location – propriétés de l'URL courent**
  - hash, host, port, protocol, search…
  - reload(), replace()

Exemple: saut à une ancre (a1) de la même page

```
function jumpNear() { location.hash="a1" }
```

Exemple: chargement d'une autre page

```
function jumpFar() {
    location = "http://www.w3schools.com"
    // ou: location.replace("http://www.w3schools.com")
}
```

Variables prédéfinies relatives au navigateur

- **Navigator – propriétés & méthodes du navigateur en général**
  - appName, appVersion, userLanguage, userAgent,…

Exemple:

```
function detectBrowser()
{
        var browser=navigator.appName
        var b_version=navigator.appVersion
        var version=parseFloat(b_version)
        if ((browser=="Netscape"||browser=="Microsoft Internet Explorer") &&
              (version>=4))
         {   alert("Your browser is good enough!") }
        else
         {   alert("It's time to upgrade your browser!") }
}
```

Variables prédéfinies relatives au navigateur

- **Screen – propriétés de l'écran**
  - availHeight, availWidth, colorDepth, …
- **History – manipuler l'historique**
  - length, back(), forward(), go()

Exemple:

```
function precedente()
{
        history.back();
        // ou : history.go(-1)
}
```

JavaScript & DOM

Version "simplifiée" de DOM :

- **certains méthodes de navigation ne sont pas disponibles (**getNextSibling()**)**
- **d'autres sont renommées et/ou transformées en propriétés**
  (firstChild **au lieu de** getFirstChild()**)**
- **création de "raccourcis"**
  - document.i1 **désigne l'élément qui a** id="i1"
    (pour certains types d'éléments seulement !)
  - table.cells[] **donne les cellules d'un tableau**
  - …

Types prédéfinis DOM HTML

Un type d'objet pour chaque élément HTML

- body
- form
- table, tablerow, tablecell
- paragraph
- anchor
- image
- …

Propriétés ↔ attributs de l'élément HTML

- **ex. image :** align, border, ismap, src, …

Événements

- provoqués par des actions de l'utilisateur sur certains éléments de la page HTML
  - chargement / fermeture de la page (onload, onunload)
  - click souris, passage de la souris sur un élément (onclick, onmouseover, onmouseout…)
  - appui d'une touche (onkeydown, onkeyup…)
  - sélection de texte (onselect)
  - envoi / r.a.z. d'un formulaire (onsubmit, onreset)
  - …
- on peut associer des actions JavaScript à l'événement
  - modèle : *<élément-html onEvenement="actionJS…">…</élément-html>*
  - toutes les combinaisons élément-événement ne sont pas possibles!

Exemples d'événements

- **chargement de la page**

```
<body onload="validerNavigateur()">
```

- **changement d'un champ de texte**

```
<input type="text" id="myId" onchange="transform(this.id)"/>
```

- **passage de la souris**

```
<div id="myId" onmouseover="emphasize(this.id)"
    onmouseout=unemphasize(this.id) >

…

</div>
```

Exemples JavaScript

Créer un lien sans utiliser <a>:

```
<element-html onclick="location.replace(newUrl)">
    …
</element-html >
```

Exemples JavaScript

Mise à jour périodique d'un horloge:

```
<script type="text/javascript">
function startTime()
{
	var today=new Date()
	var h=today.getHours()
	var m=add0(today.getMinutes())
	var s=add0(today.getSeconds())

	document.getElementById('txt').innerHTML=h+":"+m+":"+s
	t=setTimeout('startTime()',500)
}
</script>
…
<body onload="startTime()">
<div id="txt"></div>
```

Exemples JavaScript

Attributs des événements: coordonnés de la souris, bouton appuyé,…

```
<script type="text/javascript">
function show_coords(e)
{
	x=e.clientX
	y=e.clientY
	alert("X coords: " + x + ", Y coords: " + y)
	alert("bouton: " + e.button)
	alert("élément: " + e.target.innerHTML)
}
</script>

<body onmousedown="show_coords(event)">
```

Exemples JavaScript

Image dynamique

```
<script type="text/javascript">
function changeSrc()
{
  document.img0.src="hackanm.gif"
}
</script>
…

<img id="img0" src="compman.gif" alt="image" />
```