POLYTECH
TOURS
Département Informatique

PROGRAMMATION WEB DI3

JAVASCRIPT

Cyrille FAUCHEUX cyrille.faucheux@etu.univ-tours.fr

Année 2010-2011

Le JavaScript

- Langage de script (compilation non nécessaire).
  - Essentiellement utilisé pour le développement de pages Web dynamiques.
  - Non typé.
  - Orienté objet.

Le JavaScript

- Historique
  - Créé par Netscape en 1995
    - Partenariat avec Sun,
    - Inspiré de Java (**ne pas confondre !**).
  - Microsoft réagit en sortant le JScript (procès, gagné par Sun),
  - Tentative de standardisation/normalisation
    - Netscape soumet JavaScript à l'ECMA International
      - Naissance du standard ECMAScript (ECMA-262).

Le JavaScript

- Historique
  - Version officielle de JavaScript maintenue par Mozilla et Adobe,
  - Sur-ensemble d'ECMAScript,
  - Mais chacun est libre d'implémenter son propre moteur JavaScript, du moment qu'il est conforme avec la norme ECMA-262,
  - Donc possibilité d'extension
    - Ex : WebGL.
  - Et donc de non portabilité (rare).

Le JavaScript

- Historique
  - Actuellement
    - Implémentation « officielle » : version 1.8 (ECMA-262 v3 (1999) + extensions)
    - Version 2.0 en cours de développement…

POLYTECH TOURS
Département Informatique

Possibilités et restrictions

Possibilités et restrictions

- Possibilités
  - Manipulation du DOM (Document Object Model)
  - Répondre à des évènements (souris, clavier,…)
- Restrictions
  - ~~Aucune fonctionnalité graphique~~
    - Balise `canvas` en HTML 5, WebGL et SVG.
  - Pas d'accès au système de fichier
    - Mais fonctionnalité **Web Storage** du HTML 5.
      - http://www.w3.org/TR/webstorage/
      - Quand ça sera implémenté…

POLYTECH
TOURS
Département Informatique

Insertion de scripts

Insertion de scripts

- 3 méthodes (comme pour les CSS)
  - Méthode intra-ligne
  - Méthode globale
  - Méthode importée

- Remarque préliminaire : les scripts sont exécutés **dès qu'ils sont rencontrés** (ils ne peuvent interagir avec ce qui est défini après eux) !

Insertion de scripts

- Méthode intra-ligne

```
<input type="button" onclick="myFunction();" />
```

- Principalement utilisée pour répondre à des évènements.
- Possibilité de lister plusieurs actions en les séparant par des « ; ».

Insertion de scripts

- Méthode globale

```
<!doctype html>
<html>
<head>
   <title></title>
   <script langage="javascript">
        <!-- Scripts JavaScript -->
   </script>
</head>
<body>…</body>
</html>
```

Peut aussi être dans le corps de la page.

Insertion de scripts

- Méthode importée

```
<!doctype html>
<html>
<head>
   <title></title>
   <script src="fichier.js"></script>
</head>
<body>…</body>
</html>
```

Peut aussi être dans le corps de la page.

Mais c'est pas très propre…

POLYTECH TOURS
Département Informatique

Syntaxe

Syntaxe

- Préliminaires
  - Sensible à la casse
  - Commande JavaScript : se termine par un « ; »
    - `document.write("Hello World");`
  - Commentaires
    - `/* … */` (multiligne)
    - `// …`

Syntaxe

- Les variables
  - Langage dynamique non typé
    - Pas nécessaire de déclarer les variables (mais conseillé).
      - var myVar; / var myVar = …;
    - Inutile de préciser le type.
  - Nom des variables
    - Caractères alphanumériques, _, $.
  - Portée des variables
    - Déclarée dans le script : portée globale,
    - Déclarée dans une fonction avec le mot clé `var` : portée locale (globale si déclaré sans `var`).

Syntaxe

- Les variables
  - Types des variables
    - Simples
      - Entier, flottant, chaîne de caractères, booléen
        - `var myVar = 42;`
        - `var myVar = "something";`
        - `var myVar = myOthervar - 4;`
        - `var myVar = true;`
    - Complexe
      - Tableau, objet, …
        - `var myVar = new Array(1, 2, 3);`
        - `var myVar = [1, 2, 3]`
        - `var myVar = new Object();`
        - …

Syntaxe

- Les variables
  - Libération de mémoire
    - Opérateur `delete`
      - Supprime la référence à l'objet pointé (il peut toujours exister en mémoire si d'autres références existent)
      - Libère l'objet si aucune autre référence n'existe.

```
var a = [1, 2, 3]
var b = a;
b[3] = 4; // a = b = [1, 2, 3, 4]
delete b; // a vaut toujours [1, 2, 3, 4]
```

      - Les références n'existent que pour les tableaux et les objets.
      - Le *Garbage Collector* fait le même travail (potentiellement moins efficace).

Syntaxe

- Les chaînes de caractères
  - Délimitées par des guillemets simples « ’ » ou doubles « " ».
  - On préfèrera les guillemets doubles.
  - Les simples seront utiles dans certains cas :

```
<input type="button" onclick="alert(’Hello’);" />
```

  - Les caractères spéciaux
    - \n : saut de ligne (attention, saute une ligne **dans la source**, pas dans le **rendu graphique**)
    - \t : tabulation (idem)
    - …

Syntaxe

- Les opérateurs arithmétiques (`y=5`)

Opérateur	Description	Exemple	Resultat
+	Addition	`x = y + 2`	`x = 7`
-	Soustraction	`x = y - 2`	`x = 3`
*	Multiplication	`x = y * 2`	`x = 10`
/	Division	`x = y / 2`	`x = 2.5`
%	Modulo	`x = y % 2`	`x = 1`
++	Pré-incrémentation	`x = ++y`	`x = 6 (y = 6)`
++	Post-incrémentation	`x = y++`	`x = 5 (y = 6)`
--	Pré-incrémentation	`x = --y`	`x = 4 (y = 4)`
--	Post-incrémentation	`X = y--`	`x = 5 (y = 4)`

Syntaxe

- Les opérateurs d'assignement (`x=10, y=5`)

Opérateur	Exemple	Equivalent à	Résultat
=	`x = y`		`x = 5`
+=	`x += y`	`x = x + y`	`x = 15`
-=	`x -= y`	`x = x - y`	`x = 5`
*=	`x *= y`	`x = x * y`	`x = 50`
/=	`x /= y`	`x = x / y`	`x = 2`
%=	`x %= y`	`x = x % y`	`x = 0`

Syntaxe

- Les opérateurs de comparaison (`x=5`)

Opérateur	Description	Exemple
`==`	Egalité	`x == 5 est vrai` `x == "5" est vrai`
`===`	Egalité et type	`x === 5 est vrai` `x === "5" est faux`
`!=`	Différence	`x != 8 est vrai`
`>`	Supérieur	`x > 8 est faux`
`<`	Inférieur	`x < 8 est vrai`
`>=`	Supérieur ou égal	`x >= 8 est faux`
`<=`	Inférieur ou égal	`x <= 8 est vrai`

Syntaxe

- Les opérateurs logiques (x=6, y=3)

Opérateur	Description	Exemple
`&&`	ET logique	`(x < 10 && y > 1) est vrai`
`||`	OU logique	`(x==5 || y==5) est faux`
`!`	NON logique	`! (x==y) est vrai`

Syntaxe

- Les opérateurs binaires (x = 0b00000010)

Opérateur	Description	Exemple
<<	Décalage à gauche	x << 1 vaut 0b00000100
>>	Décalage à droite	x >> 1 vaut 0b00000001
&	ET binaire	x & 0b00000011 vaut 0b00000010
|	OU binaire	x | 0b00000001 vaut 0b00000011
^	XOR binaire	x ^ 0b00000011 vaut 0b00000001
~	NON binaire	~x vaut 0b11111111111111111111111111111111111111111111111111111111111101

Syntaxe

- L'opérateur « + » et les chaînes de caractères
  - Opérateur de concaténation

```
var myVar = "abc" + "def"; // myVar = "abcdef"
```

  - Si au moins un des membres de l'opération est une chaine
    - L'autre est transformé en chaine puis concaténé

```
var myVar = 12 + "34"; // myVar = "1234"
var myVar = 1.2 + "34"; // myVar = "1.234"
```

    - Surcharge de la méthode `toString()` pour les objets.

Syntaxe

- Les fonctions

```
function myFunc(arg1, …, argX)
{
  var result = myOtherFunc();
  …
  return result;
}
```

  - Dans une fonction, si une variable n'est pas déclarée avec le mot-clé `var`, alors c'est une variable **globale**.

Syntaxe

- Les structures de programmation

```
if() {
  …
} else {
  …
}
```

Syntaxe

- Les structures de programmation

```
switch(variable){
  case 1:
    …
    break;
  case 2:
    …
    break;
  default:
    …
    break;
}
```

Syntaxe

- Les structures de programmation

```
for(var i = 0; i < 5; i++)
{
  ...
}

for(var i in myArray)
{
  ...
}
```

Syntaxe

- Les structures de programmation

```
while(condition) {

}
```

Syntaxe

- Les structures de programmation
  - `break`
    - Termine une boucle sans attendre la dernière itération.
  - `continue`
    - Saute à l'itération suivante d'une boucle.

Syntaxe

- Les fenêtres
  - Message simple
    - `alert(message)`
  - Confirmation
    - `confirm(message)`
    - Retourne `true` ou false.
  - Saisie
    - `prompt(message, default)`
    - Retourne la valeur saisie (sous forme de chaîne) ou `null`.

Syntaxe

- Les exceptions
  - Une exception peut être un entier, une chaîne ou un objet.
  - Lancer une exception : `throw "Erreur ";`
  - Attraper une exception :

```
try {
  …
}
catch(err) {
  …
}
```

POLYTECH TOURS
Département Informatique

Les évènements

Syntaxe

- Les évènements

Evènement/attribut	Se déclenche quand…
onblur	L'élément perd le focus (champ de saisie).
onchange	Le contenu du champ change.
onclick	Un clic est détecté sur l'élément.
ondblclick	Un double click est détecté sur l'élément.
onerror	Une erreur est survenue durant le chargement de la page.
onfocus	L'élément gagne le focus (champ de saisie).
onkeydown	Une touche est pressée.
onkeypress	Une touche est pressée ou reste pressée.
onkeyup	Une touche est relâchée.
onmousedown	Un bouton de la souris est pressé.

Syntaxe

- Les évènements

Evènement/attribut	Se déclenche quand…
onmousemove	La souris bouge au dessus de l'élément.
onmouseout	La souris bouge en dehors de l'élément.
onmouseover	La souris bouge entre au dessus de l'élément.
onmouseup	Un bouton de la souris est relâché.
onresize	La fenêtre est redimensionnée.
onselect	Du texte est sélectionné.
onunload	L'utilisateur quitte la page.

- Syntaxe

```
<p onclick="myFunction()">…</p>
element.onclick = myFunction();
element.addEventListener('click',myFunction,false);
```

Syntaxe

- Les évènements
  - Pour récupérer les informations concernant l'évènement (souris, clavier), il faut ajouter un argument à la fonction déclenchée par l'évènement.
  - Cet argument sera automatiquement passé à la fonction.

```
<p onmousedown="myFunction(event)">…</p>
```

Syntaxe

- Les évènements
  - Attention, tous les évènement ne fournissent pas forcément ces arguments.

Propriété	Signification
`event.altkey`	Booléen indiquant si la touche ALT était pressée.
`event.button`	Indique quel bouton de la souris a été activé (attention sous Internet Explorer)
`event.clientX`	Coordonnée en X de la souris dans le navigateur.
`event.clientY`	Coordonnée en Y de la souris dans le navigateur.
`event.ctrlkey`	Booléen indiquant si la touche CTRL était pressée.

Syntaxe

- Les évènements

Propriété	Signification
`event.metakey`	Booléen indiquant si la touche SPECIALE était pressée.
`event.target`	Retourne l'élément pour lequel
`event.screenX`	Coordonnée en X de la souris dans l'écran.
`event.screenY`	Coordonnée en Y de la souris dans l'écran.
`event.shiftkey`	Booléen indiquant si la touche SHIFT était pressée.

- Plus de détail
  - http://www.w3.org/TR/2003/WD-DOM-Level-3-Events-20030331/ecma-script-binding.html

POLYTECH TOURS
Département Informatique

39

La programmation orientée objet

La POO

- La classe `String`

Méthode	Description
`charAt()`	Returns the character at the specified index
`charCodeAt()`	Returns the Unicode of the character at the specified index
`concat()`	Joins two or more strings, and returns a copy of the joined strings
`fromCharCode()`	Converts Unicode values to characters
`indexOf()`	Returns the position of the first found occurrence of a specified value in a string
`lastIndexOf()`	Returns the position of the last found occurrence of a specified value in a string

La POO

- La classe `String`

Méthode	Description
`match()`	Searches for a match between a regular expression and a string, and returns the matches
`replace()`	Searches for a match between a substring (or regular expression) and a string, and replaces the matched substring with a new substring
`search()`	Searches for a match between a regular expression and a string, and returns the position of the match
`slice()`	Extracts a part of a string and returns a new string

La POO

- La classe `String`

Méthode	Description
`substring()`	Extracts the characters from a string, between two specified indices
`toLowerCase()`	Converts a string to lowercase letters
`toUpperCase()`	Converts a string to uppercase letters
`valueOf()`	Returns the primitive value of a String object
`split()`	Splits a string into an array of substrings
`substr()`	Extracts the characters from a string, beginning at a specified start position, and through the specified number of character

La POO

- La classe `Array`

Méthode	Description
`concat()`	Joins two or more arrays, and returns a copy of the joined arrays
`join()`	Joins all elements of an array into a string
`pop()`	Removes the last element of an array, and returns that element
`push()`	Adds new elements to the end of an array, and returns the new length
`reverse()`	Reverses the order of the elements in an array
`shift()`	Removes the first element of an array, and returns that element

La POO

- La classe `Array`

Méthode	Description
`slice()`	Selects a part of an array, and returns the new array
`sort()`	Sorts the elements of an array
`splice()`	Adds/Removes elements from an array
`toString()`	Converts an array to a string, and returns the result
`unshift()`	Adds new elements to the beginning of an array, and returns the new length
`valueOf()`	Returns the primitive value of an array

La POO

- La classe `Date`

Méthode	Description
`getDate()`	Returns the day of the month (from 1–31)
`getDay()`	Returns the day of the week (from 0–6)
`getFullYear()`	Returns the year (four digits)
`getHours()`	Returns the hour (from 0–23)
`getMilliseconds()`	Returns the milliseconds (from 0–999)
`getMinutes()`	Returns the minutes (from 0–59)
`getMonth()`	Returns the month (from 0–11)
`getSeconds()`	Returns the seconds (from 0–59)
`getTime()`	Returns the number of milliseconds since midnight Jan 1, 1970

La POO

- La classe `Date`

Méthode	Description
`setDate()`	Sets the day of the month (from 1–31)
`setFullYear()`	Sets the year (four digits)
`setHours()`	Sets the hour (from 0–23)
`setMilliseconds()`	Sets the milliseconds (from 0–999)
`setMinutes()`	Set the minutes (from 0–59)
`setMonth()`	Sets the month (from 0–11)
`setSeconds()`	Sets the seconds (from 0–59)
`setTime()`	Sets a date and time by adding or subtracting a specified number of milliseconds to/from midnight January 1, 1970
…	

La POO

- La classe `Math`

Constante	Description
`E`	Returns Euler's number (approx. 2.718)
`LN2`	Returns the natural logarithm of 2 (approx. 0.693)
`LN10`	Returns the natural logarithm of 10 (approx. 2.302)
`LOG2E`	Returns the base-2 logarithm of E (approx. 1.442)
`LOG10E`	Returns the base-10 logarithm of E (approx. 0.434)
`PI`	Returns PI (approx. 3.14159)
`SQRT1_2`	Returns the square root of 1/2 (approx. 0.707)
`SQRT2`	Returns the square root of 2 (approx. 1.414)

La POO

- La classe `Math`

Méthode	Description
`abs(x)`	Returns the absolute value of x
`acos(x)`	Returns the arccosine of x, in radians
`asin(x)`	Returns the arcsine of x, in radians
`atan(x)`	Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians
`atan2(y,x)`	Returns the arctangent of the quotient of its arguments
`ceil(x)`	Returns x, rounded upwards to the nearest integer
`cos(x)`	Returns the cosine of x (x is in radians)
`exp(x)`	Returns the value of E^x

La POO

- La classe `Math`

Méthode	Description
`floor(x)`	Returns x, rounded downwards to the nearest integer
`log(x)`	Returns the natural logarithm (base E) of x
`max(x,y,z,...,n)`	Returns the number with the highest value
`min(x,y,z,...,n)`	Returns the number with the lowest value
`pow(x,y)`	Returns the value of x to the power of y
`random()`	Returns a random number between 0 and 1
`round(x)`	Rounds x to the nearest integer
`sin(x)`	Returns the sine of x (x is in radians)
`sqrt(x)`	Returns the square root of x
`tan(x)`	Returns the tangent of an angle

La POO

- Création de classes
  - Définition d'une classe
    - Une classe est définie par son constructeur
      - Initialisation des attribut pour lesquels c'est possible,
      - Les autres attributs seront déclarés dynamiquement.

```
function MyObject(myVar)
{
  this.attr1 = myVar; // Ne pas utiliser var
  …
}
var a = new MyObject(3);
alert(typeof a); // Affiche "object"
alert(a.attr1); // Affiche 3
a.attr2 = 12;
```

La POO

- Création de classes
  - Définition des méthodes
    - Dans le constructeur

```
function MyClass(…)
{
  …
  this.myFunction = function(…) {…}
}
```

    - Utilisation du prototype (à l'extérieur du constructeur)

```
MyClass.prototype.myFunction = function(…) {…}
```

      - Permet aussi de définir dynamiquement des méthodes

```
a.myFunction(…);
```

La POO

- Attributs dynamiques
  - Les attributs non-définis dans le constructeur sont instanciés dynamiquement et accessibles depuis `this`.

```
var a = new TheClass(…);
var b = new TheClass(…);
a.attr1 = 23;
b.attr2 = 42;
```

La POO

- Méthodes dynamiques
  - Les méthodes définies sur une instance d'une classe ne sont pas définis pour les autres instances de cette classe.

```
var a = new TheClass(…);
a.myFunc = function(…) {…};
var b = new TheClass(…);
a.myFunc(); // OK
b.myFunc(); // Undefined
```

  - Utiliser la méthode « Class.Prototype ».

POLYTECH
TOURS
Département Informatique

JavaScript

- **Les constantes, méthodes et objets prédéfinis**

Les constantes, méthodes et objets prédéfinis

Constante	Description
`Infinity`	A numeric value that represents positive infinity
`NaN`	"Not-a-Number" value
`undefined`	Indicates that a variable has no value

Fonction	Description
`decodeURI()`	Decodes a URI
`decodeURIComponent()`	Decodes a URI component
`encodeURI()`	Encodes a URI
`encodeURIComponent()`	Encodes a URI component
`escape()`	Encodes a string

Les constantes, méthodes et objets prédéfinis

Fonction	Description
`eval()`	Evaluates a string and executes it as if it was script code
`isFinite()`	Determines whether a value is a finite, legal number
`isNaN()`	Determines whether a value is an illegal number
`Number()`	Converts an object's value to a number
`parseFloat()`	Parses a string and returns a floating point number
`parseInt()`	Parses a string and returns an integer
`String()`	Converts an object's value to a string
`unescape()`	Decodes an encoded string

Les constantes, méthodes et objets prédéfinis

Objet	Description
`window`	Informations et méthodes concernant le rendu de la page (taille, scroll, statut, …)
`screen`	Informations concernant l'écran (résolution, profondeur des couleurs, …)
`document`	Informations et méthodes concernant la page (titre, url, accès aux balises (`getElementById()`), création de balises (`createElement()`), cookies, …)
`navigateur`	Informations concernant le navigateur (nom, version, langue, …)
`location`	L'URL complète de la page.
`history`	Informations et méthodes concernant l'historique de navigation.

POLYTECH
TOURS
Département Informatique

Les cookies

Les cookies

- Ensemble de couples clé/valeur stocké par le navigateur et transmis à chaque requête au serveur.
  - Permettent de stocker des informations concernant l'utilisateur (session, préférences, …).
  - Le serveur et le navigateur peuvent interagir avec les cookies.
  - Il sont transmis dans l'en-tête des requêtes.

Les cookies

- Ecrire un cookie

```
document.cookie = "clé=valeur; expires=date;
  path=chemin"
```

  - Avec :
    - `clé` et `valeur` le couple identifiant l'information,
    - `date` la date d'expiration au format GMT (`Fri, 30 Apr 2010 12:31:22 GMT`),
    - `chemin` l'URL minimale pour laquelle ce cookie est valide.
  - Exemple :

```
document.cookie = "authkey=31BC…40E1; expires=Fri,
  30 Apr 2010 12:35:56 GMT; path=/"
```

Les cookies

- Ecrire un cookie
  - Cet appel **ajoute** une valeur au cookie dans le **navigateur**.
  - Si aucune date d'expiration n'est spécifiée
    - Le cookie sera supprimé quand le navigateur sera fermé.
  - Si la date d'expiration est située dans le futur
    - Le cookie sera supprimé à cette date.
  - Si la date d'expiration est située dans le passé
    - Le cookie est supprimé aussitôt.

Les cookies

- Lire un cookie
  - `document.cookie` renvoie la liste des clés/valeur valides à cet instant

```
var c = document.cookie;
// c = "authkey=ABC…123; SID=DEF…456"
```

  - A vous d'écrire des fonctions pour manier plus facilement les cookies,
  - Vous aurez rarement à manier les cookies en JavaScript.

POLYTECH
TOURS
Département Informatique

Le Document Object Model

Le DOM

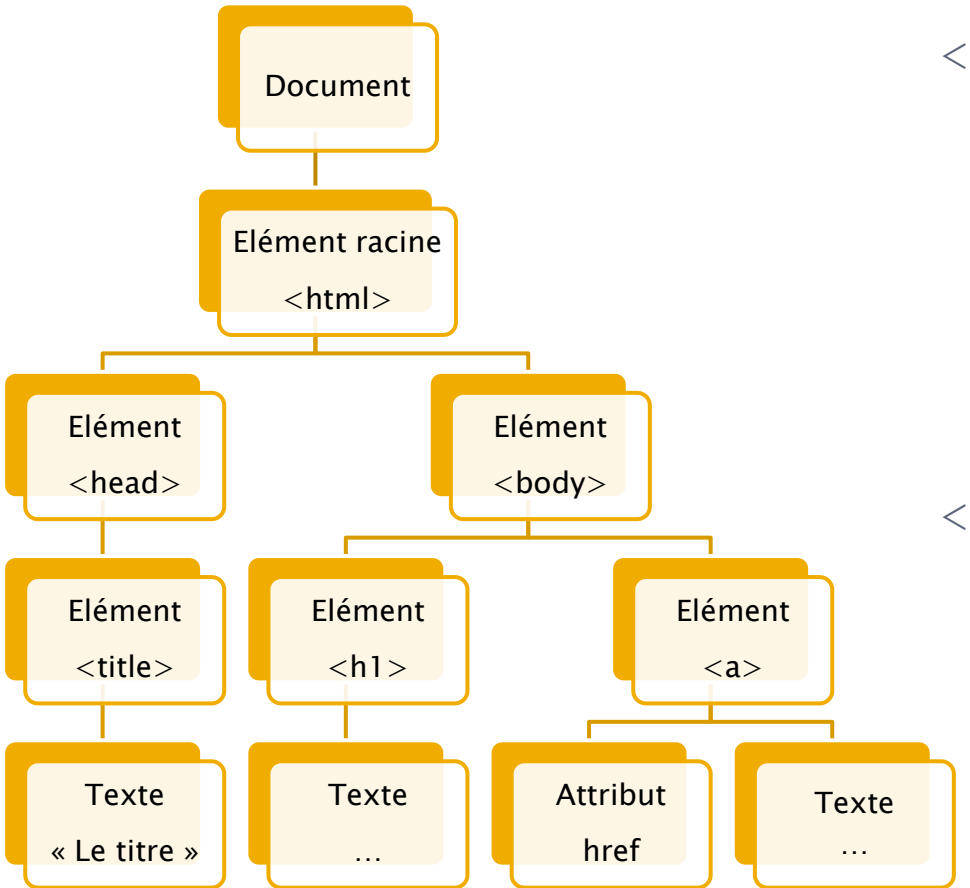
- **D**ocument **O**bject **M**odel
  - Standard du W3C.
  - Ce n'est **pas** du JavaScript.
  - **Structure de donnée** qui expose le contenu d'un document XML sous forme d'**arbre**.

Le DOM

- Tous les éléments du document HTML sont des **nœuds** de l'arbre.
  - Les **balises** sont des nœuds,
  - Les **attributs** sont des nœuds,
  - Le **texte** contenu dans une balise est un nœud.

Le DOM

- Tous ces éléments sont liés entre eux pour constituer l'**arbre du document**.




```
<html>
  <head>
    <title>Le titre</title>
  </head>
  <body>
    <h1>…</h1>
    <a href="…">…</a>
  </body>
</html>
```

Le DOM

- Dans un tel arbre :
  - Le premier nœud est appelé la **racine**,
  - Chaque nœud (sauf la racine) possède un **unique parent**,
  - Un nœud peut avoir **plusieurs enfants**,
  - Une **feuille** est un nœud qui n'a pas d'enfant,
  - Des nœuds **frères** sont des nœuds qui possèdent le **même parent**,
  - Des nœuds frères sont **ordonnés** suivant leur **position** dans le document.

Le DOM

- Interface de programmation
  - Le DOM possède notamment une interface de programmation en JavaScript.
  - Elle est définie par un ensemble de propriétés et de méthodes.

Le DOM

Principales propriétés	Description
`x.innerHTML`	Le contenu de l'élément x (ne fait pas partie du DOM mais est supportée par la plupart des navigateurs).
`x.nodeValue`	Le contenu de l'élément x (nœud texte et attributs).
`x.nodeName`	Le nom de l'élément x.
`x.parentNode`	Le nœud parent de l'élément x.
`x.childNodes`	La liste des enfants de l'élément x.
`x.attributes`	La liste des attributs de l'élément x.
`x.firstChild`	Le premier fils de l'élément x.
`x.lastChild`	Le dernier fils de l'élément x.

- Le contenu d'une balise est présent dans un nœud texte virtuel qui est le premier fils de la balise.

Le DOM

Principales méthodes	Description
`x.getElementById(id)`	Renvoi le nœud fils possédant cet l'id.
`x.getElementsByTagName(name)`	Renvoi les nœuds fils du type spécifié.
`x.appendChild(node)`	Ajoute le nœud spécifié en tant que dernier fils de l'élément x.
`x.removeChild(node)`	Supprime le nœud spécifié des fils de l'élément x.

Propriétés spéciales	Description
`document.body`	Le nœud correspondant à la balise <body>.
`x.style.propriété`	Permet d'accéder aux propriétés CSS de l'élément x (attention : les propriétés CSS dont le nom est composé, `background-color` devient par exemple `backgroundColor`).
`...`	Certaines balises HTML définissent des propriétés et méthodes spéciales. Plus de détails sur : http://www.w3schools.com/jsref/default.asp

Le DOM

- Exemples
  - Modifie le texte de l'élément portant l'id « message ».

```
document.getElementById("message").firstChild.nod
  eValue = "Hello World!";
```

  - Modifie la couleur de fond de la page.

```
document.body.style.backgroundColor="blue";
```

POLYTECH
TOURS
Département Informatique

JavaScript

- **AJAX (Asynchronous JavaScript And XML)**

AJAX

- But :
  - Permettre à un navigateur d'envoyer des requêtes
    - JavaScript pour gérer l'envoi des requêtes et la réception des données,
    - Le XML pour structurer les données transmises,
    - Le tout de façon asynchrone puisqu'il n'est pas nécessaire de recharger la page.
- Première utilisation d'AJAX :
  - Suggestion du moteur de recherche Google.

AJAX

- L'objet XMLHttpRequest
  - Cet objet est responsable de l'envoi d'une requête et de la réception/du traitement de la réponse.
  - IE7+, Firefox, Chrome, Safari, & Opera

```
xmlhttp=new XMLHttpRequest();
```

  - IE5, IE6

```
xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
```

AJAX

- L'envoi d'une requête

Méthode	Description
`open(method,url,async)`	Construit la requête en spécifiant le type de requête, l'URL et sa synchronicité. *method* : le type de requête (GET ou POST). *url* : l'adresse de la ressource sur le serveur. *async* : true (asynchrone) ou false (synchrone)
`send(string)`	Envoie la requête au serveur. *string* : Données lors de l'envoi de requêtes de type POST.

AJAX

- L'envoi d'une requête
  - Exemples
    - GET

```
xmlhttp.open("GET", "myFile.php", true);
xmlhttp.send();
```

    - POST

```
xmlhttp.open("POST", "myFile.php", true);
xmlhttp.send("nom=Bob&prenom=Leponge");
```

AJAX

- L'envoi d'une requête
  - Les en-têtes

Méthode	Description
`setRequestHeader(header,value)`	Permet l'ajout d'en-têtes . *header* : le nom de l'en-tête *value* : la valeur de l'en-tête

  - La synchronicité
    - Asynchrone : requête non bloquante (d'autres scripts peuvent s'exécuter pendant la requête),
    - Synchrone : requête bloquante (rien ne s'exécute durant la requête).

AJAX

- La réponse

Propriété	Description
`responseText`	Récupère la réponse sous forme de chaîne de caractères.
`responseXML`	Récupère la réponse sous forme d'objet XML.

  - Format texte : donnée simple (message) ou HTML généré depuis le serveur.
  - Format XML : données structurées/complexes qu'il faudra parser pour pouvoir les afficher.

Ajax

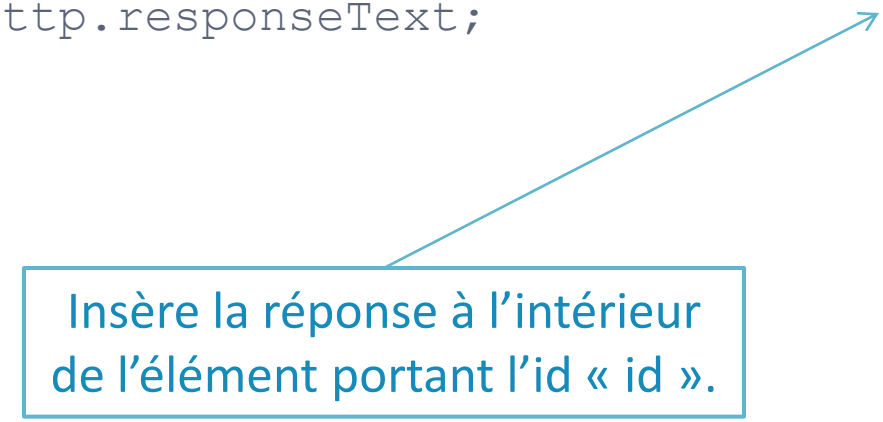
- L'état de la requête

Propriété	Description
onreadystatechange	Pointeur sur une fonction (à définir) qui sera appelée à chaque changement de l'état.
readyState	Status de la requête XMLHttpRequest. 0 : non initiélisée 1 : connexion avec le serveur établie 2 : requête reçue par le serveur 3 : réponse en cours de traitement 4 : requête terminée et réponse disponible.
status	Statut HTTP 200 : OK 404 : Page not found

Ajax

- L'état de la requête
  - On s'intéressera surtout à la fin de la requête

```
xmlhttp.onreadystatechange = function()
{
  if (xmlhttp.readyState == 4 && xmlhttp.status == 200)
  {
    document.getElementById("id").innerHTML =
  xmlhttp.responseText;
  }
}
```



AJAX

- Exemples
  - http://www.w3schools.com/Ajax/ajax_examples.asp

POLYTECH
TOURS
Département Informatique

JavaScript

- **Les frameworks JavaScript**

Les frameworks JavaScript

- Avec la démocratisation d'AJAX, apparition de nombreux frameworks Javascript
  - Utilitaires
    - Prototype,
    - JQuery,
    - …
  - Effets visuels
    - Script.aculo.us + Prototype
    - JQuery
    - …

POLYTECH TOURS
Département Informatique

Remarques & outils

Remarques

- Le JavaScript est un outil permettant de simplifier/rendre plus plaisante la navigation sur votre site.
- Le JavaScript n'est pas forcément présent sur tous les navigateurs (sécurité, accessibilité, …).
  - Vous **devriez** donc faire en sorte que votre site fonctionne même **sans** JavaScript (sauf si ce n'est pas possible).

Outils

- Console Javascript, FireBug, …