

Langages Web 1



Introduction a JavaScript

Florent Nicart

Université de Rouen

2016–2017

Généralités



- Créé en 1995 par Netscape en association avec Sun Microsystems,
- Javascript est un langage de script interprété multiparadigmes : orienté objet, à prototypes (sans classes), fonctionnel (second ordre)
- Javascript est standardisé par un comité spécialisé, l'ECMA¹ : ECMAScript
- Sert à implémenter la partie dynamique d'une application coté client.

1. European Computer Manufactures Association

Versions et standards

Ver.	date	Équivalences
1.0	Mars 1996	ECMA-262 1re edition/2ème edition
1.1	Août 1996	
1.2	Juin 1997	
1.3	Octobre 1998	
1.4		ECMA-262 3ème edition
1.5	Novembre 2000	
1.6	Novembre 2005	
1.7	Octobre 2006	Iterateurs, let, générateurs
1.8	Juin 2008	Expressions de générateur et de fermeture
1.8.1		Support JSON Natif
1.8.2	22 Juin 2009	MAJ mineurs
1.8.5	27 Juillet 2010	1.8.2 + ECMAScript 5 Compliance

Introduction

Principe
Liaison

Le langage

Syntaxe
Variables
Les Opérateurs
Flot d'exécution
Fonctions
Objets
JSON

L'API D.O.M.

Motivations
Niveaux
DOM Core
DOM HTML
DOM et CSS
Exemples
Conformité

AJaX

Introduction
XMLHttpRequest
exemples

Caractéristiques

Introduction

Principe

Liaison

Le langage

Syntaxe

Variables

Les Opérateurs

Flot d'exécution

Fonctions

Objets

JSON

L'API D.O.M.

Motivations

Niveaux

DOM Core

DOM HTML

DOM et CSS

Exemples

Conformité

AJAX

Introduction

XMLHttpRequest

exemples

Javascript est **dynamique** :

- typage faible et dynamique (types associés aux valeurs et non aux variables) ;
- « basé objets » : pas de classe ni d'héritage, les objets sont des tableaux associatifs (...) ;
- évaluation à l'exécution : *eval()* peut exécuter du code généré. (à la Lisp)

Caractéristiques

Introduction

Principe

Liaison

Le langage

Syntaxe

Variables

Les Opérateurs

Flot d'exécution

Fonctions

Objets

JSON

L'API D.O.M.

Motivations

Niveaux

DOM Core

DOM HTML

DOM et CSS

Exemples

Conformité

AJaX

Introduction

XMLHttpRequest

exemples

Javascript est **fonctionnel** :

- fonction du second ordre : les fonctions sont des objets qui peuvent être mis en variables, paramètres ou valeur de retour de fonction ;
- fonctions anonymes ;
- imbrication de définitions de fonctions, avec portée de la fonction externe étendue à la fonction interne ;
- fermeture (*closure*) : capture lexicale des variables de la fonction englobante (*cf* OCAML).

Caractéristiques

Introduction

Principe

Liaison

Le langage

Syntaxe

Variables

Les Opérateurs

Flot d'exécution

Fonctions

Objets

JSON

L'API D.O.M.

Motivations

Niveaux

DOM Core

DOM HTML

DOM et CSS

Exemples

Conformité

AJAX

Introduction

XMLHttpRequest

exemples

Javascript est basé **prototypes** :

- les prototypes remplacent les classes et l'héritage ;
- les fonctions sont utilisées comme constructeurs lorsque leur appel est préfixé par `new` ;
- Les fonctions sont des méthodes : la distinction est faite lors de l'appel (liaison de `this` ou non).

Liaison avec HTML

Introduction

Principe

Liaison

Le langage

Syntaxe

Variables

Les Opérateurs

Flot d'exécution

Fonctions

Objets

JSON

L'API D.O.M.

Motivations

Niveaux

DOM Core

DOM HTML

DOM et CSS

Exemples

Conformité

AJaX

Introduction

XMLHttpRequest

exemples

- Javascript est principalement² destiné à être exécuté par le client.
- Il est exécuté à l'intérieur d'une machine virtuelle : Google V8 (Chrome), TraceMonkey (Mozilla), etc.
- Ces machines peuvent effectuer de la compilation à la volée (Just-In-time).
- La machine virtuelle accède aux objets du navigateur via des APIs.
- Le code Javascript est inclu ou référencé dans le document HTML ...

2. cf Node.js (V8), CommonJS (spécification), etc.

Liaison avec HTML

- Inclusion du code dans des éléments `<script>` (dans head ou n'importe où dans body) :

En HTML 4 :

```
1 <script type="text/javascript">
2   // <!--
3   Code Javascript
4   // -->
5 </script>
```

En XHTML :

```
1 <script type="text/javascript">
2   // <![CDATA[
3   Code Javascript
4   // ]]>
5 </script>
```

- Inclusion d'un fichier externe :

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

- Code HTML alternatif (dans body) :

```
1 <noscript>
2   Message a afficher en cas d'absence de Javascript
3 </noscript>
```


Liaison avec HTML

L'exécution du code peut être déclenchée de différentes manières :

- une instruction explicite dans un élément `<script>` (exécution au chargement de la page), ex :

```
1 <script type="text/javascript">
2 // <!--
3     alert('Ceci est une boîte de dialogue!');
4 // -->
5 </script>
```

- un évènement déclenché sur un élément, ex :

```
1 <p onclick="javascript:alert('Ouille!') ">
2     ...
3 </p>
```

- clic sur un lien javascript :

```
1 <a href="javascript:alert('Pas un vrai lien!') ">cliquez ici </a>
```


Exemple

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21

```
...  
<head>  
  <title >Test JS</title >  
  <script type="text/javascript">  
    // <!--  
      message = 'Hello<em>World</em>!' ;  
    // -->  
  </script>  
</head>  
<body>  
  <p>Bonjour  
  <script language=' javascript'>  
    alert(message) ;  
    document.write (   
      prompt(' quel<u>est<u>votre<u>nom<u>?', ' Indiquer<u>votre<u>nom<u>ici' )  
    );  
    confirm(' quel<u>bouton<u>allez-vous<u>choisir<u>?' );  
  </script>  
</body>  
</html>
```

Note : les différents éléments <script> partagent le même environnement (cf la variable message).

Syntaxe élémentaire

- La syntaxe est proche de C/C++/Java.
- Le langage Javascript est sensible à la casse.
- Commentaires : `//` ou `/* . . . */`
- Instructions séparées par `;` (optionnel).
-  Le retour de ligne peut terminer implicitement des instructions. Ex :

```
1 function calc(x,y) {  
2     return  
3     x+y ;  
4 }  
5  
6 c = calc(10, 16); // c = undefined !
```

- Toujours utiliser `;` (Compresseurs de code → code sur une ligne).

Les variables 1/2

- Le mot-clé `var` permet de déclarer **explicitement** une ou plusieurs variables.
- La lecture d'une variable non déclarée provoque une erreur.
- L'affectation d'une variable non déclarée la déclare **implicitement**.
- Portée locale à la fonction pour les variables explicites, globale pour les implicites.
- Pas de portée de bloc en Javascript.

```
1  var g;           // Variable globale
2
3  function portee() {
4      var x=5;     // Variable locale.
5      y=10;        // Variable implicite donc globale.
6      y=z;         // Erreur, z non declaree.
7  }
```

Les variables 2/2

- Le nom des variables doit commencer par une lettre et peut contenir des lettre, des chiffres et `_`, les espaces et caractères spéciaux/accrntués sont proscrits.
- Les mots clés du langage sont proscrits,
- Le type est associé à la valeur et non à la variable. Ex :

```
1 var maVariable = 'Philippe'; // Type chaine.  
2 maVariable =10;           // Type entier.
```

- Trois types principaux :
 - **String**
 - **Number** (compris entre 10^{-308} et 10^{308}) avec trois valeurs spéciales : **Positive Infinity** (ou **+Infinity**), **Negative Infinity** (ou **-Infinity**), et **NaN** (Not a Number) résultant d'un calcul invalide.
 - **Boolean** avec les valeur **true** et **false**

Les valeurs spéciales

Introduction

Principe

Liaison

Le langage

Syntaxe

Variables

Les Opérateurs

Flot d'exécution

Fonctions

Objets

JSON

L'API D.O.M.

Motivations

Niveaux

DOM Core

DOM HTML

DOM et CSS

Exemples

Conformité

AJAX

Introduction

XMLHttpRequest

exemples

JavaScript inclut aussi deux types de données spéciaux :

- **null** : signifie l'absence de données dans une variable.
- **undefined** : une variable dont le contenu n'est pas clair car elle n'a jamais stocké de valeur, pas même null est dite non définie (indéfinie).

Les Opérateurs

JavaScript inclut les opérateurs classiques :

- opérateurs arithmétiques : `+` `-` `*` `/` `%`
- pre/post in/décrémentation : `var++` `var-` `++var`
`--var`
- opérateurs logiques : `&&` `||` `!`
- concaténation de chaîne de caractères : `+`
- affectation : `=` `+=` `-=` `*=` `/=`
- comparaisons : `==` `===` `!=` `!==` `<=` `<` `>=` `>`

<code>3</code>	<code>==</code>	<code>'3'</code>	<code>true</code>
<code>2</code>	<code>==</code>	<code>3</code>	<code>false</code>
<code>3</code>	<code>===</code>	<code>3</code>	<code>true</code>
<code>3</code>	<code>===</code>	<code>'3'</code>	<code>false</code>

Instructions de contrôle de flot

Conditionnelle :

```
1  if ( condition ) {  
2      instructions  
3  }  
4  [ else if {  
5      instructions  
6  }]  
7  [ else {  
8      instructions  
9  }]
```

Aiguillage :

```
1  switch ( variable ) {  
2      case 'valeur1' :  
3          Instructions  
4          break ;  
5      ...  
6      default :  
7          Instructions  
8          break ;  
9  }
```

Boucles for :

```
1  for ( i =0; i <N ; i ++ ) {  
2      Instructions  
3  }  
4  
5  for ( p in objet ) {  
6      Instructions  
7  }
```

Boucles while :

```
1  while ( condition ) {  
2      Instructions  
3  }  
4  
5  do {  
6      Instructions  
7  } while ( condition ) ;
```


Déclaration de fonctions

- Syntaxe de déclaration :

```
1  function nom(arg1, ..., argN) {  
2      Instructions  
3      [ return valeur ; ]  
4  }
```

- Comme les variables, les arguments et valeurs de retour ne sont pas typés.
- Les procédures sont des fonctions sans valeur de retour.
- La déclaration ne fixe pas le nombre d'arguments :

```
1  function calc(x,y) {  
2      return x+y;  
3  }  
4  
5  r=calc(10,16, 33); // r = 26, pas d'erreur  
6  r=calc(10);        // r = NaN, pas d'erreur
```

Caractéristiques de langage fonctionnel

- **Fonctions anonymes** (ou *lambda*). Exemple :

```
1 <html>
2   <body onload="setTimeout (function () {
3     alert ('chargement de la page termine il y a une seconde et demie' )
4     }, 1500) ;">
5   </body>
6 </html>
```

- Fonctions imbriquées et **fermeture** (ou *clôture*³) :
Capture de l'environnement englobant par la fonction interne. Ex :

```
1 function displayClosure () {
2   var count = 0;
3   return function () {
4     return count++;
5   };
6 }
7 var inc = displayClosure ();
8 inc (); // returns 0
9 inc (); // returns 1
10 inc (); // returns 2
```

Fonctions pré-définies

La fonction `eval`

La fonction **eval** exécute le code Javascript contenu dans une chaîne de caractères.

Exemple :

```
1  ...
2  <SCRIPT LANGUAGE="JavaScript">
3      function evaluation() {
4          document.formulaire.calcul.value=eval(document.formulaire.saisie.value); }
5  </SCRIPT>
6  ...
7  <FORM NAME="formulaire">
8      Saisissez une expression mathématique :
9      <INPUT TYPE="text" NAME=saisie MAXLENGTH=40 SIZE=40>
10     <INPUT TYPE="button" VALUE="évaluation." onClick="evaluation()">
11     <INPUT TYPE="text" NAME=calcul MAXLENGTH=40 SIZE=40>
12 </FORM>...
```

Fonctions pré-définies

Manipulations sur les types

Test :

- `isFinite(n)` : renvoie `true` si `n` est un nombre fini
- `isNaN(n)` : renvoie `true` si `n` n'est pas un nombre
- `typeof` : voir plus loin

Conversions :

- `parseFloat(s)` : converti une chaîne représentant un flottant en nombre,
- `parseInt(str, [rdx])` : converti une chaîne représentant un entier en nombre,

```
1 var numero="125";  
2 var nombre=parseFloat(numero); //retourne le nombre 125  
3  
4 var prix=30.75;  
5 var arrondi = parseInt(prix, 10); //retourne 30
```

Fonctions pré-définies

Manipulations sur les types (suite)

- `Number(o)` : convertit un objet en nombre,

```
1 var jour = new Date("December_17,_1995_03:24:00");//convertit la date en  
    millisecondes  
2 alert (Number(jour));
```

- `String(o)` : convertit un objet en chaîne,

```
1 jour = new Date(430054663215);//Convertit le nombre en date Mois jour,  
    Année etc.  
2 alert (String(jour));
```

- `encodeURIComponent(s)` : échape la chaîne `s` par rapport au langage des URI

```
1 var uri = "http://w3schools.com/my_test.php?name=stale&car=saab";  
2 var res = encodeURIComponent(uri);  
3 get("https://www.site.fr/ctrl?url="+uri);
```

res →

http%3A%2F%2Fw3schools.com%2Fmy%20test.php%3Fname%3D

Les Objets en Javascript

Introduction

Principe

Liaison

Le langage

Syntaxe

Variables

Les Opérateurs

Flot d'exécution

Fonctions

Objets

JSON

L'API D.O.M.

Motivations

Niveaux

DOM Core

DOM HTML

DOM et CSS

Exemples

Conformité

AJAX

Introduction

XMLHttpRequest

exemples

- Sont des entités soit prédéfinies du langage ou de l'environnement, soit créés par le programmeur.

- Ce sont des tableaux associatifs :

`objet.attribut ⇔ objet["attribut"]`

le `.` n'étant qu'un raccourci.

- Les objets (quant à eux) sont polymorphes⁴ :

```
1 var unRectangle = new Rectangle(20 , 10);           // Pas d'attribut couleur.  
2 unRectangle.couleur="Bleu";    // Dispose maintenant d'un attribut couleur.
```

- Il n'y a pas de classe, d'héritage, ou de polymorphisme.
- Les prototypes simulent ces fonctionnalités.

4. ou à schéma dynamique (pas de classe)

Les Objets en Javascript

- Un objet est un tableau associatif...
- ... donc un objet se parcourt comme un tableau simple :

```
1 var o = new ...; // Un objet
2 for (var a in o) {
3     window.console.log("attribut_" + a + "=" + o[a]);
4 }
```

- tout membre est une entrée du tableau (attributs + méthodes).

Les Objets en Javascript

Les méthodes

Introduction

Principe
Liaison

Le langage

Syntaxe
Variables
Les Opérateurs
Flot d'exécution
Fonctions
Objets
JSON

L'API D.O.M.

Motivations
Niveaux
DOM Core
DOM HTML
DOM et CSS
Exemples
Conformité

AJAX

Introduction
XMLHttpRequest
exemples

- Les méthodes sont des « fonctions globales ».

```
1  function iam() {  
2      alert("Je_suis_je" + this);  
3  }  
4  
5  var unRectangle = new Rectangle(20 , 10);  
6  
7  iam(); // Affiche "Je suis [object Window]"  
8  unRectangle.whoami=iam; // Ajoute dynamiquement la methode.  
9  unRectangle.whoami(); // Affiche "Je suis [object Object]"
```

- En fait, les fonctions sont des méthodes sur l'objet **Global** par défaut (dans un navigateur web, **Global** fait référence à `Window`).

Les Objets en Javascript

Les constructeurs

Introduction

Principe

Liaison

Le langage

Syntaxe

Variables

Les Opérateurs

Flot d'exécution

Fonctions

Objets

JSON

L'API D.O.M.

Motivations

Niveaux

DOM Core

DOM HTML

DOM et CSS

Exemples

Conformité

AJAX

Introduction

XMLHttpRequest

exemples

- Les constructeurs sont des fonctions dont l'appel est préfixé par `new` :

```
1 function Rectangle(lo , la) {  
2     this.longueur = lo;  
3     this.largeur = la;  
4 }  
5 var unRectangle = new Rectangle (20 , 10);
```

- Des classes d'objets peuvent être simulées initialement grâce aux « constructeurs ».

```
1 var r1 = new Rectangle (20 , 10);  
2 var r2 = new Rectangle (40 , 30);  
3 r1.color="red";      // r1 et r2 n'ont plus la meme  
4                      // structure a partir d'ici.  
5 alert(r2.color);     // Undefined.
```

- Les nouveaux attributs ne sont pas partagés.

Les Objets en Javascript

Les constructeurs (2)

Introduction

Principe

Liaison

Le langage

Syntaxe

Variables

Les Opérateurs

Flot d'exécution

Fonctions

Objets

JSON

L'API D.O.M.

Motivations

Niveaux

DOM Core

DOM HTML

DOM et CSS

Exemples

Conformité

AJAX

Introduction

XMLHttpRequest

exemples

- Il n'y a vraiment pas de différence entre constructeur et méthode :

```
1  function klass() {  
2      this.toto=5;  
3  }  
4  
5  k=new klass(); // Utilisation comme constructeur.  
6  
7  alert(k.toto); // 5  
8  
9  o=new Object(); // Objet vide.  
10 alert(o.toto); // Undefined  
11 o.k=klass;      // association en tant que methode.  
12 o.k();          // Appel en tant que methode  
13 alert(o.toto); // 5
```

Les Objets en Javascript

Les prototypes 1/3



Introduction

Principe

Liaison

Le langage

Syntaxe

Variables

Les Opérateurs

Flot d'exécution

Fonctions

Objets

JSON

L'API D.O.M.

Motivations

Niveaux

DOM Core

DOM HTML

DOM et CSS

Exemples

Conformité

AJAX

Introduction

XMLHttpRequest

exemples

Les propriétés d'un objet se divisent en deux catégories :

- les propriétés dites "propres", contenues dans l'objet lui-même,
- les propriétés de prototype.

En JavaScript, tout objet est associé à un autre objet, appelé **prototype**. Le prototype lui-même possède un prototype...

Les Objets en Javascript

Les prototypes 2/3

Introduction

Principe
Liaison

Le langage

Syntaxe
Variables
Les Opérateurs
Flot d'exécution
Fonctions
Objets
JSON

L'API D.O.M.

Motivations
Niveaux
DOM Core
DOM HTML
DOM et CSS
Exemples
Conformité

AJAX

Introduction
XMLHttpRequest
exemples

- Les prototypes permettent de définir (dynamiquement)⁵ des attributs partagés par plusieurs objets (c.-à-d. de simuler un membre statique).
- Le prototype est accessible à travers la propriété `prototype` du constructeur :

```
1  function Rectangle(lo , la) {  
2      this.longueur = lo;  
3      this.largeur = la;  
4  }  
5  var unRectangle = new Rectangle (20 , 10);  
6  alert(unRectangle.couleur);           // undefined  
7  Rectangle.prototype.couleur = "rouge"; // Ajout d'une propriété au  
   prototype.  
8  alert(unRectangle.couleur);^^I        // "rouge"
```

Les Objets en Javascript

Les prototypes 3/3

Introduction

Principe

Liaison

Le langage

Syntaxe

Variables

Les Opérateurs

Flot d'exécution

Fonctions

Objets

JSON

L'API D.O.M.

Motivations

Niveaux

DOM Core

DOM HTML

DOM et CSS

Exemples

Conformité

AJAX

Introduction

XMLHttpRequest

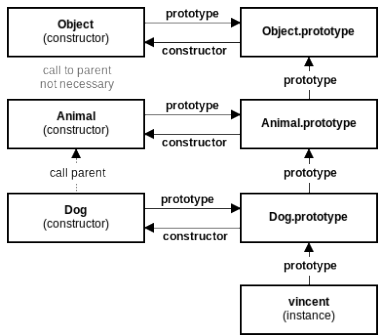
exemples

- les prototypes constituent des chaînes d'objets.
- l'accès à une propriété (`x.prop`) commence par l'examen des propriétés propres, puis celles du prototype, puis celles du prototype du prototype, etc.
- Les attributs de prototypes sont surchargeables :

```
1  function Rectangle(lo , la) {  
2      this.longueur = lo;  
3      this.largeur = la;  
4  }  
5  var r1 = new Rectangle (20 , 10);  
6  var r2 = new Rectangle (33 , 22);  
7  Rectangle.prototype.couleur = "rouge";  
8  alert(r1.couleur);           // "rouge"  
9  alert(r2.couleur);           // "rouge"  
10 r2.couleur="green";  
11 alert(r1.couleur);           // "rouge"  
12 alert(r2.couleur);           // "green"
```

L'héritage en Javascript

Si si, même sans classes !



```
1 function Animal(age) { /* ... */ } // Classe de base.
2 Animal.prototype.age = 1;
3
4 function Dog(name) { /* ... */ } // Classe derivee.
5 Dog.prototype = new Animal();
6
7 var v = new Dog("Vincent");
8 alert(v.age); // v->prototype->prototype->age
```

Une bonne nouvelle ?

EcmaScript 6

Introduction

Principe

Liaison

Le langage

Syntaxe

Variables

Les Opérateurs

Flot d'exécution

Fonctions

Objets

JSON

L'API D.O.M.

Motivations

Niveaux

DOM Core

DOM HTML

DOM et CSS

Exemples

Conformité

AJAX

Introduction

XMLHttpRequest

exemples

- EcmaScript 6 introduit une syntaxe plus conventionnelle :

```
1  class Carre extends Polygone {  
2      constructor(longueurCote) {  
3          super(longueurCote, longueurCote);  
4      }  
5      get aire() {  
6          return this.hauteur * this.largeur;  
7      }  
8      set longueurCote(nouvelleLongueur) {  
9          this.hauteur = nouvelleLongueur;  
10         this.largeur = nouvelleLongueur;  
11     }  
12 }  
13  
14 var carre = new Carre(2);
```

- Mais ça n'est que du sucre syntaxique,
- et le support est partiel.

Les Objets en Javascript

L'opérateur `typeof`

Introduction

Principe

Liaison

Le langage

Syntaxe

Variables

Les Opérateurs

Flot d'exécution

Fonctions

Objets

JSON

L'API D.O.M.

Motivations

Niveaux

DOM Core

DOM HTML

DOM et CSS

Exemples

Conformité

AJAX

Introduction

XMLHttpRequest

exemples

- L'opérateur `typeof` renvoie une chaîne de caractères indiquant le type de l'opérande.

```
1 var i = 1;
2 typeof i;           // retourne number
3 var titre="Les_raisins_de_la_colere";
4 typeof titre;       // retourne String
5 var jour = new Date();
6 typeof jour;        // retourne Object
7 var choix = true;
8 typeof choix;       // retourne Boolean
9 var cas = null;
10 typeof cas;         // retourne Object
11 typeof alert;       // retourne function
12 typeof Math;        // retourne Object
```


Objets prédéfinis

- L'objet Global définit les propriétés et méthodes communes à tous les objets :
 - Les méthodes et propriétés de cet objet n'appartiennent à aucune classe et cet objet n'a pas de nom.
 - La seule façon de faire référence à cet objet est `this`.
 - Chaque variable ou fonction globale est propriété de Global
 - Propriétés de Global : `Infinity` `NaN` `undefined`
 - Quelques méthodes : `parseFloat(s)`, `parseInt(s, base)`, `isNaN(expression)`, `eval(expression)`, `escape(URL)` et `unescape(URL)` (Codage des URL).
- Classes prédéfinies : `Array`, `Boolean`, `Date`, `Function`, `Math` `Number`, `Image`, `Option`, `RegExp`, `String`, `Navigator`, `Window`, `Screen`

Le format JSON

- En JavaScript, il est possible de construire des objets en les initialisant directement :

```
1 var MonObjet = {  
2   nom : "menu",  
3   id : 10,  
4   visible : true  
5 }
```

est équivalent à

```
1 var MonObjet = new Object() ;  
2 MonObjet.nom = "menu";  
3 MonObjet.id = 10;  
4 MonObjet.visible = true ;  
5 }
```


ou encore

```
1 function TheThing() {  
2   this.nom = "menu";  
3   this.id = 10;  
4   this.visible = true ;  
5 }  
6 var MonObjet = new TheThing() ;
```

Le format JSON

- Cette notation est récursive :

```
1  var MonObjet = {  
2      nom : "menu",           // Chaîne  
3      ip : [192, 168, 1, 10],  // Tableau de nombres  
4      dims : {                // (sous)Objet  
5          w : 50,  
6          h : 200  
7      },  
8      display : function () { // Fonction  
9          alert(this.nom);  
10     }  
11 }
```

- Cette notation a donné lieu à un format d'échange de donnée : **JSON** (*JavaScript Object Notation*).
- Elle est standardisée par l'IETF (RFC 4627 ).
- Le type MIME pour un flux texte en JSON est "application/json".

Le format JSON

- Elle concurrence XML pour des échanges de données simples ou à destination de programmes écrits en JavaScript.
- En effet, son utilisation est très efficace en conjonction avec `eval()` :

```
1 var jsonString = '{_host:_ "pipo",_ip:_ [192,_ 168,_ 1,_ 10]_}' ;  
2  
3 eval("var _hostData_ = " + jsonString) ;  
4  
5 alert (hostData . host) ;  
6 ...
```

- Toutefois, son utilisation directe avec `eval()` est une mauvaise idée ! (voir plus loin).

Introduction

Principe
Liaison

Le langage

Syntaxe
Variables
Les Opérateurs
Flot d'exécution
Fonctions
Objets
JSON

L'API D.O.M.

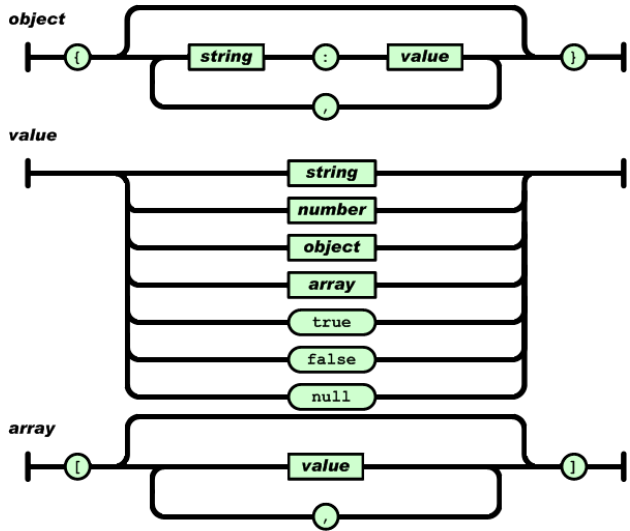
Motivations
Niveaux
DOM Core
DOM HTML
DOM et CSS
Exemples
Conformité

AJAX

Introduction
XMLHttpRequest
exemples

Le format JSON

Syntaxe (c.f. json.org)



Le format JSON


Sécurité

- Bien que JSON n'inclue pas le type `Function`, rien n'empêche d'injecter du code JavaScript dans un flux JSON :

```
1 var fluxJson= '{\n
2   uuyy:uMath.log(10),\n
3   uuX:uwindow.setTimeout(\n
4     uuuuufunction(){alert("Code_malicieux!");},u100)\n
5   }';
6
7 eval("var_utoto=u" + fluxJson);
8 ...
```

- Sans danger si le flux est parsé par le programme cible.
- Très risqué s'il est évalué directement dans une machine virtuelle JavaScript.



On utilisera systématiquement `JSON.parse()`  et non `eval()` !

Introduction à D.O.M.

Introduction

Principe

Liaison

Le langage

Syntaxe

Variables

Les Opérateurs

Flot d'exécution

Fonctions

Objets

JSON

L'API D.O.M.

Motivations

Niveaux

DOM Core

DOM HTML

DOM et CSS

Exemples

Conformité

AJaX

Introduction

XMLHttpRequest

exemples

- Javascript permet d'ajouter du dynamisme aux documents web,
- i.e. Javascript peut opérer dynamiquement des modifications sur le contenu du document (ex : affichage de *div* cachées, application d'un « template », validation d'un formulaire avant envoi, etc.)
- Pour cela il utilise **la programmation évènementielle** et une interface de programmation à base d'objets.

Introduction à D.O.M.

Pourquoi D.O.M. ?

Introduction

Principe

Liaison

Le langage

Syntaxe

Variables

Les Opérateurs

Flot d'exécution

Fonctions

Objets

JSON

L'API D.O.M.

Motivations

Niveaux

DOM Core

DOM HTML

DOM et CSS

Exemples

Conformité

AJAX

Introduction

XMLHttpRequest

exemples

- Les applications web d'aujourd'hui ont besoin d'être indépendante de la plateforme et du navigateur (*user-agent*).
- Le contenu dynamique à base de scripts fait parti de l'application : il permet de décharger le serveur de certaines tâches d'interaction utilisateur.
- Il est cependant inconcevable de devoir produire de tels programmes en tenant compte des différences d'environnement (portabilité de l'application)...

Introduction à D.O.M.

Pourquoi D.O.M. ?

Introduction

Principe
Liaison

Le langage

Syntaxe
Variables
Les Opérateurs
Flot d'exécution
Fonctions
Objets
JSON

L'API D.O.M.

Motivations
Niveaux
DOM Core
DOM HTML
DOM et CSS
Exemples
Conformité

AJAX

Introduction
XMLHttpRequest
exemples

Autrefois :

- Netscape :

```
1 document.layers['nom'].attribut // ou bien
2 document.layers.nom.attribut
```

- I.E. :

```
1 document.all['nom'].attribut
```

- Du coup il est nécessaire de connaître le navigateur (ou la Machine virtuelle) exécutant le code pour l'adapter.

Introduction à D.O.M.

Où suis-je ?

Introduction

Principe

Liaison

Le langage

Syntaxe

Variables

Les Opérateurs

Flot d'exécution

Fonctions

Objets

JSON

L'API D.O.M.

Motivations

Niveaux

DOM Core

DOM HTML

DOM et CSS

Exemples

Conformité

AJaX

Introduction

XMLHttpRequest

exemples

- Exemple de détection de navigateur :

```
navigator.userAgent = Mozilla/5.0 (X11;  
Linux i686; rv:7.0.1) Gecko/20100101  
Firefox/7.0.1
```

```
1 var detect = navigator.userAgent.toLowerCase() ;  
2  
3 if (detect.indexOf('konqueror')+1) { ... }  
4 else if (detect.indexOf('safari')+1) { ... }  
5 else if (detect.indexOf('omniweb')+1) { ... }  
6 else if (detect.indexOf('opera')+1) { ... }  
7 else if (detect.indexOf('webtv')+1) { ... }  
8 else if (detect.indexOf('icab')+1) { ... }  
9 else if (detect.indexOf('msie')+1) { ... }  
10 else ...
```

- Identifie une compatibilité par rapport à un ensemble de fonctionnalités requises,
- Mais, peu fiable et lourd...

Introduction à D.O.M.

Où suis-je ?

Introduction

Principe
Liaison

Le langage

Syntaxe
Variables
Les Opérateurs
Flot d'exécution
Fonctions
Objets
JSON

L'API D.O.M.

Motivations
Niveaux
DOM Core
DOM HTML
DOM et CSS
Exemples
Conformité

AJAX

Introduction
XMLHttpRequest
exemples

- Autre méthode, le **test d'objet** :

```
1  if (document.layers.nom.attribut) { ...}  
2  else if (document.all['nom'].attribut) {...}  
3  
4  if (window.focus) { // Attention different de window.focus() !  
5  ...
```

- Javascript permet de tester l'existence d'une propriété d'un objet.
- Permet donc de savoir si une fonctionnalité est disponible quelque soit le navigateur.

→ Un modèle objet standardisé permet de réduire le nombre de cas où ce code est nécessaire...

Qu'est-ce que D.O.M.

- **D.O.M.** (pour **Document Object Model** est une recommandation du W3C⁶
« *The Document Object Model is a platform-and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents.* »
- Le document est chargé (en général intégralement) en mémoire sous forme de modèle arborescent,
- l'interface DOM donne accès à ce modèle depuis n'importe quel langage : javascript, Java (`org.w3c.dom.*`), PhpDom, ...
- L'interface consiste en une collection de types, d'objets, de fonctions (méthodes) et d'attributs.

6. <http://www.w3.org/DOM/>

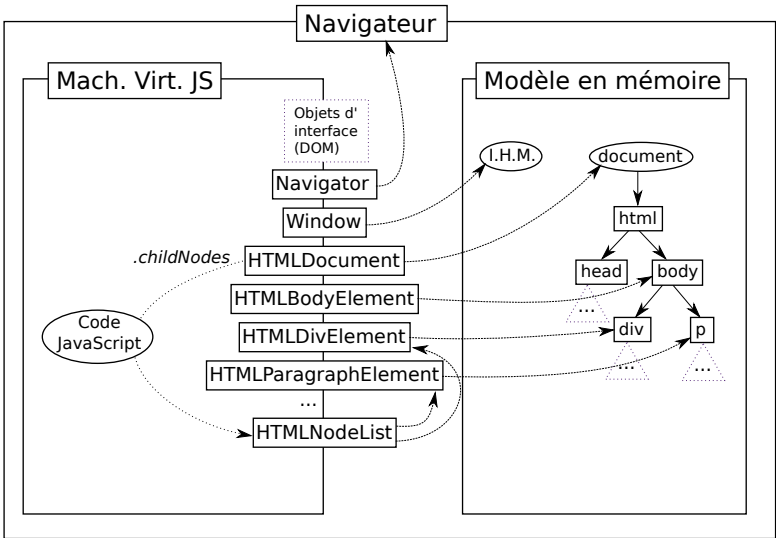
Un premier exemple

```
var balises_a = document.getElementsByTagName("a");  
for(var i = 0; i < balises_a.length; i++) {  
    alert("Href of " + i + "-th element is : " +  
        balises_a[i].href + "\n");  
}
```

- `document` : objet global représentant le document,
- `getElementsByTagName("a")` : retourne une liste de noeuds (objets) de tous les éléments `<a> . . . `
- `liste_balises_a.length` : la longueur de la liste
- `liste_balises_a[i].href` : propriété de l'interface `HTMLAnchorElement`, reflète la valeur de l'attribut "href"
- `alert()` : méthode DOM niveau 0 pour créer une fenêtre popup

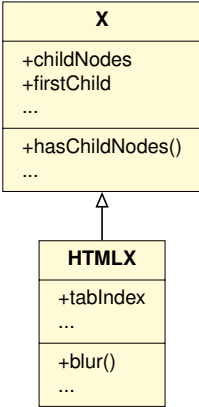
Principe de DOM

Une API d'interfaces



Principe de DOM

Rapport entre DOM et le DOM HTML



- DOM est un noyau générique basé sur XML (indépendant de toute application)
- HTML-DOM est un sur-ensemble de DOM (spécifique à l'« application » HTML)

Niveaux et familles DOM

Les spécifications W3C DOM sont organisés en "niveaux"

« **Level 0** » : implémentations variées non standardisés
basées sur le DOM de Netscape 4.x, le DHTML de MS, etc.

Level 1 (octobre 1998)

- DOM Level 1 Core : navigation et manipulation de documents HTML et XML,
- DOM Level 1 HTML : extensions spécifiques à HTML.

...

Niveaux et familles DOM (suite)

... **Level 2 (2001)**, XML namespace support, filtered views and events :

- DOM Level 2 Core Specification (étend DOM Level 1 Core)
- DOM Level 2 Views Specification :
- DOM Level 2 Events Specification : gestion d'événements (mais pas du clavier)
- DOM Level 2 Style Specification : lecture et modification du CSS
- DOM Level 2 Traversal and Range Specification
- DOM Level 2 HTML Specification : extensions pour HTML

...

Niveaux et familles DOM (suite)

Introduction

Principe

Liaison

Le langage

Syntaxe

Variables

Les Opérateurs

Flot d'exécution

Fonctions

Objets

JSON

L'API D.O.M.

Motivations

Niveaux

DOM Core

DOM HTML

DOM et CSS

Exemples

Conformité

AJAX

Introduction

XMLHttpRequest

exemples

...

Level 3 (2004) : contient 6 spécifications

- DOM Level 3 Core
- DOM Level 3 Load and Save
- DOM Level 3 XPath
- DOM Level 3 Views and Formatting
- DOM Level 3 Requirements

Note : le support des différents niveaux et familles varie suivant les navigateurs. Pour tester :

<http://www.w3.org/2003/02/06-dom-support.html>

DOM Core

Types de noeuds

Document → Element (un max), ProcessingInstruction, Comment, DocumentType (un max)

DocumentFragment → Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference

DocumentType → aucun enfant

EntityReference → Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference

Element → Element, Text, Comment, ProcessingInstruction, CDATASection, EntityReference

Attr → Text, EntityReference

ProcessingInstruction → aucun enfant

Comment → aucun enfant

Text → aucun enfant

CDATASection → aucun enfant

Entity → Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference

Notation → aucun enfant

DOM Core

Types de données

Introduction

Principe
Liaison

Le langage

Syntaxe
Variables
Les Opérateurs
Flot d'exécution
Fonctions
Objets
JSON

L'API D.O.M.

Motivations
Niveaux
DOM Core
DOM HTML
DOM et CSS
Exemples
Conformité

AJAX

Introduction
XMLHttpRequest
exemples

Node : un noeud d'un arbre DOM (e.g. document, element, attributes, etc. et qui héritent ses méthodes et propriétés)

Document : un document HTML, XHTML et XML (implémente l'interface "Document".)

Element : un élément (représentation d'une balise et de son contenu), hérite de Node (interface pour les nœuds).

NodeList : liste ordonnée de nœuds telle que renvoyée par document.getElementsByTagName(). Les éléments d'une nodeList sont accessibles par un index de deux manières différentes (mais identiques quand au résultat) :
list.item(1) list[1]

Attribute : représente un attribut d'un élément.

NamedNodeMap : comme NodeList mais donne accès aux éléments à la fois par leur nom ou par leur index.

L'objet Node

Attributs

- baseURI** : the base URI of a node
- childNodes** : the NodeList of child nodes for a node
- firstChild** : the first child of a node
- lastChild** : the last child of a node
- localName** : the local part of the name of a node
- namespaceURI** : the namespace URI of a node
- nextSibling** : the node immediately following a node
- nodeName** : the name of a node, depending on its type
- nodeType** : the type of a node
- nodeValue** : sets or returns the value of a node, depending on its type
- ownerDocument** : the root element (document object) for a node
- parentNode** : the parent node of a node
- prefix** : Sets or returns the namespace prefix of a node
- previousSibling** : Returns the node immediately before a node
- textContent** : Sets or returns the textual content of a node and its descendants

L'objet DOM–Node

Quelques méthodes

Introduction

Principe
Liaison

Le langage

Syntaxe
Variables
Les Opérateurs
Flot d'exécution
Fonctions
Objets
JSON

L'API D.O.M.

Motivations
Niveaux
DOM Core
DOM HTML
DOM et CSS
Exemples
Conformité

AJAX

Introduction
XMLHttpRequest
exemples

appendChild() : adds a node to the end of the list of children of a node

cloneNode() : clones a node

compareDocumentPosition() : compares the document position of two nodes

hasAttributes() : true if a node has any attributes

hasChildNodes() : true if a node has any child nodes

insertBefore(new,ref) : inserts a node before an existing child node

isEqualNode(n) : checks if two nodes are equal

isSameNode(n) : checks if two nodes are the same node

removeChild(n) : removes a child node

replaceChild(new,old) : replaces a child node

setUserData(key,data,handler) : Associates an object to a key on a node

L'objet DOM–Element

Ajoute deux propriétés à Node :

attributes : a NamedNodeMap of attributes for the element

tagName : the name (markup) of the element

Et quelques méthodes associées :

getAttribute() : the value of an attribute

getAttributeNode() : returns an attribute node as an
Attribute object

getElementsByTagName() : the NodeList of matching
element nodes, and their children

removeAttribute() : removes a specified attribute

removeAttributeNode() : removes a specified attribute
node

setAttribute() : adds a new attribute

setAttributeNode() : adds a new attribute node

DOM–HTML

Introduction

Principe
Liaison

Le langage

Syntaxe
Variables
Les Opérateurs
Flot d'exécution
Fonctions
Objets
JSON

L'API D.O.M.

Motivations
Niveaux
DOM Core
DOM HTML
DOM et CSS
Exemples
Conformité

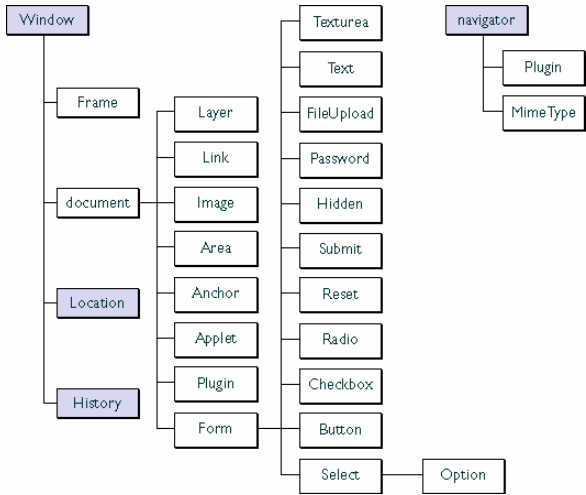
AJAX

Introduction
XMLHttpRequest
exemples

- En web, on utilise DOM–HTML + DOM core (AJaX).
- DOM–HTML est une spécialisation du noyau DOM (core DOM) de XML vers HTML.
- Consiste en l'ajout de nouveaux attributs et méthodes sur les objets existants et l'ajout de nouveaux objets :
DOM–{Elements, Anchor, Area, Base, Body, Button, Form, Frame/IFrame, Frameset, Image, Input Button, Input Checkbox, Input File, Input Hidden, Input Password, Input Radio, Input Reset, Input Submit, Input Text, Link, Meta, Object, Option, Select, Style, Table, td, th, tr, Textarea}

DOM-HTML

Organisation des objets



L'objet HTML–Element

Quelques attributs

Introduction

Principe
Liaison

Le langage

Syntaxe
Variables
Les Opérateurs
Flot d'exécution
Fonctions
Objets
JSON

L'API D.O.M.

Motivations
Niveaux
DOM Core
DOM HTML
DOM et CSS
Exemples
Conformité

AJAX

Introduction
XMLHttpRequest
exemples

En plus des attributs de DOM–Element :

accessKey : sets or returns an accesskey for an element

className : sets or returns the class attribute of an element

clientHeight,clientWidth : viewable height/width of the content on a page

dir : sets or returns the text direction of an element

disabled : sets or returns whether an element is disabled, or not

id : sets or returns the id of an element

innerHTML : sets or returns the HTML contents (+text) of an element

lang : sets or returns the language code for an element

style : sets or returns the style attribute of an element (Voir plus loin...)

tabIndex : sets or returns the tab order of an element

title : sets or returns the title attribute of an element

L'objet HTML–Element

Quelques méthodes

Introduction

Principe

Liaison

Le langage

Syntaxe

Variables

Les Opérateurs

Flot d'exécution

Fonctions

Objets

JSON

L'API D.O.M.

Motivations

Niveaux

DOM Core

DOM HTML

DOM et CSS

Exemples

Conformité

AJAX

Introduction

XMLHttpRequest

exemples

Toujours en plus de celles de DOM–Element :

blur() : removes focus from an element

click() : executes a click on an element

focus() : gives focus to an element

item() : returns an element based on its index within the document tree

toString() : converts an element to a string

L'objet HTML–Document

Propriétés 1/2

Introduction

Principe

Liaison

Le langage

Syntaxe

Variables

Les Opérateurs

Flot d'exécution

Fonctions

Objets

JSON

L'API D.O.M.

Motivations

Niveaux

DOM Core

DOM HTML

DOM et CSS

Exemples

Conformité

AJAX

Introduction

XMLHttpRequest

exemples

- Collections donnant un accès direct à des éléments du document (par indice ou par nom) : **anchors[]** : an array of all the anchors in the document
forms[] : an array of all the forms in the document
images[] : an array of all the images in the document
links[] : an array of all the links in the document

L'objet HTML–Document

Propriétés 2/2

Introduction

Principe

Liaison

Le langage

Syntaxe

Variables

Les Opérateurs

Flot d'exécution

Fonctions

Objets

JSON

L'API D.O.M.

Motivations

Niveaux

DOM Core

DOM HTML

DOM et CSS

Exemples

Conformité

AJAX

Introduction

XMLHttpRequest

exemples

- Autres propriétés : **cookie** : all name/value pairs of cookies in the document
documentMode : the rendering mode used by the browser
domain : the domain name of the server that loaded the document
lastModified : the date and time the document was last modified
readyState : the (loading) status of the document
referrer : the URL of the document that loaded the current document
title : sets or returns the title of the document
URL : the full URL of the document

L'objet HTML–Document

Quelques méthodes

Introduction

Principe

Liaison

Le langage

Syntaxe

Variables

Les Opérateurs

Flot d'exécution

Fonctions

Objets

JSON

L'API D.O.M.

Motivations

Niveaux

DOM Core

DOM HTML

DOM et CSS

Exemples

Conformité

AJAX

Introduction

XMLHttpRequest

exemples

close() : closes the output stream previously opened with `document.open()`

getElementById() : accesses the first element with the specified id

getElementsByTagName() : accesses all elements with a specified name

getElementsByTagName() : accesses all elements with a specified tagname

open() : opens an output stream to collect the output from `document.write()` or `document.writeln()`

write() : writes HTML expressions or JavaScript code to a document

writeln() : same as `write()`, but adds a newline character after each statement

L'objet HTML–Document

Écriture dans le flux du document

Introduction

Principe
Liaison

Le langage

Syntaxe
Variables
Les Opérateurs
Flot d'exécution
Fonctions
Objets
JSON

L'API D.O.M.

Motivations
Niveaux
DOM Core
DOM HTML
DOM et CSS
Exemples
Conformité

AJaX

Introduction
XMLHttpRequest
exemples

```
1 <body>
2   <h1>Horloge webante !</h1>
3
4   <script type="text/javascript">
5     //<!--
6     var date=new Date() ;
7     document.writeln(' Nous sommes le <strong>: </strong><br>' );
8     //-->
9   </script>
10
11   <h1>Bonne journee ... </h1>
12 </body>
```

Résultat :
Horloge webante !

Nous sommes le : **25/10/2011**

Bonne journée...

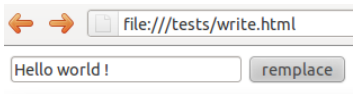
L'objet HTML–Document

Ré-écriture du document

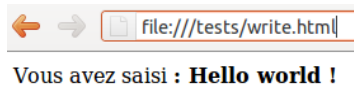
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

```
...  
<script type="text/javascript"> //<!--  
function go() {  
    var saisie=document.getElementById("userinput").value ;  
    document.open() ; // Implicite. Attention, vide l'arbre !  
    document.writeln('Vous avez saisi<strong>:'+ saisie+'</strong><br>') ;  
    document.close() // Pas implicite !  
}  
//--> </script> </head>  
  
<body>  
    <form>  
        <input type="text" id="userinput">  
        <input type="button" onclick="go();" value="replace">  
    </form>  
</body>
```

Avant :



Après



L'objet HTML–Form

- Javascript peut être utilisé pour faire de la **pré**–validation de formulaire.
- i.e. bloquer l'envoi d'un formulaire tant que toutes les saisies ne sont pas correctes.
- L'objet **Form** donne accès à toutes les éléments et événements du formulaire :

Collections de l'objet HTML–Form :
elements[] : an array of all elements in a form

L'objet HTML-Form

Propriétés

Introduction

Principe

Liaison

Le langage

Syntaxe

Variables

Les Opérateurs

Flot d'exécution

Fonctions

Objets

JSON

L'API D.O.M.

Motivations

Niveaux

DOM Core

DOM HTML

DOM et CSS

Exemples

Conformité

AJAX

Introduction

XMLHttpRequest

exemples

Propriétés de l'objet HTML-Form :

acceptCharset : sets or returns the value of the accept-charset attribute

action : sets or returns the value of the action attribute

enctype : sets or returns the value of the enctype attribute

length : returns the number of elements in the form

method : sets or returns the value of the method attribute

name : sets or returns the value of the name attribute of the form

target : sets or returns the value of the target attribute

L'objet HTML–Form

Méthodes et événements

Introduction

Principe

Liaison

Le langage

Syntaxe

Variables

Les Opérateurs

Flot d'exécution

Fonctions

Objets

JSON

L'API D.O.M.

Motivations

Niveaux

DOM Core

DOM HTML

DOM et CSS

Exemples

Conformité

AJAX

Introduction

XMLHttpRequest

exemples

Méthodes de l'objet HTML–Form :

reset() : resets a form

submit() : submits a form

Évènements de l'objet HTML–Form :

onreset : function called when the reset button is clicked

onsubmit : the submit button is clicked

Accès aux styles

Introduction

Principe
Liaison

Le langage

Syntaxe
Variables
Les Opérateurs
Flot d'exécution
Fonctions
Objets
JSON

L'API D.O.M.

Motivations
Niveaux
DOM Core
DOM HTML
DOM et CSS
Exemples
Conformité

AJAX

Introduction
XMLHttpRequest
exemples

- Il est possible de modifier le style *"inline"* d'un élément à partir de javascript
- grâce à l'attribut **style** des éléments. Syntaxe :

```
1 document.getElementById("id").style.property="value";
```

- Chaque propriété de l'objet **Style** correspond à une propriété CSS
- Règle de nommage : enlever le "-" et capitaliser les mots (sauf le premier).
- Exemple : "background-color" → "backgroundColor"

Accès aux styles

Exemples de propriétés de style

DOM CSS	Propriété CSS
background	background
backgroundColor	background-color
borderColor	border-color
fontFamily	font-family
fontSize	font-size
marginBottom	margin-bottom
marginLeft	margin-left

Exemple 1

Ajout d'un nouveau nud

Ceci est un premier paragraphe

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>une page simple</title>
5 </head>
6 <body>
7 <p>Ceci est un premier paragraphe</p>
8 </body>
9 </html>
```

```
1 var newP=document.createElement("p");
2 document.getElementsByTagName("body").item(0).appendChild(newP);
```

```
1 var newText=document.createTextNode("Ceci est un nouveau paragraphe!");
2 newP.appendChild(newText);
```

Exemple 1

Ajout d'un nouveau nud

Ceci est un premier paragraphe

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>une page simple</title>
5 </head>
6 <body>
7 <p>Ceci est un premier paragraphe</p>
8 </body>
9 </html>
```

```
1 var newP=document.createElement("p");
2 document.getElementsByTagName("body").item(0).appendChild(newP);
```

Rien n'apparaît car le paragraphe est vide. Comment ajouter du contenu ?

```
1 var newText=document.createTextNode("Ceci est un nouveau paragraphe!");
2 newP.appendChild(newText);
```

Exemple 1

Ajout d'un nouveau nud

Ceci est un premier paragraphe

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>une page simple</title>
5 </head>
6 <body>
7 <p>Ceci est un premier paragraphe</p>
8 </body>
9 </html>
```

```
1 var newP=document.createElement("p");
2 document.getElementsByTagName("body").item(0).appendChild(newP);
```

```
1 var newText=document.createTextNode("Ceci est un nouveau paragraphe!");
2 newP.appendChild(newText);
```

Ceci est un premier paragraphe

Ceci est un nouveau paragraphe !

Exemple 2

Déplacement d'un sous-arbre

Introduction

Principe
Liaison

Le langage

Syntaxe
Variables
Les Opérateurs
Flot d'exécution
Fonctions
Objets
JSON

L'API D.O.M.

Motivations
Niveaux
DOM Core
DOM HTML
DOM et CSS
Exemples
Conformité

AJAX

Introduction
XMLHttpRequest
exemples

Ceci est un premier paragraphe

Ceci est un nouveau paragraphe !

```
1 var body=document . getElementsByTagName ( "body" ) . item ( 0 ) ;
2 var list=body . getElementsByTagName ( "p" ) ;
3   alert ( list+ ":" +list . length ) ;      // ?
4 var first=list . item ( 0 ) ;
```

```
1 body . removeChild ( list . item ( 0 ) ) ;
2   alert ( list+ ":" +list . length ) ; //
   ?
```

```
1 body . appendChild ( first ) ;
2   alert ( list+ ":" +list . length ) ; //
   ?
```

Exemple 2

Déplacement d'un sous-arbre

Introduction

Principe
Liaison

Le langage

Syntaxe
Variables
Les Opérateurs
Flot d'exécution
Fonctions
Objets
JSON

L'API D.O.M.

Motivations
Niveaux
DOM Core
DOM HTML
DOM et CSS
Exemples
Conformité

AJAX

Introduction
XMLHttpRequest
exemples

Ceci est un premier paragraphe

Ceci est un nouveau paragraphe !

```
1 var body=document . getElementsByTagName ( "body" ) . item ( 0 ) ;  
2 var list=body . getElementsByTagName ( "p" ) ;  
3   alert ( list+" : "+list . length ) ;      // ?  
4 var first=list . item ( 0 ) ;
```

```
1 body . removeChild ( list . item ( 0 ) ) ;  
2   alert ( list+" : "+list . length ) ; //  
   ?
```

Ceci est un nouveau paragraphe !

```
1 body . appendChild ( first ) ;  
2   alert ( list+" : "+list . length ) ; //  
   ?
```

Exemple 2

Déplacement d'un sous-arbre

Introduction

Principe
Liaison

Le langage

Syntaxe
Variables
Les Opérateurs
Flot d'exécution
Fonctions
Objets
JSON

L'API D.O.M.

Motivations
Niveaux
DOM Core
DOM HTML
DOM et CSS
Exemples
Conformité

AJAX

Introduction
XMLHttpRequest
exemples

Ceci est un premier paragraphe

Ceci est un nouveau paragraphe !

```
1 var body=document . getElementsByTagName ( "body" ) . item ( 0 ) ;  
2 var list=body . getElementsByTagName ( "p" ) ;  
3   alert ( list+ ":" +list . length ) ;      // ?  
4 var first=list . item ( 0 ) ;
```

```
1 body . removeChild ( list . item ( 0 ) ) ;  
2   alert ( list+ ":" +list . length ) ;      //  
   ?
```

Ceci est un nouveau paragraphe !

```
1 body . appendChild ( first ) ;  
2   alert ( list+ ":" +list . length ) ;      //  
   ?
```

Ceci est un nouveau paragraphe !

Ceci est un premier paragraphe

Exemple 2

Comportement des listes DOM

```
1 var body=document.getElementsByTagName( "body" ). item (0) ;
2 var list=body.getElementsByTagName( "p" ) ;
3 alert( list+":"+list.length) ; // 2
4 var first=list.item(0) ;
5 body.removeChild( list.item(0) ) ; // Pas une operation sur la liste!
6 alert( list+":"+list.length) ; // 1
7 body.appendChild( first ) ; // Pas une operation sur la liste!
8 alert( list+":"+list.length) ; // 2
```


- D'une manière générale avec DOM, les listes (comme `HTMLCollection` ou `NodeList`) restent connectée à la requête et aux objets qui ont servi à les produire.
- Toute altération de ces objets provoque la mise à jour de la liste.
- Attention aux effets de bord :

```
1 var list=document.getElementsByClassName( "menu" ) ;
2 for( var i=0; i<list.length; ++i) {
3   ^^list.item(i).setAttribute( "class", "autrechose" ) ;
4 }
```

Portabilité

... et conformité des navigateurs

- DOM doit concourir à améliorer la portabilité des applications,
- cela repose cependant sur la conformité des navigateurs :

**JsUnit 2.0beta TestRunner**
Running on Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Ubuntu Chromium/45.0.2454.

Enter the filename of the Test Page to be run:

http://

Trace level: ☒ Close old trace window on new run Page load timeout:

Status: Done (90.059 seconds)

Progress:

Runs: 238 **Errors:** 12 **Failures:** 32

Errors and failures:

```
http://www.w3.org/2004/04/ecmascript/level1/core/documentinvalidcharacterexceptioncreateentref.html:documentinvalidc
http://www.w3.org/2004/04/ecmascript/level1/core/documentinvalidcharacterexceptioncreateentref1.html:documentinvalidc
http://www.w3.org/2004/04/ecmascript/level1/core/documentinvalidcharacterexceptioncreatepi.html:documentinvalidchara
http://www.w3.org/2004/04/ecmascript/level1/core/documentinvalidcharacterexceptioncreatepi1.html:documentinvalidchar
http://www.w3.org/2004/04/ecmascript/level1/core/hc_characterdataindexsizeerrdeletedatacountnegative.html:hc_charact
```

7. Résultats identiques entre Firefox et Chrome !

Qu'est ce qu'*AJaX*

Introduction

Principe

Liaison

Le langage

Syntaxe

Variables

Les Opérateurs

Flot d'exécution

Fonctions

Objets

JSON

L'API D.O.M.

Motivations

Niveaux

DOM Core

DOM HTML

DOM et CSS

Exemples

Conformité

AJaX

Introduction

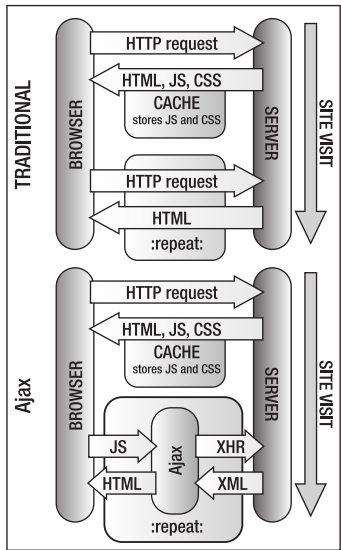
XMLHttpRequest

exemples

- **AJaX** signifie **A**synchronous **J**avascript and **X**ML.
- Terme introduit en février 2005 par Jesse James Garrett⁸ pour décrire une méthodologie de développement reposant sur des langages et technologies courants :
 - HTML/XHTML et CSS pour la mise en forme,
 - JavaScript/DOM l'interaction utilisateur dynamiquement avec l'information présentée,
 - XML, XSLT (rare) pour l'échange de données et leur transformations
 - et surtout l'objet `XMLHttpRequest` pour la communication asynchrone avec le serveur.

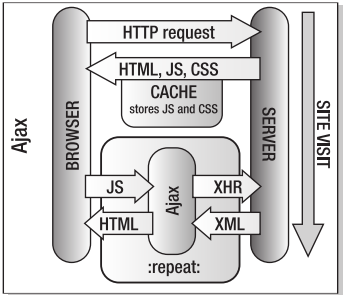
8. www.adaptivepath.com/ideas/ajax-new-approach-web-applications

Principe d'AjAX



- Dans une application classique, il faut recharger la page intégralement pour mettre à jour la *vue* → transport de l'état, trafic inutile.
- En AJAX, l'objet *XMLHttpRequest* permet d'effectuer des requêtes HTTP depuis le code Javascript de la page chargée par le client.
- ...

Principe d'AjAX



- La page (la vue) est chargée une seule fois.
- Le code javascript transmis gère les interactions avec l'utilisateur.
- Des nouvelles données sont lues à partir du serveur grâce à *XMLHttpRequest* autant de fois que nécessaire.
- Ces données sont incorporées dynamiquement à la page/vue par Javascript/DOM et mises en forme par HTML/CSS.
- L'interface est plus réactive et plus ergonomique.

Support d'AjAX

Introduction

Principe

Liaison

Le langage

Syntaxe

Variables

Les Opérateurs

Flot d'exécution

Fonctions

Objets

JSON

L'API D.O.M.

Motivations

Niveaux

DOM Core

DOM HTML

DOM et CSS

Exemples

Conformité

AjAX

Introduction

XMLHttpRequest

exemples

- AJAX est promu par l'**Open AJAX Initiative** initié par IBM et regroupant : 24SevenOffice, Adobe Systems, BEA Systems, Borland, the Dojo Foundation, Eclipse Foundation, Google, Ilog, Yahoo !, Laszlo Systems, Mozilla Corporation, Novell, Openwave Systems, SAP, Oracle, Red Hat, Tibco, Zend et Zimbra
- Le composant principal **XMLHttpRequest** est supporté par tous les navigateurs récents.
- C'est à l'origine un composant ActiveX introduit par Microsoft et proposé en 2006 pour standardisation par le W3C (www.w3.org/TR/XMLHttpRequest/)

Le composant *XMLHttpRequest*

- *XMLHttpRequest*, ou *XHR*,
- D'abord un ActiveX intégré par Microsoft aux navigateur IE 5 à 7 à partir de 1998,
- repris successivement sous Mozilla 1.0 (mai 2002), Safari 1.2 (février 2004), Konqueror 3.4 (mars 2005) puis Opera 8.0 (avril 2005), etc.
- Proposé pour standardisation au W3C en 2006 :
 - API en version 1 le 15 avril 2007,
 - API en version 2 le 25 février 2008.
 - Successeur prévu : *XMLHttpRequest Level 2* avec (peut-être) *Access control* pour accéder à des domaines différents ou *XDomainRequest* (proposé par MS).
- C'est un objet, Javascript est donc nécessaire.

Instanciación *XMLHttpRequest*

Code générique pour l'instanciation de XHR :

```
1 function createXhrObject() {
2     if (window.XMLHttpRequest)
3         return new XMLHttpRequest();
4
5     ^^ if (window.ActiveXObject) {
6         var names = [
7             "Msxml2.XMLHTTP.6.0",
8             "Msxml2.XMLHTTP.3.0",
9             "Msxml2.XMLHTTP",
10            "Microsoft.XMLHTTP"
11        ];
12        for (var i in names) {
13            try {
14                return new ActiveXObject(names[i]);
15            } catch (e) {}
16        }
17        return null; // not supported
18    }
19
20    // Instanciación :
21    xhr = createXhrObject();
22    if (!xhr) window.alert("Votre navigateur ne prend pas en charge l'objet
23        XMLHttpRequest.");
```

Contraintes / Sécurité

- Pas de « récursivité », le code Javascript rappatrié n'est pas exécuté,
- pas de requêtes sur des domaines croisés (*cross-domain*) : le site web adressé doit être celui ayant envoyé la page.
- Possible à l'avenir grâce à *Access control* ou *XDomainRequest(XDR)* : le site web adressé informe le navigateur qu'il autorise le site à l'origine de la requête. Ex :

```
1 <?php
2     header("Access-Control-Allow-Origin:␣*"); // Tous les
        domaines
3     header("Access-Control-Allow-Origin:␣monsite.com"); // Seul monsite.
        com peut y accéder
4     header("Access-Control-Allow-Origin:␣free.fr"); // Les sous-domaines
        de Free.fr sont autorisés, donc nayi.free.fr y a accès
5 ?>
```

- Alternative : utiliser le site d'origine comme proxy.

Inconvénients

Introduction

Principe

Liaison

Le langage

Syntaxe

Variables

Les Opérateurs

Flot d'exécution

Fonctions

Objets

JSON

L'API D.O.M.

Motivations

Niveaux

DOM Core

DOM HTML

DOM et CSS

Exemples

Conformité

AJAX

Introduction

XMLHttpRequest

exemples

- Si JavaScript est désactivé, Ajax ne peut fonctionner. Il faut demander à l'internaute de l'activer sur son navigateur ou le rediriger vers la version « classique ».
- Les données chargées de façon dynamique ne font pas partie de la page. Elles ne sont donc pas prises en compte par les moteurs de recherche.
- L'aspect asynchrone fait que les modifications se font avec un délai (si le traitement sur le serveur est long), ce qui peut être déconcertant.
- Le bouton « Page précédente » du navigateur ne fonctionne pas sur les requêtes AJAX

Modes d'utilisation *XMLHttpRequest*

Introduction

Principe
Liaison

Le langage

Syntaxe
Variables
Les Opérateurs
Flot d'exécution
Fonctions
Objets
JSON

L'API D.O.M.

Motivations
Niveaux
DOM Core
DOM HTML
DOM et CSS
Exemples
Conformité

AJAX

Introduction
XMLHttpRequest
exemples

L'objet XHR s'utilise de différentes façons selon que l'on souhaite récupérer

- du texte brute : html, json ou autre. Aucune analyse n'est faite.
- un flux XML : le flux est analysé (parsé) et on récupère un arbre DOM.

et de quelle manière :

- synchrone : la fonction utilisée est bloquante.
- asynchrone : la fonction appelée et non bloquante, l'exécution continue et chaque étape de la requête provoque l'exécution d'un gestionnaire d'évènement (listener).

Propriétés de *XMLHttpRequest*

- **onreadystatechange** : Gestionnaire d'événements pour les changements d'état.
- **readyState** : statut de l'objet. Les codes possibles pour le statut de l'objet sont :
 - 0 = non initialisé
 - 1 = ouverture. (open() appelée avec succès)
 - 2 = envoyé. (send() appelée avec succès)
 - 3 = en train de recevoir. (transfert pas terminé)
 - 4 = terminé. (Les données sont disponibles).
- **responseText** : Réponse sous forme de chaîne de caractères.
- **responseXML** : Réponse sous forme d'objet DOM.
- **status** : code numérique de réponse du serveur HTTP
- **statusText** : message accompagnant le code de réponse. (ex : "200 : OK", ...).

Méthodes de *XMLHttpRequest*

- **abort()** : abandonne la requête.
- **getAllResponseHeaders()** : renvoie l'ensemble de l'entête de la réponse sous forme de chaîne de caractères.
- **getResponseHeader("champEntete")** : renvoie la valeur d'un champ d'entête HTTP.
- **open ("method","URL"[,asyncFlag[, "userName" [, "password"]]])** : prépare une requête en indiquant la méthode, l'URL, le drapeau de synchronisation, le nom d'utilisateur et le mot de passe.
- **send (contenu)** : exécute la requête, éventuellement en envoyant les données.
- **setRequestHeader("champ","valeur")** : assigne une valeur à un champ d'entête HTTP qui sera envoyé lors de la requête.

Exemple simple

Lecture d'un flux texte brut en mode synchrone :

```
1 <html>
2 <head><title >Exemple 1</title ></head>
3 <body>
4   <script>
5     ...
6     function ajax() {
7       var xhr=createXHR() ;
8       xhr.open( "GET", "http://localhost/ajax/reponse.txt"
9         , false ) ;
10      xhr.send( null ) ; ^^ | // Appel bloquant
11      if (xhr.status == "200") alert(xhr.responseText) ;
12      else alert( "Erreur␣:␣"+xhr.statusText) ;
13    }
14  </script>
15  <p onclick="ajax();">Cliquez—moi !</p>
16 </body>
</html>
```

Exemple asynchrone 1/2

Lecture d'un flux XML en mode asynchrone :

```
1 <html>
2 <head><title >Exemple 1</title ></head>
3 <body>
4   <script >
5     ...
6     function update() {
7       var xhr=createXHR() ;
8       // Place le gestionnaire d'evenements :
9       xhr.onreadystatechange = updateData ;
10      xhr.open( "GET", "http://localhost/ajax/data.php",
11                true) ;
12      xhr.send(null) ;^^| // Appel non bloquant
13    }
14  </script>
15  <p onclick="update() ;">Actualiser </p>
16 </body>
17 </html>
```

Exemple asynchrone 2/2

Exploitation des données XML :

```
1 <script >
2     ...
3     function updateData () {
4         if (xhr.readyState == 4) // Telechargement fini.
5         if (xhr.status == 200) { // Pas d'erreur.
6             var dt=xhr.responseXML; // Un objet DOM element
7             // Exploitation ave DOM :
8             var cours=dt.getElementsByTagName("Cours");
9             var msg="Liste des cours : ";
10            for (var i=0; i<cours.length; ++i) {
11                msg += cours[i].attributes["label"].value;
12            }
13            alert(msg);
14        }
15        else alert("Erreur : "+xhr.statusText);
16    }
17    ...
18 </script >
```

Quelques références

Introduction

Principe

Liaison

Le langage

Syntaxe

Variables

Les Opérateurs

Flot d'exécution

Fonctions

Objets

JSON

L'API D.O.M.

Motivations

Niveaux

DOM Core

DOM HTML

DOM et CSS

Exemples





Conformité

AJAX

Introduction

XMLHttpRequest

exemples

- <http://www.w3.org/DOM/> 
- <http://json.org/> 
- Validation JSON : `JSON.parse()` 
- <http://api.jquery.com/> 
- « Beginning JavaScript with DOM Scripting and Ajax from Novice to Professional », Apress - 2006