Dynamisme côté client

Le langage JavaScript

fabien.hermenier@unice.fr

dynamisme

côté client
côté serveur

complémentaire !

authentification
accès à des données privées

sûre
simple côté serveur

pre-verification d'un formulaire,
interface avancées (onglets, menu,...)

simple côté client
meilleure interactivité
charge serveur plus faible

Principe du dynamisme côté client

le navigateur télécharge :
HTML, CSS, images
des scripts executables

les scripts sont interprétés pour:
interagir avec le navigateur
manipuler la page courante

JavaScript != Java

buzz word en 1995

Implantation du standard ECMAScript

Langage multi-paradigmes:

- procédural
- orienté objet
- fonctionnel

/!\ JS requiert de la rigueur

ambiguïtés
typage dynamique
typage faible (conversions implicites)
très laxistes sur l'absence de variables
de fonctions

http://javascript.crockford.com/javascript.html

les types de base

nombres	`12; 7.48; NaN; -Infinity; Infinity`
strings	`"toto",'tata'`
booleen	`false, true`
objet	fonctions, tableaux, dates, regex, erreurs
indéfini	`undefined` (absence d'initialisation) `null` (absence délibérée)

```
//Une variable globale
var counter;

function add(x, y) {
    //r est locale
    var r = x + y;
    return r;
}

function inc() {
    counter++;
}
```

```
/**
* Calcule un factoriel de
* de manière recursive
*/
function factoRec(n) {
    if (n == 0) {
        return 1;
    }
    return n * factoRec(n - 1);
}
```

les fonctions

```
function foo(bar, baz) {
   return bar + baz;
}
```

`foo(2, 3)` retourne 5

`foo(2,3,4,false)` retourne 5
(paramètres supplémentaires ignorés)

`foo(2)` retourne `NaN`
(paramètre manquant == `undefined`)

/!\

```
null != undefined
 47 + undefined == NaN
 47 + null == 47
```

les objets

des paires clef -> valeur

```
//Définition
var o1 = new Object();
var o = {};
var o = {‘clef1’ : 1, ‘clef2’ : false}
```

```
//lecture
var v1 = o.clef1
var v2 = o[‘clef1’]
```

```
//ecriture
o.clef1 = v;
o[‘clef’] = v;
```

valeurs hétérogènes et clefs non consécutive autorisées

tableaux

un objet pour des clés entières
+ propriété length prédéfinie

```
//définition
var a1 = new Array();
var a3 = [];
var a3 = [‘eric’,’butters’,’kenny’];
```

```
//lecture
var v = t[0];
```

```
//ecriture
t[1] = ‘dr. chaos’;
```

```
// /!\ length == index du prochain element
t[0] = ‘the coon’; t[10] = ‘captain obvious’
t.length == 11
```

fonctions prédéfinies:

- ajout/retrait `pop, push, shift, unshift`
- manipulations usuels `reverse, sort, slice`

parcours d'un tableau

attention aux trous !

```
function avg_0_en_IaI(notes) {
   var s = 0;
   for (var i = 0; i < notes.length; i++) {
      s += notes[i];
   }
   return s / notes.length;
}

function avg_bien(notes) {
   var s = 0;
    var nb = 0;
   for (var i in notes) { //parcours les index
      s += notes[i];
      nb++;
   }
   return s / nb;
}
```

objets perso. Définition et création

`new` création une instance.
la fonction appelée devient le constructeur

`this` référence l'instance de l'objet et ses méthodes

```
function Etudiant(p, n) {
   this.prenom = p
   this.nom = n
   this.nomComplet = function() {
      return this.prenom + ‘ ‘ + this.nom
   }
}

var b = new Student(‘Kyle’,’Browslovsky’)
b.nomComplet() // == ‘Kyle Browslovsky’
```

JS + XHTML

```
<html>
<head>
    <script src='js/monFichier.js'></script>
</head>
<body>
</body>
</html>
```

Le script JS est un fichier indépendant

séparation des préoccupations

des bibliothèques JS ré-utilisables

Objets JS prédéfinis

`document`	la page XHTML actuelle
`navigator`	le navigateur utilisé
`screen`	l'écran de l'internaute
`history`	historique
`location`	url de la page courante

Pour chercher/modifier des balises XHTML, des propriétés CSS, changer d'URL, executer des requêtes, ...

XHML+JS dans la pratique

Programmation évènementielle:

l'interpréteur JS va exécuter du code en fonction d'évènement

Exemples:

- la balise <p> a été chargée
- toutes les 3 secondes
- clique sur le bouton
- passage de la souris sur un élément

XHTML+JS dans la pratique

onXXXX="du code javascript "

```
…
<body onload="alert(‘chargé’)">
  <h1 onclick="alert(‘hop’)">Titre</h1>
  <p onmouseover="alert(‘dessus’)">
    blablabla
  </p>
  <p><a href=‘#’ onclick="alert(‘hop’)">
      un lien
  </a></p>
</body>
```

http://www.w3schools.com/jsref/dom_obj_event.asp

le DOM

- modèle arborescent représentant la page
- tous les noeuds de l'arbre sont des éléments du document

l'objet `document` pour manipuler le DOM

http://users.polytech.unice.fr/~hermenie/adw/cours/base-js/dom.html

Des méthodes pour naviguer dans le DOM:

```
var x = document.getElementById('list')
x.childNodes;
x.getElementsByTagName('li')[0]
…
```

http://www.w3schools.com/js/js_htmldom.asp

l'objet `document` pour manipuler le DOM

http://users.polytech.unice.fr/~hermenie/adw/cours/base-js/dom.html

Des méthodes pour modifier le DOM:

```
var x = document.getElementById('list')
x.style.color = 'blue';
var items = document.getElementsByTagName('li')
items[0].innerHTML = 'hello'
x.innerHTML += '<li>Nouvel item</li>'
…
```

http://www.w3schools.com/jsref/dom_obj_style.asp

http://www.w3schools.com/js/js_htmldom.asp

Coder efficacement en JS

- avancer par petites étapes
- `console.log()` pour debugger
- `alert('bla')` pour debugger salement
- depuis la console du navigateur
  - test des fonctions
  - affichage des erreurs/warnings