Le langage Javascript (1re partie : les bases)

HENALLUX

Technologies web

IG2 – 2015-2016

Au programme…

- **La syntaxe de Javascript**
  - Règles générales
- **Les valeurs manipulables en Javascript**
  - Types de valeurs
  - Littéraux (comment écrire ces valeurs)
  - Opérations
  - Conversions explicites et implicites

En théorie

- **Les instructions en Javascript**
- **Les fonctions en Javascript**
  - Définition de fonctions
  - Fonctions anonymes
  - Variables locales et variables globales
  - Hoisting

En théorie

La syntaxe de Javascript

- Javascript ignore les blancs.
  - Blanc = espace, tabulation, retour à la ligne…
  - comme HTML
  - autorise une indentation claire [clean code]

- À l'inverse du clean code : les scripts minimalisés
  - Noms de variables aussi courts que possibles
  - aucun blanc inutile
  - illisible par un humain
  - mais permet d'obtenir un fichier plus petit

La syntaxe de Javascript

- Un exemple (extrait) de code minimaliste
  - À éviter dans le cadre de ce cours !

```
ig"),db=function(a,b,c){var d="0x"+b-65536;return
d!==d||c?b:0>d?String.fromCharCode(d+65536):String.fromCharCode(d>>10|5529
6,1023&d|56320)};try{I.apply(F=J.call(v.childNodes),v.childNodes),F[v.chil
dNodes.length].nodeType}catch(eb){I={apply:F.length?function(a,b){H.apply(
a,J.call(b))}:function(a,b){var
c=a.length,d=0;while(a[c++]=b[d++]);a.length=c-1}}}function
fb(a,b,d,e){var
f,h,j,k,l,o,r,s,w,x;if((b?b.ownerDocument||b:v)!==n&&m(b),b=b||n,d=d||[],!
a||"string"!=typeof a)return
d;if(1!==(k=b.nodeType)&&9!==k)return[];if(p&&!e){if(f=_.exec(a))if(j=f[1]
){if(9===k){if(h=b.getElementById(j),!h||!h.parentNode)return
d;if(h.id===j)return d.push(h),d}else
if(b.ownerDocument&&(h=b.ownerDocument.getElementById(j))&&t(b,h)&&h.id===
j)return d.push(h),d}else{if(f[2])return
I.apply(d,b.getElementsByTagName(a)),d;if((j=f[3])&&c.getElementsByClassNa
me&&b.getElementsByClassName)return
I.apply(d,b.getElementsByClassName(j)),d}if(c.qsa&&(!q||!q.test(a))){if(s=
r=u,w=b,x=9===k&&a,1===k&&"object"!==b.nodeName.toLowerCase()){o=g(a),(r=b
.getAttribute("id"))?s=r.replace(bb,"\\$&"):b.setAttribute("id",s),s="[id=
'"+s+"'] ",l=o.length;while(l--
)o[l]=s+qb(o[l]);w=ab.test(a)&&ob(b.parentNode)||b,x=o.join(",")}if(x)try{
return]
```

La syntaxe de Javascript

- Écrire des commentaires en Javascript
  - Syntaxe similaire au C et à Java
  - `//` pour le reste de la ligne
  - `/*` … `*/` pour plusieurs lignes / une partie de ligne

- À propos des identificateurs en Javascript :
  - commencent par une lettre, `$` ou `_`
  - composés de lettres, chiffres, `$` et `_`
  - (sauf les mots réservés)
  - (éviter les `$` et `_` en début : variables prédéfinies)

- Javascript est sensible à la casse !
  - `mavariable` ≠ `maVariable` ≠ `MAVARIABLE`

La syntaxe de Javascript

- Liste des mots réservés en Javascript

abstract	debugger	final	instanceof	public	transient
boolean	default	finally	int	return	true
break	delete	float	interface	short	try
byte	do	for	long	static	typeof
case	double	function	native	super	var
catch	else	goto	new	switch	void
char	enum	if	null	synchronized	volatile
class	export	implements	package	this	while
const	extends	import	private	throw	with
continue	false	in	protected	throws	

- *Note : certains de ces mots ne sont pas utilisés dans la version actuelle de Javascript…*

La syntaxe de Javascript

- Les instructions Javascript se terminent par `;`.
  - Mais ce n'est pas obligatoire… un retour à la ligne peut suffire (à éviter, pour la clarté).
  - Ceci dit, attention !

```
function produit (x, y) {
  var prod =
    x + y
  return
    x + y
}
```

Les valeurs manipulables

- **Types de valeurs**

- **Littéraux**
  - Comment écrire des valeurs ?
- **Opérations**
- **Conversions explicites et implicites**
  - Un sujet délicat !

En théorie

Ensuite : *instructions en Javascript*

Valeurs : types de valeurs (1/2)

- Obtenir le type d'une valeur / variable :
  - `typeof 12` donne "number"
  - `typeof true` donne "boolean"
  - `typeof "salut"` donne "string"

  - Autres types :
    - "function" : les fonctions
    - "object" : tout le reste, y compris les tableaux
      - dont `null` (attention, pas NULL ni Null !)
    - "undefined" : variables sans valeur
      - pour les variables déclarées mais pas (encore) initialisées

Valeurs : types de valeurs (2/2)

- JS est un langage non/faiblement typé.
  - Pas besoin de donner un type lors de la déclaration d'une variable.
  - Une variable peut changer de type au cours de l'exécution.

```
var x = 7;
x += " nains";
```

- Deux implications
  - Le type d'une variable n'est connu qu'à l'exécution : pas de vérification de type !
  - Javascript réalise de nombreuses conversions implicites (avec des règles parfois étonnantes).

Les instructions en Javascript

- Plutôt standard… !

Ensuite : *fonctions en Javascript*

Instructions : affectations, blocs

- Affectations

  id = *expr*

```
x = 37
total = prix * quantite
a = b = c = 4 * x
```

  - Ainsi que les raccourcis **+= -= *= /= %= ++ --**

- Blocs d'instructions

  { *instructions* }

Instructions : conditionnelles

- Structure conditionnelle simple (`if`)

```
if (condition)
  instruction
[ else instruction ]
```

- Structure conditionnelle avec gardes (`switch`)

```
switch (expr)
{
  case val : instruction
  …
  [ default : instruction ]
}
```

  - `break` pour éviter de passer d'un cas à l'autre !

Instructions : itératives

- Boucles `while` et `do`

```
while (condition)
  instruction
```

```
do
  instruction
while (condition)
```

- Boucle `for` (voir aussi for-in pour les objets)

```
for (init ; condition ; incrémentation)
  instruction
```

  - On peut déclarer à la volée dans l'init.
  - Utiliser `,` pour séparer les composantes complexes du for.

- Échappements

```
break
```

```
continue
```

Instructions : conditions

- Les conditions (if, while, …) sont converties en booléens :
  - Tout devient true SAUF
    - pour "number" : `0` et `NaN`
    - pour "string" : `""`
    - `undefined`
    - `null`

- Donc, `if (!x)` signifie…
  - si x est null, pas défini, 0, NaN, false ou la chaîne vide

- Donc, `if (x)` signifie…
  - si x est défini en tant que nombre non nul, chaîne non vide, booléen true ou objet (autre que null).

Les fonctions en Javascript

- **Définition de fonctions**

- **Fonctions anonymes**
  - Littéraux de type « fonction »
- **Fonctions prédéfinies**
- **Fonctions et scopes**
- **Variables locales et variables globales**
- **Hoisting**

En théorie

Fonctions

- Déclaration d'une fonction

```
function nom ( ident , ident … ) {
  instructions
}
```

  - Quitter la fonction / renvoyer une valeur :

```
return [ expr ]
```

- Appel d'une fonction

```
nom ( expr , expr … )
```

  - Attention : aucune vérification ni de type ni de nombre entre les paramètres formels et les arguments effectifs !