V1.1 septembre 2012

Introduction aux WebAPPS en HTML5

Alain Duglas

JavaScript

Définition

- Est un langage de type Script.
- Il est spécifié par l'organisation ECMA.
- Plusieurs implémentations:
  - Jscript de Microsoft.
  - ECMAScript implémentation de référence.
  - JavaScript.
- La dernière version de la recommandation est ECMA 262.

Définition

- La syntaxe est basée sur le modèle ‘C’.
- L’écriture en JavaScript est sensible à la casse.
- JavaScript est un langage utilisant le principe du caractère de fin de ligne d’instruction.
  - Le caractère est ‘;’
- Les commentaires de source sont identiques au langage ‘C’:
  - ‘//’ commentaire ligne
  - ‘/* xxxx */’ commentaire bloc XXXX étant le bloc de code.

Définition

- Les identifiants (nom de variable, nom de fonction, …) sont soumis aux contraintes suivantes:
  - Pas de limite dans le nombre de caractère.
  - Pas d'espace dans l'identifiant.
  - Composé de lettre et de chiffre et des caractères '_' et '**$**'.
  - Les chiffres ne peuvent pas débuter les identifiants.

i	**OK**
Nom_de_variable	**OK**
v101	**OK**
$vv	**OK**
1var	**KO**

Définition

- Les mots-clés de JavaScript

break	do	if	switch	var
case	else	in	this	void
catch	false	instanceof	throw	while
continue	finally	new	true	with
default	for	null	try	
delete	function	return	typeof	

Définition

- Les mots réservés de JavaScript pour les évolutions futures.

abstract	enum	int	short
boolean	export	interface	static
byte	extends	long	super
char	final	native	synchronized
class	float	package	throws
const	goto	private	transient
debugger	implements	protected	volatile
double	import	public	

Les variables

- JavaScript est un langage qui impose la déclaration des variables.
- Les variables sont faiblement typées.
- La déclaration des variables est réalisée avec le mot-clef ‘var’.
- Le type de la variable est défini à son initialisation ou sa première utilisation
- Les types proposés sont:
  - Numérique.
  - Booléens.
  - Chaîne de caractères.
  - Objet.
  - Tableau.

Exemples:
var compteur; //déclaration de la variable compteur
var compteur=0; //déclaration et initialisation

Les variables numériques

- Les variables sont donc non typées
- Pour javascript les numériques sont codés sur 64 bits en virgule flottante.
- Il ne fait pas de différence entre entier et décimale.

Exemples:
var compteur = 1; //entier
var pi = 3.14;
var test = 6.02^{e}23;

Les variables booléens

- Les valeurs possibles sont:
  - **true**.
  - **false**.
- Il est impossible d'utiliser comme en 'C' 0 pour **false** et 1 pour **true**.

Les variables chaine de caractères

- Le codage des chaines de caractères est UNICODE 16 bits.
- L'initialisation d'une chaine de caractères est réalisée par des séquences de caractères délimités par les caractères simples ou double guillemet. **Les simples guillemets sont à privilégier.**

Exemples:
var message1 = 'message de test 1';
var message2 = "message de test 2";

Les variables chaine de caractères

- Comme en ‘C’ le caractère ‘\’ est un caractère d’échappement.

échappement	description
\b	Retour arrière
\f	Nouvelle page
\n	Nouvelle ligne
\r	Retour chariot
\'	Simple guillemet
\"	Double guillemet
\\	anitslash
\xDD	Caractère latin 'DD' valeur hexa
\uDDDD	Caractère unicode 'DDDD' valeur hexa

Les variables chaine de caractères

- Les chaines de caractères possèdent des caractéristiques:
  - **'length'** permet de récupérer le nombre de caractères de la chaine.
  - L'opérateur '+' permet de concaténer des chaines.
  - Les opérateurs '==' et '!=' permettent de comparer l'égalité du contenu de 2 chaines.
  - Il est possible de comparer le contenu de chaine avec les opérateurs '<','>','<=', '>='.
  - Les chaines sont immuables. Il est impossible de changer le contenu d'une chaine.

Les variables chaine de caractères

```
var mess1= 'message de test';
   if (mess1.length > 0){
       //traitement
   }
```

```
var ch1 = 'val ch1';
var ch2 = 'val ch2';
   if (ch1 == ch2){
       //traitement
   }
```

Les variables tableaux

- Un tableau est en réalité un type Objet.
- Un tableau peut contenir des valeurs:
  - Numériques.
  - Chaine de caractères.
  - Booléens.
- Les tableaux peuvent êtres indicés ou associatifs.

```
var tabIndice = new Array; //création tableau
tabIndice [0] = 100; //insertion élément
```

```
var tabAsso = new Array; //création tableau
tabAsso ['key001'] = 100; //insertion élément
```

Les variables tableaux

- Les propriétés et les méthodes utilisables sur les tableaux

Nom	Type	Description
length	P	Nombre d'éléments
Concat()	M	Concaténation de tableau
join()	M	Transformation en chaine
pop()	M	Suppression dernier élément
push()	M	Ajout élément
reverse()	M	Inverser les éléments
shift()	M	Suppression premier élément
slice()	M	Extraction d'une partie du tableau
splice()	M	Ecraser des éléments
sort()	M	Trie du tableau
unshift()	M	Insertion en début de tableau
forEach(callback,instance)	M	Permet de parcourir un tableau indexé (pas IE < 9)

Les conversions

- JavaScript propose une API de conversion

Fonction	Description	arguments
parseInt(str,rdx)	Conversion d'une chaine en une valeur entière	**str**: valeur à convertir **rdx**: (option) base de conversion (10 par défaut)
parseFloat(str)	Conversion d'une chaine en valeur float.	**str**: valeur à convertir

Les variables divers

- Il existe pour les variables deux valeurs remarquables:
  - **null**.
  - **undefined**.

- '**null**' indique que la variable ne possède aucune valeur. C'est utilisé pour la manipulation des variables objets.

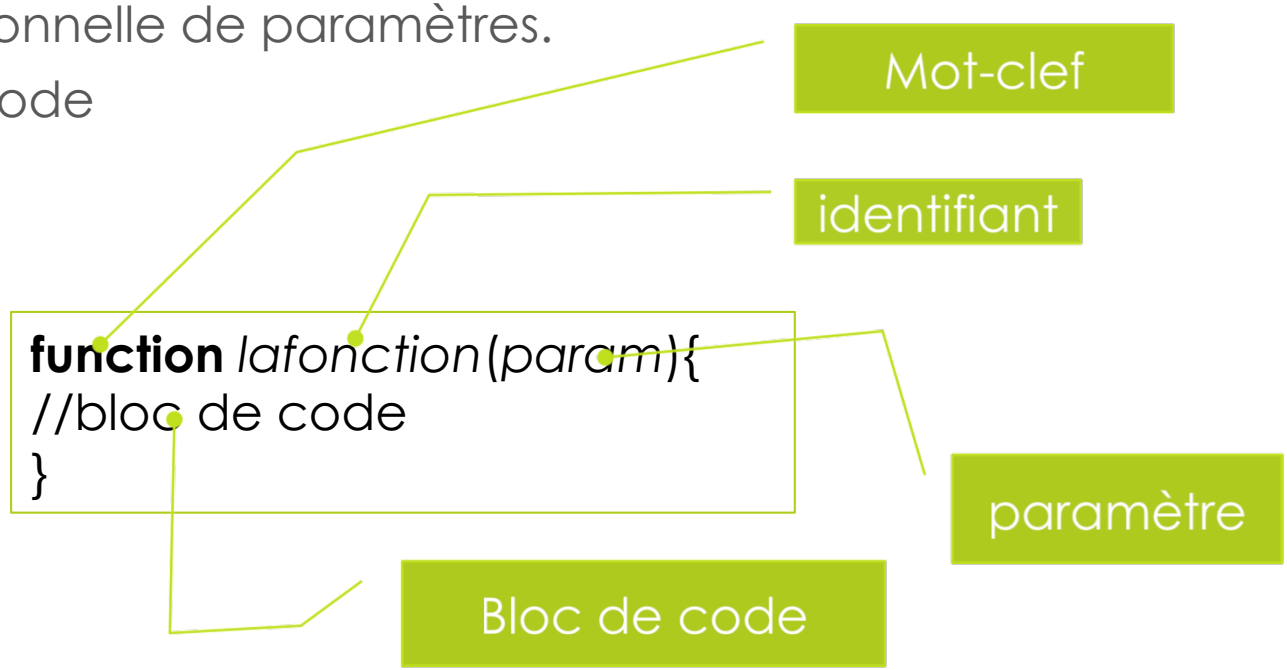
- '**undefined**' est utilisé pour la manipulation des variables non initialisées.

Les structures de contrôles

- JavaScript propose les structures de contrôles classiques:
  - Fonction.
  - Itérative.
  - Conditionnelle.

Les fonctions

- Une fonction est caractérisée par:
  - Le mot-clef '**function**'.
  - Un identifiant (nom).
  - Une liste optionnelle de paramètres.
  - Un bloc de code



Les fonctions

- Il est possible de définir des fonctions anonymes (sans nom).
- Les fonctions anonymes sont très utilisées dans les environnements évènementiels.

```
var fct = function(prm){
  //traitement
}

fct('test prm');// appel de la fonction par la variable
```

Les fonctions

- Il est possibles de récupérer les paramètres d'une manière dynamique en utilisant la pseudo variable **arguments**.

Pseudo variable

```
function test(){
        alert(arguments.length);
       alert(arguments[1];
        }
test(1,2,3);
```

Les structures itératives

- Il existe plusieurs sortes de structure itérative:
  - Do/while.
  - For.
  - For/in

DO/WHILE

- Permet d'exécuter une itération tant qu'une condition est vraie.

```
var compteur = 0;

do{
   compteur++;
}while(compteur < 10);
```

FOR

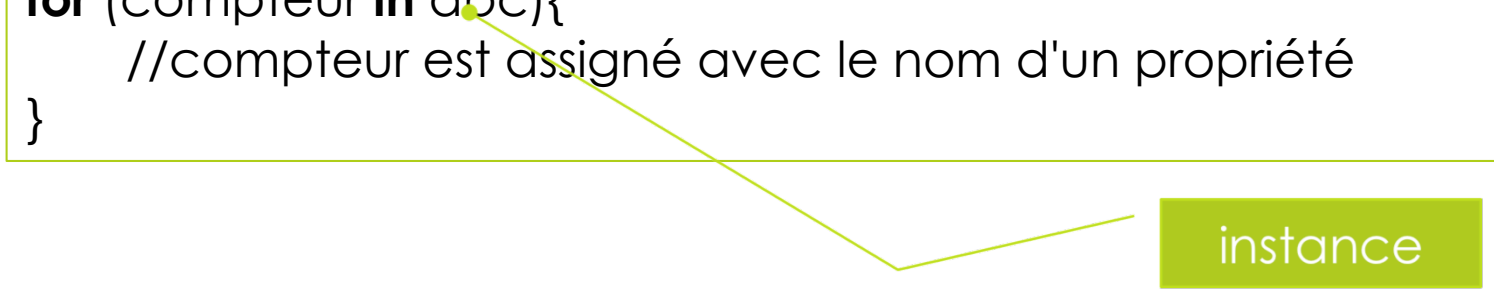
- Itération traditionnelle.
- La structure possède plusieurs zones.
  - Zone d'initialisation effectuée en premier avant la première itération.
  - Zone de test de rebouclage avant l'itération suivante.
    - Si test 'false' alors sortie de la structure.
  - Zone de rebouclage. Exécutée après l'itération et avant le test de rebouclage.

```
var compteur;
for (compteur=0; compteur < 10 ; compteur++){
//traitement
}
```

FOR/IN

- Permet de parcourir les propriétés d'une instance d'un objet.
- A chaque itération le nom d'une propriété objet est récupéré.
- Si l'objet est un tableau les indices ou les clef sont retournés.

```
var compteur; //variable de travail
var abc = new objt(); //création d'une instance

for (compteur in abc){
      //compteur est assigné avec le nom d'un propriété
}
```



Les structures conditionnelles

- Les structures proposées sont:
  - La structure '**if**'
  - La structure '**switch**'
  - L'opérateur ternaire '**?**'.

IF

- Même principe qu'en '**C**'.
- Test d'une valeur booléenne.
  - Une variable.
  - Un opérateur de comparaison.
  - Le retour d'une fonction.
- Deux blocs(préféré) ou ligne de code possible:
  - Normal.
  - Alternatif (optionnel avec le mot-clef '**else**').

operateur

```
if (compteur == 10){
    //traitement normal
}
else{
    //traitement alternatif
}
```

Switch

- Identique au langage 'C'
- Test de cas multiple sur une valeur.

```
switch(compteur){
    case 1:
        //traitement 1
    break;
    case 2:
        //traitement 2
    break;
    default:
        //traitement défaut
}
```

Variable à tester

Cas de test ici –1--

Cas par défaut
Si aucun vrai

Les opérateurs

Opérateur	Description
.	Accès aux membres d'un objet.
[]	Accès aux éléments d'un tableau.
()	Appel d'une fonction.
new	Création d'une instance d'un objet.
++	Incrémentation PRE ou POST.
--	Décrémentation PRE ou POST.
-	Moins unaire.
+	Plus unaire.
~	Complément binaire.
!	Complément logique.

Les opérateurs

Opérateur	Description
delete	Permet de supprimer un élément.
typeof	Retourne le type opérande.
void	Retourne une valeur indéfinie.
*,/,%	Multiplication, division, reste.
+,-	Addition, soustraction.
<<	Décalage d'un entier à gauche.
>>	Décalage d'un entier à droite.
>>>	Décalage d'un entier à droite sans signe.
<,<=	Inférieur et inférieur ou égal.
>,>=	Supérieur et supérieur ou égal.

Les opérateurs

Opérateur	Description
instanceof	Test du type de l'instance.
in	Test si une propriété existe.
==	Test d'égalité.
!=	Test de différence.
===	Test égalité d'identité d'objet (même instance)
!==	Test différence d'identité d'objet (pas la même instance).
&	ET logique.
^	OU exclusive logique.
|	OU logique.
&&	Test ET logique.

Les opérateurs

Opérateur	Description
||	Test OU logique.
?:	Opérateur ternaire.
=	Affectation.
*=,+=,-=,etc	Opérateurs combinés.
,	Evaluation multiple.

Les exceptions

- JavaScript propose un mécanisme de gestion des anomalies de type **exception**.
- Deux acteurs dans ce modèle:
  - Un producteur d'anomalie.
    - Utilisation du mot-clef '**throw**'.
  - Un consommateur.
    - Utilisation des mots-clés '**try** / **catch** / **finaly**'.

Les exceptions

- '**try**' est l'indicateur de bloc à protéger.
- '**catch**' est l'indicateur des blocs de traitement des anomalies.
- '**finally**' est l'indicateur de bloc commun.

```
try{//bloc à protéger
}
catch(param){//gestionnaire d'erreur
}
finally{//exécuter dans tous les cas
}
```

Javascript et les Objets

- Javascript possède les concepts objets mais leurs implémentations sont différentes des langages comme 'C++' ou 'Java'.
- La notion de classe n'existe pas.
- La notion d'objet existe.

Création instance

Création donnée membre

Création fonction membre

```
var obj = new Object();
obj["dtmenbre"] = "valeur1";
obj["methode"] = function(parametre1) {return parametre1; }
var ret = obj.dtmenbre;
var ret2 =  obj.methode("valeur1");
```

Javascript et les Objets

- Il est aussi possible de supprimer une propriété d'un objet
  - L'instruction est **delete**.
- L'instruction delete est aussi possible sur les tableaux mais attention au effets de bord pour les tableaux indicé. Pas de problème pour les tableaux associatif.
  - La place dans le tableau n'est pas supprimé
  - Il est préférable d'utiliser "**splice**".

```
delete laliste['key'];
```

Javascript et les objets

- Contrairement à d'autre langage le mot-clé '**this**' est très important.
- Le système ne fait aucune différence entre fonction et une fonction membre. Il faut donc préfixer l'utilisation des membre d'instance par '**this**'.
- Attention "**this**" est l'instance courante de l'appelant.

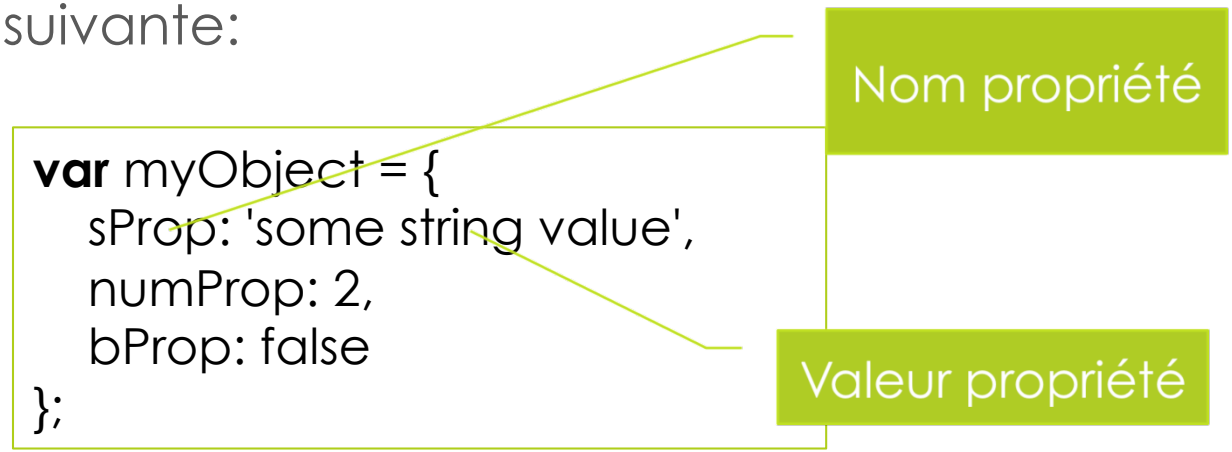
Javascript et les objets

- Exemple de pseudo classe

```
function MaClasse(parametre1, parametre2) {
    this.attribut1 = parametre1;  //création affectation d'une donnée membre'
    this.attribut2 = parametre2;  //création  affectation d'une donnée membre
    this.methode = function() {  //création d'une fonction membre
        //traitement de la méthode
    }
}
var obj = new MaClasse("v1", "v2");  //création d'une instance'
obj.methode(); //appel de la méthode sur l'instance'
```

Javascript et les objets

- Une autre méthode pour créer des objets est l'utilisation de la syntaxe "Javascript Object Literal".
- Cette syntaxe permet la création d'un instance de la manière suivante:



Javascript et les objets

- Il y une technologie associée à Javascript qui permet de manipuler les objects.
- Cette technologie est **JSON**.
  - **J**ava**S**cript **O**bject **N**otation.
- JSON permet:
  - Créer des instances.
  - Sérialiser et déserialiser des instances.

JSON

- Exemple

```
var str_object= "{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}}";
```

JSON

- Pour pouvoir utiliser un objet JSON chargé via un fichier ou une requête réseau il est nécessaire d'utiliser l'instruction JSON.parse

Text JSON

```
var myObject = JSON.parse(myJSONtext);
```

JSON

- La serialisation d'un objet au format JSON dans une chaîne de caractère est réalisée avec l'instruction JSON.stringify

```
var obj = new Object();
obj.total = 100;
var tab = new Array();
tab.push(1);
tab.push(2);
tab.push(3);
obj.liste = tab;
var str = JSON.stringify(obj);
```

Création d'une propriété

Création d'un tableau

Création d'une propriété

Sérialisation

Javascript Heritage

- Il est possible d'implémenter une relation d'héritage en Javascript.

```
function LaClasseBase(prm1) {
        this.memb1 = prm1;
        this.proc = function() { //traitement}
}
function LaClasse(prm1, bsprm) {
        LaClasseBase.call(this,prm1);//constructeur base
        this.lc1 = bsprm;
        this.mth = function(){}
}

var obj = new LaClasse("pzm", "pbase");
obj.proc(); //methode classe base
obj.mth();  //methode classe courante
```

Les prototypes

- En réalité JavaScript est un langage orienté Prototype
- Chaque Type possède un prototype.
  - Le prototype est accessible via la propriété "prototype".

```
function LaClasse(p1) {this.prp1 = p1;};
LaClasse.prototype = {
        lamethode: function(p2) {return this.p1 + p2;}
        };
var obj = new LaClasse(100);
var ret = obj.lamethode(10);
```

prototype

Code dynamique

- JavaScript propose un mécanisme de traitement du code dynamique.
- l'instruction utilisé est 'eval'

```
var codeFct = "function dynfct(p) { return p + 1; }";

eval(codeFct);

function fct(x)
{
  var res = dynfct(x);
  alert(res);
}
```

Chargement du code

Fonctions globales Javascript

Fonction	Description
encodeURI(uri)	Encode l'uri sauf les caractères ,/?:@&=+$
decodeURI(uri)	Inverse de encodeURI
encodeURIComponent(uri)	Encode tous les caractères
decodeURIComponent(uri)	Inverse de encodeURIComponent
escape(str)	Encodage d'une chaine (portable) sauf *@-_+./
eval(str)	Code dynamique

Fonctions globales Javascript

Fonction	Description
isFinite(val)	Test si valeur dans la plage autorisée
isNaN(val)	Test si valeur est n'est pas un number
Number(str)	Conversion en nombre
unescape(str)	Inverse de escape
String(str)	Conversion d'un nombre en chaine
parseInt(str)	Conversion chaine en entier
parseFloat(str)	Conversion chaine en float

V1.1 septembre 2012

JavaScript DOM

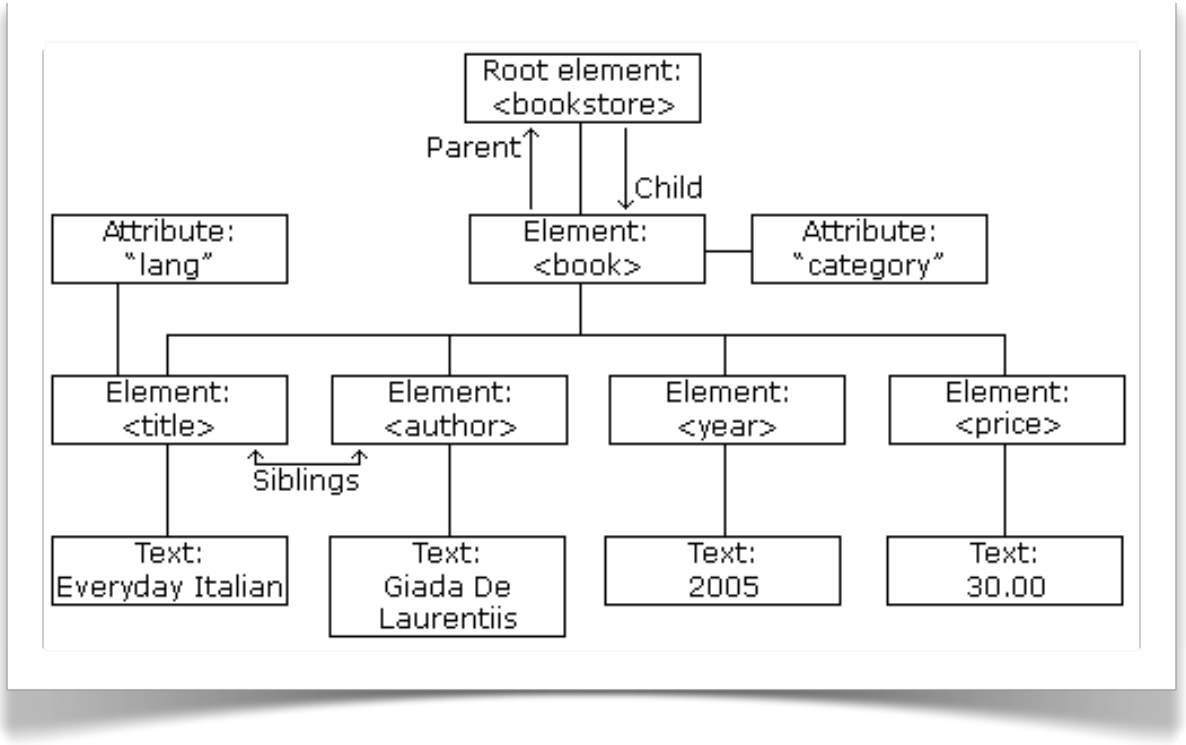
DOM

- JavaScript permet de manipuler une structure de type DOM dans les environnements Web clients.
- **D**ocument **O**bject **M**odel (DOM) du w3c.
- DOM est une interface normalisée par le W3C.
  - Plusieurs niveaux disponibles.
    - DOM 1
      - Recommandation publiée en 1998.
    - DOM 2
      - Recommandation publié en 2000.
    - DOM 3
      - Recommandation publiée en 2004

DOM

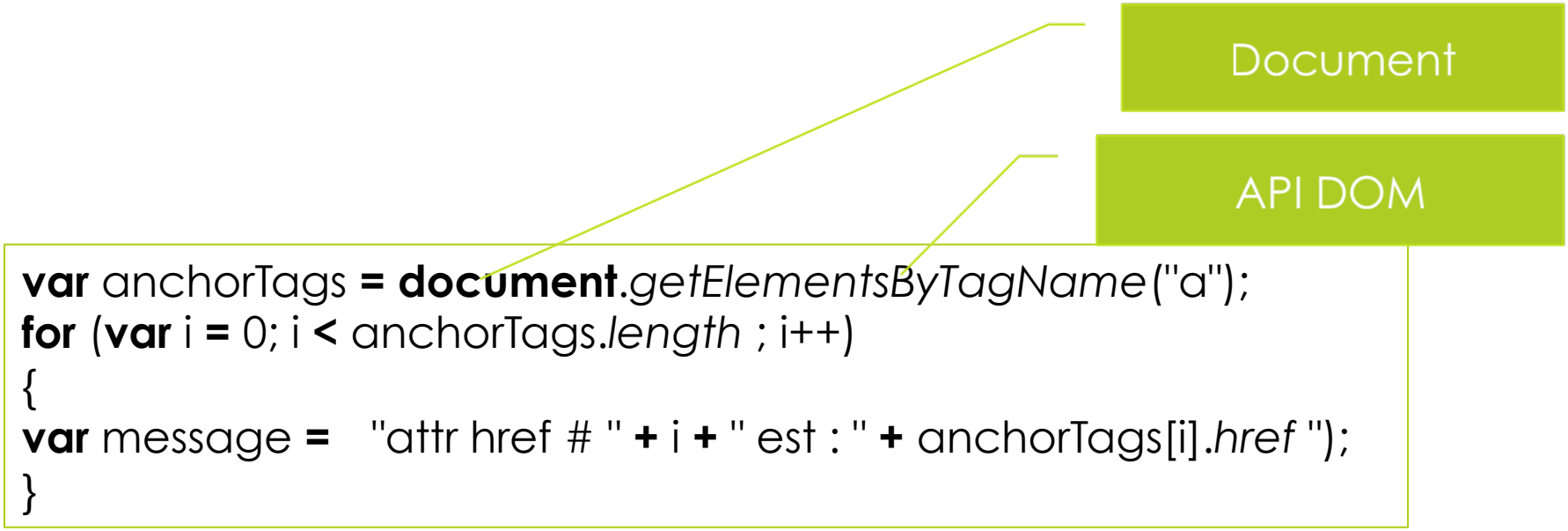
- Définition du DOM
- Le DOM est une recommandation du W3C qui décrit une interface indépendante de tout langage de programmation et de toute plate-forme, permettant à des programmes informatiques et à des scripts d'accéder ou de mettre à jour le contenu, la structure ou le style de documents XML. Le document peut ensuite être traité et les résultats de ces traitements peuvent être réincorporés dans le document tel qu'il sera présenté.

DOM


Root element:
<bookstore>
Parent
Child
Attribute:
"lang"
Element:
<book>
Attribute:
"category"
Element:
<title>
Element:
<author>
Element:
<year>
Element:
<price>
Siblings
Text:
Everyday Italian
Text:
Giada De
Laurentiis
Text:
2005
Text:
30.00

DOM

- Exemple de l'interaction DOM et Javascript.



DOM

- Création d'un nœud (**node**)
  - Un nouvel élément n'est pas ajouté implicitement au document

Verbe	Description
createDocumentFragment	Création d'un fragment de document.
createElement	Création d'un élément
createTextNode	Création d'un élément avec du texte en sous élément.

DOM

- Gestion des nœuds (**nodes**)

Verbe	Description
childNodes[]	Retourne le tableau des enfants
parentNode	Récupère l'élément parent
firstChild	Récupère le premier enfant
getElementsByTagName	Récupère un tableau d'enfant avec un nom spécifique.
lastChild	Récupère le dernier enfant
nextSibling	Récupère le prochain élément au même niveau dans l'arbre.
previousSibling	Récupère l'élément précèdent dans l'arbre au même niveau.

DOM

- Parcourt dans l'arbre

Verbe	Description
body	Retourne le tableau des enfants
documentElement	Récupère l'élément parent
getElementByID	Récupère l'élément qui possède l'id spécifié.
getElementsByTagName	Récupère un tableau d'enfant avec un nom spécifique.

DOM

- Ajout/suppression de nœuds.

Verbe	Description
appendChild	Ajout un enfant
cloneNode	Clone un enfant
innerHTML	Ajoute du text orienté HTML
insertBefore	Insert un enfant avant un autre
removeChild	Retire un enfant de l'arbre
replaceChild	Replace un enfant dans l'arbre (à la même place)

DOM

- Informations sur un noeud

Verbe	Description
data	Données du nœud.
hasChildNodes	Y a t'il des enfants.
id	ID du nœud.
nodeName	Nom du nœud.
nodeType	Type du nœud.
nodeValue	Valeur du nœud.

DOM

- Les attributs

Verbe	Description
Attributes[]	Retourne les attributs d'un nœud.
getAttribute	Retourne un attribut spécifié par son nom.
removeAttribute	Retire un attribut.
setAttribute	Ajoute un attribut avec sa valeur sur un noeud.

DOM

Il est important de se référer au W3C pour l'API officielle.

Les évènements

- DOM propose un modèle évènementiel.
- Il est possible d'être notifié pour:
  - Souris.
  - Keyboard.
  - Frame/ Objet
  - Form
- Dans le cas de système tactile une api spécifique **multi-touch** est disponible.

Les évènements

- L'abonnement est disponible via:
  - Attribut évènementiel de type onxxx sur les éléments du DOM.
  - Fonction globale.
    - addEventListener / removeEventListener
    - Spécifique à IE
      - attachEvent / detachEvent.

```
var formulaire = document.getElementById("idele");
formulaire.addEventListener("submit", fct_callback, false);
```

Type Event

callback

Pas de capture

V1.1 septembre 2012

Communication

Communication

- Une api native est proposée par les navigateur pour communiquer en utilisant le protocole **HTTP**.
- La communication est possible dans 2 modes:
  - **Synchrone**.
  - **Asynchrone**.
- Le mode **Asynchrone** est privilégié.
- L'objet est **XmlHttpRequest**.

Communication

- Pour communiquer avec le serveur il est nécessaire de créer une instance de XmlHttpRequest.
- L'API d'accès est normalisée mais pas la création de l'instance.

Communication

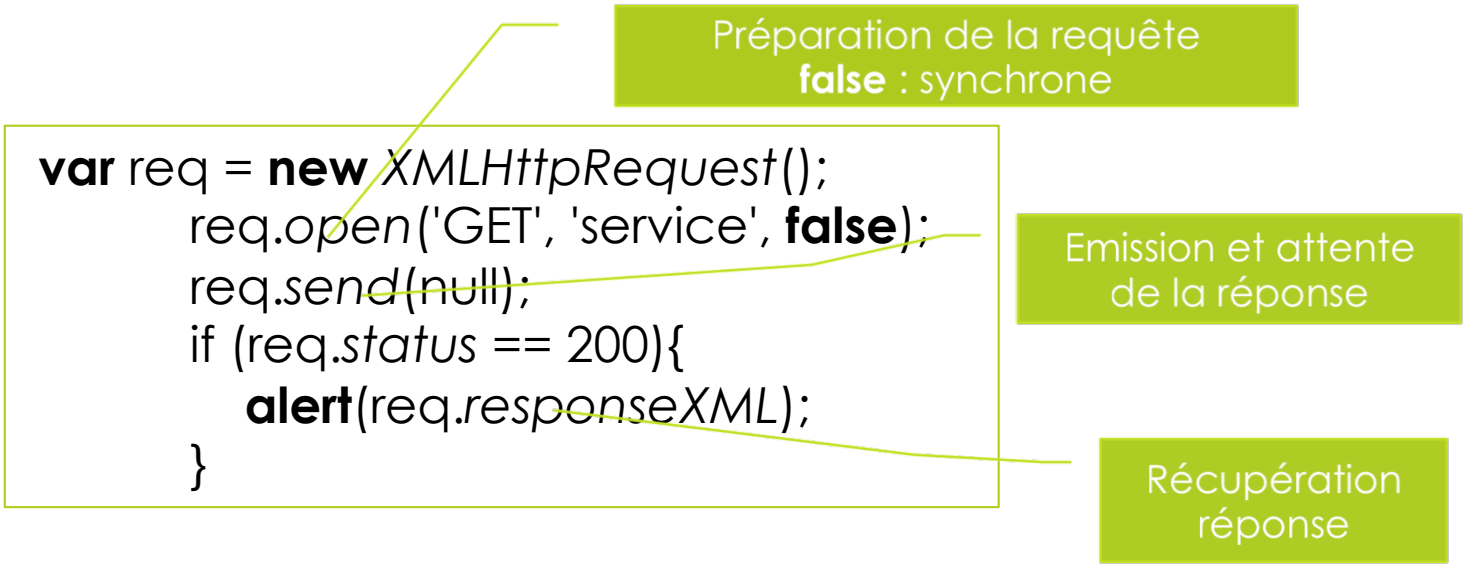
```
function createXhrObject()
{
    if (window.XMLHttpRequest)
        return new XMLHttpRequest();

    if (window.ActiveXObject)
    {
        var names = ["Msxml2.XMLHTTP.6.0", "Msxml2.XMLHTTP.3.0", "Msxml2.XMLHTTP", "Microsoft.XMLHTTP"
        ];
        for(var i in names)
        {
            try{ return new ActiveXObject(names[i]); }
            catch(e){}
        }
    }
return null;
}
```

WebKit,Chrome,safarie,firefox,ie >= 7

IE < 7

Communication

- Exemple en mode synchrone.




```
var req = new XMLHttpRequest();
        req.open('GET', 'service', false);
        req.send(null);
        if (req.status == 200){
            alert(req.responseXML);
        }
```

Communication

- Exemple en mode asynchrone

```
var req = new XMLHttpRequest();
        req.open('GET', 'service',true);
        req.onreadystatechange = function(){
            if (req.readyState == 4){
                if (req.status == 200){
                    alert(req.responseXML);
                }
            }
        }
        req.send(null);
```

Préparation de la requête
true : asynchrone

Abonnement événement de communication

Test fin de communication
code : 4

Emission

CSS

Définition CSS

- Les feuilles de styles (en anglais "Cascading Style Sheets", abrégé CSS) sont un langage qui permet de gérer la présentation d'une page Web. Le langage CSS est une recommandation du World Wide Web Consortium (W3C), au même titre que HTML ou XML

Définition CSS

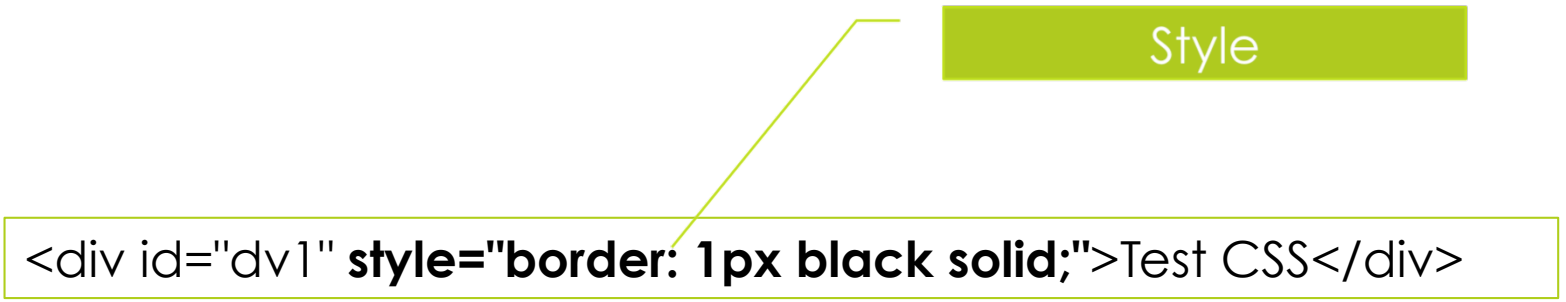
- Les styles permettent de définir des règles appliquées à un ou plusieurs documents HTML. Ces règles portent sur le positionnement des éléments, l'alignement, les polices de caractères, les couleurs, les marges et espacements, les bordures, les images de fond, etc.

Définition CSS

- Il est possible d'appliquer le style de plusieurs façons.
  - par un attribut «style» sur l'élément html.
  - par un bloc style dans la page.
  - par un fichier chargé.

Définition CSS

- Attribut style sur une balise



Définition CSS

- Bloc de style

Sélection

```
<style type="text/css">
        #dv1{
          border: 1px black solid;
        }
 </style>
```

Définition CSS

- Un fichier à part et sa liaison dans le fichier html

Fichier: lestyle.css

```
#dv1{
        border: 1px black solid;
    }
```

Liaison dans la page html

```
<link type="text/css" rel="stylesheet" href="styles/lestyle.css"/>
```

Définition CSS

- l'application du style est exécutée d'une manière hiérarchique.
  - le fichier chargé (le moins prioritaire).
  - le bloc (un peu plus prioritaire).
  - l'attribut (le plus prioritaire).

Définition CSS

- CSS permet d'appliquer un ou plusieurs style sur les éléments du DOM.
- CSS utilise des selecteurspour accéder aux éléments.
- Les sélecteurs possibles sont:
  - Sur les TagName.
  - Sur les id des éléments.
  - Sur des classes d'éléments.
    - Une classe est un regroupement logique d'éléments.
  - Sur des combinaisons de sélecteur.

Sélecteur CSS

sélecteur	Description	CSS Min X
.cls	Sélection des éléments associé à la class '**cls**'	1
#r001	Sélection de l'élément associé à l'id '**r001**'	1
*	Sélection de tous les éléments	2
item	Sélection des éléments '**<item>**'	1
Item,article	Sélection des éléments '**<item>**' et '**<article>**'	1
parent enfant	Sélection des éléments '**<enfant>**' dans les éléments '**<parent>**'	1

Sélecteur CSS

sélecteur	Description	CSS Min X
racine>branche	Sélection des éléments '**<branche>**' dont le parent est un élément '**<racine>**'.	1
b1+b2	Sélection des éléments '<b2>' situés immédiatement après les éléments '<b1>'.	2
[attr]	Sélection de tous les éléments possédant un attribut '**attr**'.	2
[attr=valeur]	Sélection de tous les éléments possédant un attribut '**attr**' avec la valeur '**valeur**'.	2
[attr~=valeur]	Sélection de tous les éléments possédant un attribut '**attr**' dont la valeur possède au moins '**valeur**'.	2

Sélecteur CSS

sélecteur	Description	CSS Min X
e:link	Sélection des éléments '**e**' non visités.	1
e:visited	Sélection des éléments '**e**' visités.	1
e:active	Sélection des éléments '**e**' actifs.	1
e:hover	Sélection des éléments '**e**' survolés par la souris.	1
e:focus	Sélection des éléments '**e**' ayant le focus	2

Sélecteur CSS

sélecteur	Description	CSS Min X
e:first-letter	Sélection de la première lettre du texte des éléments '**e**'.	1
e:first-line	Sélection de la première ligne du texte des éléments '**e**'.	1
e:first-child	Sélection du premier enfant des éléments '**e**'.	2
e:before		2
e:after		2

Sélecteur CSS

sélecteur	Description	CSS Min X
el1~el2	Sélection des éléments '**<el2>**' précédés par un élément '**<el1>**'.	3
[attr^=value]	Sélection des éléments possédant l'attribut '**attr**' dont la valeur commence par '**value**' .	3
[attr$=value]	Sélection des éléments possédant l'attribut '**attr**' dont la valeur se termine par '**value**' .	3
[attr*=value]	Sélection des éléments possédant l'attribut '**attr**' dont la valeur contient '**value**' .	3
::not(sel)	Sélection des éléments qui ne répondent pas au sélecteur '**sel**'	3

Propriétés CSS

- Après la sélection des éléments du document, CSS applique des '**properties**' à la sélection.

Animation	Grid	Print
Background	Hyperlink	Ruby
Border and outline	Linebox	Speech
Box	List	Table
Color	Margin	Text
Content Paged Media	Marquee	2D/3D Transform
Dimension	Multi-column	Transition
Flexible Box	Padding	User-interface
Font	Paged Media	
Generated content	Positioning	

Unités utilisables

- Unités absolues
  - Millimètre "**mm**".
  - Centimètre "cm".
  - Inch "**in**" (1 inch = 2,54 cm).
  - Point "**pt**" (1 pt = 1/72 in).
  - Pica "**pc**" (1 pc = 12 pt).
- Unités relative
  - Pixel "**px**" dépendant du système d'affichage.
  - "**em**" (1 em = 12 pt) utilisé pour les caractères.
  - "**ex**" pour les polices.
  - Pourcentage "**%**".

Les couleurs

- Notation nominale.
  - 16 couleurs (aqua, black, blue,…).
- Notation hexadécimale.
  - Débute par un "#" puis un triplet de proportion de rouge, vert et bleu.
  - Exemple **#ff98ca**.
- Notation fonctionnelle.
  - Utilisation de la fonction rgb(RR,VV,BB).
  - Exemple **rgb(255,255,0)**.

Les polices

- Les familles de police
  - Propriété : **font-family**
  - Valeur : **Nom police**
  - Exemple: **font-family: *helvetica***;
- Style de police
  - Propriété: **font-style**
  - Valeur: **normal , italic , oblique**.
  - Exemple: **font-style : *italic***;
- Taille
  - Propriété: **font-size**
  - Valeur: **taille**
  - Exemple: **font-size: *10pt;***

Le texte

- Espacement entre les mots.
  - Propriété **word-spacing**.
- Espacements entre les lettres.
  - Propriété **lettre-spacing**.
- Alignement du texte.
  - Propriété **text-align**
- Retrait première ligne
  - Propriété **text-indent**.
- Interligne
  - Propriété **line-height**.

Les média

- CSS propose des media différents qui sont des support physique.

Media	Description
all	Pour tous les devices
aural	Synthétiseur de parole
braille	Device tactile braille
embossed	idem
handled	Petit appareil
print	Pour les aperçu avant impression
projection	Projecteur ou transparent
screen	Ecran ou moniteur
tty	Ecran textuel
tv	télévision

Les média

- Il est possible de spécifier le ou les média (media query)

```
<style media="screen">/*pour écran*/</style>
<style media="print">/*pour impression*/</style>
<style media="screen,print"></style>
```

```
@media screen
  {
  p.test {font-family:verdana,sans-serif;font-size:14px;}
  }
@media print
  {
  p.test {font-family:times,serif;font-size:10px;}
  }
```

Les média

- Il est aussi possible de spécifier un taille d'écran dans le **media query**.

```
<link rel="stylesheet" type="text/css" href="style_mini.css"
    media="screen and (max-width: 480px)" />

@media screen and (max-device-width:480px) {
/*todo */
}
```

Positionnement

- Le positionnement en CSS
- Propriété : **position**
  - Static
    - Valeur par défaut
  - Fixed
    - Position par rapport à la fenêtre du navigateur
  - Relative
    - Position par rapport à la position normale.
  - Absolute
    - Position par rapport au container Parent.

V1.1 septembre 2012

CSS3

CSS3

- CSS3 apporte de nouvelles propriétés.
- Ces nouvelles propriétés sont accessibles plusieurs noms:
  - Le nom normalisé.
  - Le nom pour l'implémentation webkit.
    - Préfixe –webkit-nom-prop.
  - Le nom pour l'implémentation firefox.
    - Préfixe –moz-nom-prop.
  - Le nom pour l'implémentation IE.
    - Préfixe –ms-nom-prop.
  - Le nom pour l'implémentation Opéra.
    - Préfixe –o-nom-prop.
- La présence multiple pour toutes les implémentations est fréquent.

CSS3

```
#lediv
{
    transform: rotate(30deg);
    -ms-transform: rotate(30deg); /* IE 9 */
    -moz-transform: rotate(30deg); /* Firefox */
    -webkit-transform: rotate(30deg); /* Safari and Chrome */
    -o-transform: rotate(30deg); /* Opera */
}
```

CSS3

- Quelques nouveautés de CSS3:
  - Ajout sur les bordures.
  - Les effets de texte.
  - Ajout sur les fonts.
  - Les transformations 2D et 3D.
  - Les transitions.
  - Les animations.

Bordures

- Les nouvelles propriétés sont:
  - border-radius.
  - box-shadow.
  - border-image.

```
div
{
box-shadow: 10px 10px 5px #888888;
}
```

```
div
{
border:2px solid;
border-radius:25px;
-moz-border-radius:25px;
}
```

```
div
{
border-image:url(border.png) 30 30 round;
-moz-border-image:url(border.png) 30 30 round;
-webkit-border-image:url(border.png) 30 30 round;
-o-border-image:url(border.png) 30 30 round;}
```

Effets de texte

- Les nouvelles propriétés:
  - text-shadow.
  - word-wrap.

```
h1
{
text-shadow: 5px 5px 5px #FF0000;
}
```

Transformations 2D

- Transformation 2D:
  - transform.
  - tranform-origin.

fonction	description
matrix(x,x,x,x,x,x)	Transformation 2D via une matrice
translate(x,y)	Transformation 2D suivant les axes x et y
translateX(n)	Transformation 2D via l'axe x
translateY(n)	Transformation 2D via l'axe y
Scale(x,y)	Transformation 2D d'echelle suivant les axes x et y
scaleX(n)	Transformation 2D d'echelle suivant l'axe x
scaleY(n)	Transformation 2D d'echelle suivant l'axe y
rotate(a)	Rotation 2D avec angle a

Transformations 2D

fonction	description
skew(xa,ya)	Transformation 2D skew sur les axes x et y a
skewX(a)	Transformation 2D skew suivant l'axe x
skewY(a)	Transformation 2D skew suivant l'axe y

```
div
{
transform: rotate(30deg);
-ms-transform: rotate(30deg);
-webkit-transform: rotate(30deg);
-o-transform: rotate(30deg);
-moz-transform: rotate(30deg);
}
```

Transformations 3D

- Les propriétés de transformation 3D:
  - transform.
  - transform-orign.
  - transform-style.
  - perspective.
  - perspective-origin.
  - backface-visibility.

Transformations 3D

```
#bt1{
transform: rotateY(130deg);
-webkit-transform: rotateY(130deg);
-moz-transform: rotateY(130deg);
}
```

Transitions

- Les propriétés de transitions:
  - transition.
  - transition-property.
  - transition-duration.
  - transition-timing-function.
  - transition-delay.

Transitions

```
#bt1{
width:100px;
height:100px;
background:red;
transition:width 2s;
-moz-transition:width 2s;
-webkit-transition:width 2s;
-o-transition:width 2s;
}

#bt1:hover
{
width:300px;
}
```

Animations

```
div
{
width:100px;
height:100px;
background:red;
animation:myfirst 5s;
-moz-animation:myfirst 5s;
-webkit-animation:myfirst 5s;
}
```

```
@-webkit-keyframes myfirst
{
from {background:red;}
to {background:yellow;}
}
```

```
@keyframes myfirst
{
from {background:red;}
to {background:yellow;}
}

@-moz-keyframes myfirst
{
from {background:red;}
to {background:yellow;}
}
```

Librairie JavaScript

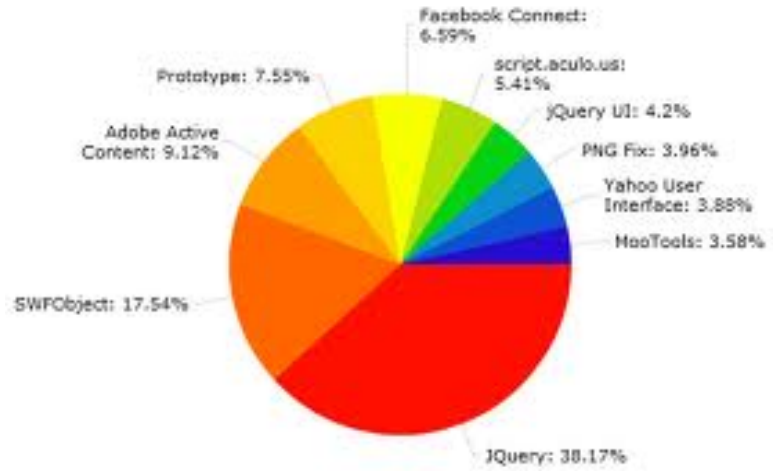
Libraries JavaScript

- Le développement JavaScript/HTML5 est inconsevable sans utilisation d'une ou plusieurs librarires JavaScript.
- Les librairies prennent en charge beaucoup d'éléments à faible valeur ajoutée.
  - Le support des multiples navigateurs.
  - L'aide au parcourt de du DOM.
  - Des composants de base (boite de dialogue, …).
- Exemple de librairies:
  - Jquery.
  - prototype.

JQuery

JQuery

- Nous allons décrire JQuery car actuellement cette librairie est la plus utilisée.



JQuery

- Cette librariries et la base d'autre projet de la communauté JQuery.
  - JQuery UI
  - JQuery Mobile.

Introduction

- JQuery est une librairie qui permet de parcourir le DOM courant de la page .
- JQuery permet l'accès à tous les éléments de l'arbre via des sélecteurs.
  - De façon unitaire (un seul élément).
  - Dans un jeux de résultat (multiple éléments).
- Le principe de sélection (sélecteur) est conforme à la philosophie des sélecteurs CSS.

Base JQuery

- Jquery doit être inséré dans la page via la balise script
  - 2 versions disponibles:
    - Version de production (minimize).
    - Version de développement (normal).

JQuery

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

Hello JQuery #1

- Voici un exemple d'utilisation de Jquery.
- Dans cet exemple nous allons nous abonner à l'événement '**Document Ready**' puis nous abonner à l'événement '**click**' sur la balise **<a>**.

Hello JQuery #2

- La page html

```
<html>
 <head>
 <script type="text/javascript" src="jquery.js"></script>
 <script type="text/javascript">
    //le script à venir
 </script>
 </head>
 <body>
      <a href="">Hello</a>
 </body>
</html>
```

HTML5

Introduction

- HTML5 est avant tout un langage de description d'interface homme machine.
- Mais il propose aussi des API normalisées pour gérer entre autre:
  - La persistance.
  - Le mode dégradé.
  - La communication API
  - Les traitements concurrents.

introduction

- Le travail autour d'HTML5 a débuté par le WHATWG (**W**eb **H**ypertext **A**pplication **T**echnology **W**orking **G**roup) puis repris par le W3C en mars 2007.
- Les spécifications ne sont pas encore définitives mais le W3C encourage les développeurs à utiliser HTML5 aujourd'hui.
- HTML5 est éligible à la mise en œuvre de WebAPPS.

Représentation HTML

Représentation

- HTML5 existe en deux modes:
  - HTML5 non rigoureux (HTML5).
  - HTML5 rigoureux (XHTML5).

Représentation

- Les documents doivent référencer une DTD.
- La déclaration de la DTD en HTML5 est réduite à sa plus simple expression.

```
<!DOCTYPE html>
```

Représentation

- Les navigateurs peuvent différencier HTML5 de XHTML5 par l'utilisation du type MIME.
- Le type mime peut être spécifié par:
  - Le header HTTP pendant la communication.
  - Dans la balise **meta** du document.

```
<meta http-equiv="Content-type" content="application/xhtml+xml">
```

Représentation

Exemple sans xml

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
    </body>
</html>
```

Représentation

Exemple avec xml

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html>
<html xmlns="http://www/w3/org/1999/xhtml>
    <head>
    </head>
    <body>
    </body>
</html>
```

Représentation

- La balise "**meta**"
  - C'est une balise d'information.
  - Les informations sont mises à disposition pour certain service mais sans garantie d'utilisation.
- Cette balise est surtout utilisée par:
  - Les navigateurs.
  - Les moteurs de recherche.
  - Les outils d'indexation.
  - Etc.

Représentation

- La syntaxe d'une balise "**meta**" est :

```
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />

<meta name="description" content="Une description de la page..." />
```

Nom de l'information

Valeur de l'information

```
<meta name | http-equiv="Mot-clé" content="Valeur" />
```

Représentation

- Exemple d'utilisation de la balise "**meta**" dans le cas des mobiles.
  - Name: "**viewport**"
  - Content: information pour l'affichage.

```
<meta name="viewport" content="width=550">
<meta name="viewport" content="initial-scale=2.0">
<meta name="viewport" content="width=device-width">
```

Représentation

- Propriétés possibles (non exhaustif) de **viewport**

Propriété	Description	valeur
width	Largeur de l'écran. Il est conseillé de ne pas mettre une valeur numérique (pixel).	• **device-width** Spécifie la largeur de l(écran de l'appareil. • **valeur numérique**
height	Idem mais pour la hauteur.	• **device-height** Spécifie la hauteur de l'écran de l'appareil. • **valeur numérique**.

Représentation

Propriété	Description	valeur
user-scalable	Spécifie si l'utilisateur peut modifier l'échelle (zoom)	• **1, yes, true** Zoom autorisé • **0, no, false** Zoom non autorisé
Initial-scale	Valeur initiale du zoom	• **1.0** Zoom 1 1px du viewport est 1 pixel de l'écran

Représentation

Propriété	Description	valeur
minimal-scale	Valeur minimal de l'échelle	• **0 à 10.0**
maximal-scale	Valeur maximal de l'échelle	• **0 à 10.0**

Représentation

- Dans le cas de petit écran de mobile il est possible de forcer le scrolling de la fenêtre.

```
<body onload="window.scrollTo(0,1)">
<!– code de la page →
</body>
```

Représentation

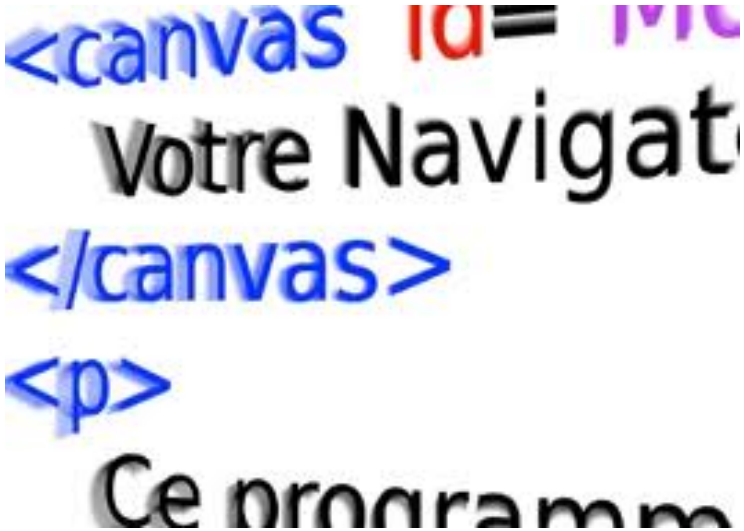
- Les nouveaux éléments du formulaire

Elément	Description
tel	Numéro de téléphone
email	Adresse email
url	url
search	Moteur de recherche
range	Plage numérique

Représentation

- HTML5 propose une gestion de 'placeholder'.
  - Ceci permet de spécifier une valeur avant la saisie dans un élément de formulaire.

Canvas



Canvas

- C'est une zone permettant à un script de dessiner.
  - Un canvas est une balise de type html qui doit être ajouté au DOM de la page.

```
<canvas id="cn1" width="200" height="300">
</canvas>
```

Canvas

- Access à un Canvas en JavaScript

```
var cnv = document.getElementById('cn1');
var ctx = cnv.getContext('2d');
ctx.fillStyle = 'red';
ctx.fillRect(20,30,50,50);
```

Récupération du canvas

Récupération du context

Actions de dessin

Canvas

- Via son context le canvas propose une api de tracé de dessin.
  - Via du tracé de forme
  - Via la gestion d'un cheminement de dessin.

Canvas

- API de forme

Fonction	Description	arguments
fillRect(x,y,w,h)	Rectangle plein	x,y :coin supérieur gauche du rectangle. w,h : largeur et hauteur
strokeRect(x,y,w,h)	Rectangle surligné	idem
clearRect(x,y,w,h)	Rectangle vide	idem
fill()	Applique un remplissage	Pas d'argument
Stroke()	Style de trait	idem

Canvas

- API de cheminement

Fonction	Description	arguments
beginPath()	Début du cheminement	Pas d'argument
closePath()	Fin du cheminement	idem
moveTo(x,y)	Début d'un sous cheminement	x,y: point de départ
lineTo(x,y)	Trace une ligne du point actuel à x,y	x,y: nouveau point

Canvas

Fonction	Description	arguments
rect(x,y,w,h)	Ajout d'un rectangle au chemin	x,y :coin supérieur gauche du rectangle. w,h : largeur et hauteur
arcTo(x1,y1,x2,y2,r)	Ajout d'un arc de cercle au chemin	x1,y1 :point1 x2,y2 :point2 r : rayon de l'arc
arc(x,y,r,a1,a2,c)	Ajoute un arc	x,y: point de départ R: rayon A1,a2: angle debut et fin C:sens rotation
bezierCurveTo(cx1,cy1,cx2,cy2,x,y)	Ajoute une courbe de Bézier au chemin	cx1,cy1: point 1 cx2,cy2: point 2 x,y : point d'arrivé

Canvas

Fonction	Description	arguments
fillStyle	Remplissage forme	Code couleur CSS
strokeStyle	Style des lignes	Code couleur et style CSS
lineWith	Epaisseur ligne	nombre
lineJoin	Style pour les jointure de ligne	**bevel**, **round**, **miter**
lineCap	Style des fin de ligne	**butt**, **round**, **square**

Canvas

Fonction	Description	arguments
miterLimit	Valeur pour miter	nombre
globalAlpha	Valeur de transparence	Nombre positif différent de '0'

Canvas

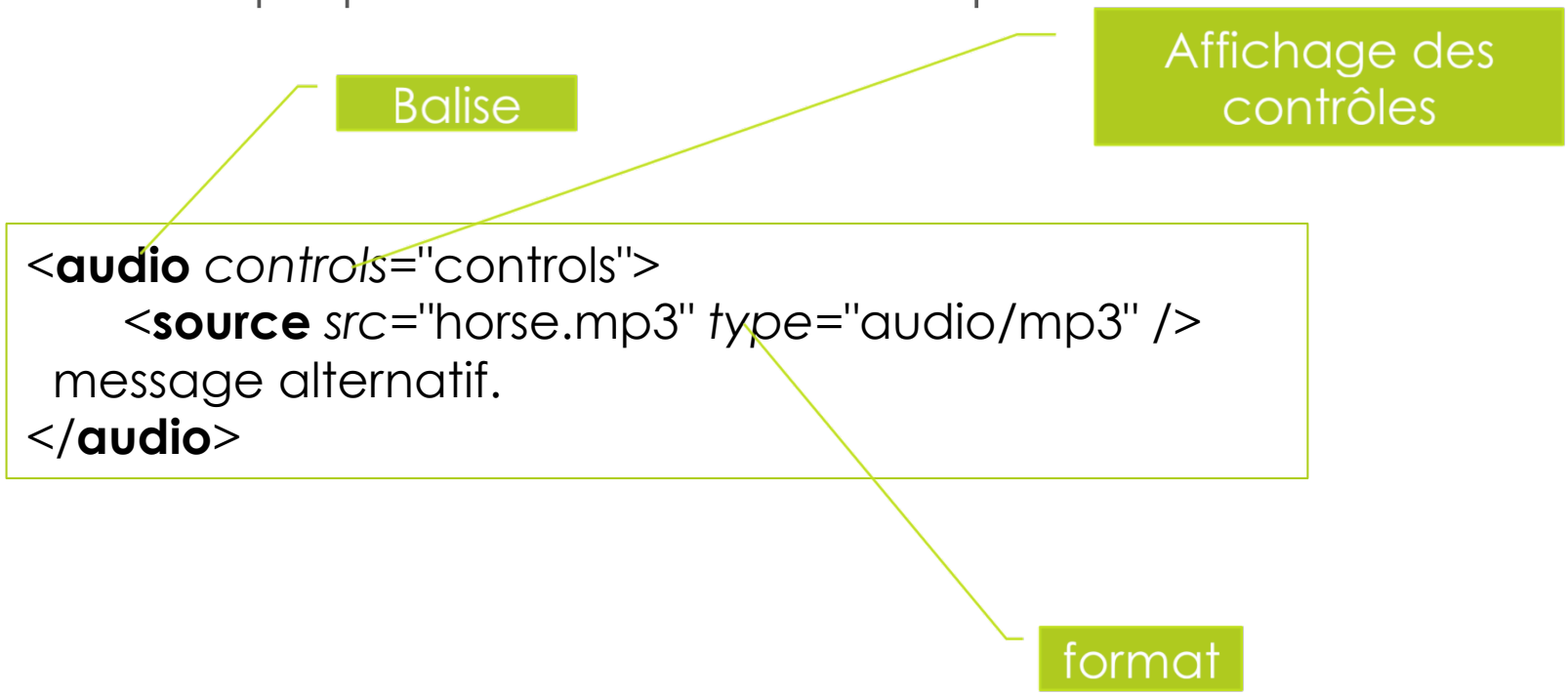
- Il est possible d'utiliser des mécanismes de transformation pour:
  - Changement d'échelle (zoom).
  - Rotation.
  - Déplacement horizontal et vertical.

Média

- HTML5 propose une balise dédiée pour la lecture vidéo.

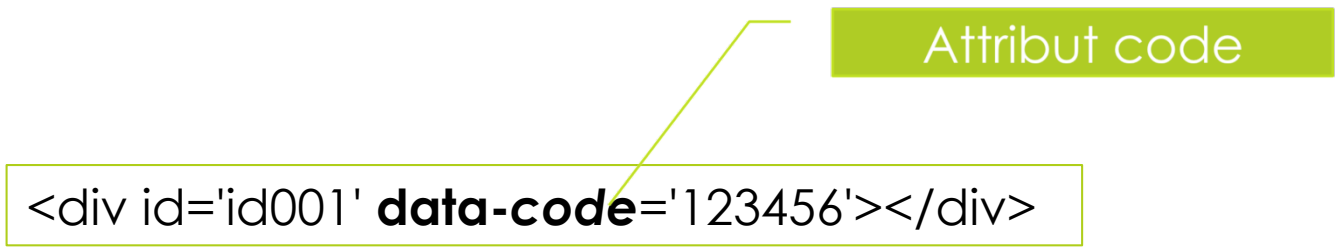
Média

- HTML5 propose une balise dédiée pour la lecture audio.



Plus

- Avec HTML 5 il est possible de mettre en œuvre des attributs propriétaires sur les éléments du document.
- Les attributs propriétaires sont structurés
  - **data-*XXXXX***
    - **data-** est le radical il est obligatoire.
    - ***XXXXX*** est le nom propriétaire



API Html5

API HTML5

- Geolocalisation.
- Persistance (Web Storage).
- Mode dégradé (Offline).
- Web Socket.
- BackGround (Web Worker).
- Communication API.

Geolocalisation

Geolocalisation

- Cette api permet de récupérer la position du navigateur.
- Les éléments pouvant permettre le positionnement sont:
  - Adresse IP.
  - GPS.
  - GSM.
- En fonction de l'élément utilisé pour la localisation le temps de traitement peut être plus ou moins important.
  - L'adresse IP est souvent résolue par un serveur.

Geolocalisation

```
function testGeo() {
  if (navigator.geolocation) {
    alert('HTML5 Geolocation OK');
      } else {
     alert('HTML5 Geolocation KO');
      }
}
```

Geolocalisation

- L'API de geolocalisation
  - getCurrentPosition.
    - Permet de récupérer la position courante.
  - watchPosition.
    - Permet d'être notifié si la position change.
    - Fonction d'abonnement
  - clearWatch.
    - Fonction de désabonnement.

Géolocalisation

callback

callback

```
geolocation.getCurrentPosition(updateLocation, handleLocationError);

geolocation.watchPosition(updateLocation, handleLocationError);
<<Retourne un jeton>>

geolocation.clearWatch(watchId);
```

Fonction callback

```
function recupPos(position) {
  var latitude = position.coords.latitude;
  var longitude = position.coords.longitude;
  var accuracy = position.coords.accuracy;
//.... traitement
}
```

Persistance

Persistance

- Il est possible avec HTML 5 de gérer des données locales au navigateur.
- Il y avait des projets non normalisés pour mettre en œuvre ce concept.
  - Google GEAR.
- Les données locales sont hébergées par le navigateur.
- Il est impossible d'accéder directement au système de fichier de la machine hôte.

Persistance

- 4 modes de persistance.
  - Session storage.
  - Local storage.
  - Data base
  - IndexedDB.

Persistance "Storage"

- Le modèle Storage est la mise à disposition de dictionnaire persistants.
  - Le modèle est de type KEY / VALUE.
- Deux modes :
  - Session Storage.
    - Les données sont liées à une session utilisateur (web session).
  - Local storage
    - Les données sont liées à un serveur (url).

Persistance "Storage"

- Test de la présence des fonctions de persistance sur le navigateur

```
function testPersitance() {
  //sessionStorage
  if (window.sessionStorage) {
    alert('sessionStorage OK');
  } else { alert('sessionStorage KO');
  }
  //localStorage
  if (window.localStorage) {
    alert('localStorage OK');
  } else { alert('localStorage KO');
  }
}
```

Persistance "Storage"

- Enregistrement dans la session d'une donnée

```
window.sessionStorage.setItem('KEYDATA', 'data value');
ou
window.sessionStorage.KEYDATA = 'data value';
```

- Récupération d'une donnée dans la session

```
window.sessionStorage.getItem('KEYDATA');
Ou
alert(window.sessionStorage.KEYDATA);
```

Persistance "Storage"

- Enregistrement locale d'une donnée

```
window.localStorage.setItem('KEYDATA', 'data value');
ou
window.localStorage.KEYDATA = 'data value';
```

- Récupération d'une donnée locale

```
window.localStorage.getItem('KEYDATA');
Ou
alert(window.localStorage.KEYDATA);
```

Persistance "DataBase"

- Il est possible d'accéder à une base de données embarquée dans le navigateur.
- La majorité des navigateurs utilisent la base SQLite.
- Ce mode de persistance est considérée aujourd'hui comme obsolète.

Persistance "DataBase"

- Utilisation d'une base de données locale.

```
function onSelect(tx,rw){
var wrk = '';
        for (var cpt = 0; cpt < rw.rows.length,cpt++){
         wrk += rw.ID +'--';
        }
}
function accDB(){
        var size = 10*1024*1024; //10MB
        var db = openDatabase('labase','1.0','labse de test',size);
        db.executeSql('CREATE TABLE IF NOT EXIST' +
                        'latable(ID INTEGER PRIMARY KEY ASC, info TEXT, info 2 TEXT)',[]);
        db.executeSql('INSERT INTO latable(info , info2' VALUES (? , ?)',['i1','i2']), onOK,onError);

        db.executeSql('SELECT * FROM latable',[], onSelect,onError);
}
```

Persistance "IndexedDB"

- IndexedDB est le modèle à privilégier.
- C'est un système de persistance de type NoSQL (Not Only SQL).
- Ce système permet de persister des objets dans des collections.
- Les Objets insérés peuvent être indexés.

Persistance "IndexedDB"

- Ouverture d'une base indexedDB

```
var db = null;
var actDB = indexedDB.open("repository");
actDB.onsuccess = function(e) {
    db = e.target.result;
};
actDB .onfailure = function(e) {console.log(e);};
```

Persistance "IndexedDB"

- Création d'une collection (Object Store)
  - La création de la collection est associée à une description des objets à insérer.
  - L'information **KeyPath** permet de spécifier à la base l'attribut(s) unique des objets à insérer.

```
var reqSetV = db.setVersion(v);
reqSetV.onfailure = function(e) {console.log(e);};
reqSetV.onsuccess = function(e) {
var store = db.createObjectStore("ledico",
      {keyPath: "identity"});
 };
```

Persistance "IndexedDB"

- Création d'un indexe

```
store.createIndex("idxname","name", {unique: true});
```

Persistance "IndexedDB"

- Ajout d'un objet dans la collection

```
var trans = db.transaction(["ledico"], IDBTransaction.READ_WRITE, 0);
 var store = trans.objectStore("ledico");
 var request = store.put({
   "info": "un message de test",
   "identity" : "valeur unique"
 });

 request.onsuccess = function(e) {//TODO
 };

 request.onerror = function(e) {console.log(e.value);};
```

Persistance "IndexedDB"

- Récupération d'objet dans la base

```
var trans = db.transaction(["ledico"], IDBTransaction.READ_WRITE, 0);
var dico = trans.objectStore("ledico");
  // Get everything in the store;
  var rangeK = IDBKeyRange.lowerBound(0);
  var curseur = dico.openCursor(rangeK );
  curseur.onsuccess = function(e) {
    var row = e.target.result;
    if(row){
    //TODO
    row.continue(); //next
    }
  };
```

Persistance "IndexedDB"

- Récupération d'objets avec l'utilisation d'un index.

```
var rangeK = IDBKeyRange.lowerBound(0);
var widx =
db.transaction(["ledico"]).objectStore("ledico").index("idxname");
widx.openCursor(rangeK).onsuccess = function(evt){
//TODO
};
```

Offline

Offline

- HTML5 propose une API de gestion locale de l'application.
- Il est possible de sauver des éléments de l'application web:
  - Pages.
  - Scripts.
  - Style.
- L'API utilisée est "**application Cache**".

Offline

```
function testCache(){
	if (window.applicationCache) {
		alert('Cache OK');
	}
	else{
		alert('Cache KO');
	}
}
```

Offline

- Il est nécessaire d'installer un fichier '**manifest**' pour décrire les éléments offlines.

```
<!DOCTYPE html> <html manifest="test.cachetest">
<!– TODO →
</html>
```

Offline

- Fichier **manifest** minimal

```
CACHE MANIFEST
index.html
stylesheet.css
images/logo.png
scripts/main.js
```

Offline

- Un fichier **manifest** réaliste

```
CACHE MANIFEST
CACHE:
index.html
stylesheet.css
images/logo.png
scripts/main.js
static.html
NETWORK:
login.php
http://lesite.com
FALLBACK:
Login.php static.html
```

Offline

- Définition des zones:
- CACHE
  - Les fichiers décrits dans cette zone sont utilisables en mode offline.
- NETWORK:
  - Les fichiers décrits dans cette zone sont utilisables uniquement si l'application est ON LINE (pas de cache).
- FALLBACK
  - Relation entre un fichier en erreur et un fichier local.

Offline

- La récupération et le traitement du fichier "**manifest**" par le navigateur peut être associé à un type mime.
- Le type mime normalisé pour le fichier "**manifest**" est :
  - **text/cache-manifest**
- Dans le cas de l'utilisation du serveur Apache il peut être nécessaire d'ajouter la ligne suivant au fichier ".htaccess"

Offline

- La gestion du cache est une action importante des applications web.
- Les possibilités d'update du cache sont:
  - L'utilisateur clear les données du navigateur.
  - Le fichier manifest est updaté sur le serveur.
  - Par programmation.

Offline

- Récupération de l'état du cache

```
var stateCache = window.applicationCache
switch(stateCahe.status){
        case stateCache.UNCACHED:
        case stateCache.IDLE:
        case stateCache.CHECKING:
        case stateCache.DOWNLOADING:
        case stateCache.UPDATEREADY:
        case stateCache.OBSOLETE:
}
```

Offline

- Demande de mise à jour du cache par le navigateur

```
var lecache = window.applicationCache;
lecache.update();
//TODO
if (lecache.status == window.applicationCache.UPDATEREADY){
      lecache.swapCache();
}
```

Offline

- Il est possible d'être notifié d'événement autour du '**cache application**'

```
window.applicationCache.addEventListener('updateready',callback);
ou
window.applicationCache.onupdateready = function(){//TODO};
```

  - Les évènements sont:
    - Cached.
    - Checking.
    - Downloading.
    - Error.
    - Noupdate.
    - Obsolete.
    - Progress.
    - Updateready.

Communication

Web Socket

- Test de la présence du module WebSocket

```
function testWebSock() {
  if (window.WebSocket) {
   alert('WebSocket OK');
  } else {
     alert('WebSocket KO');
  }
}
```

Web Socket

- exemple

```
var url = "ws://leserver:8080/service";
w = new WebSocket(url);
w.onopen = function() {
  alert("open");
  w.send("message de test");
}
w.onmessage = function(e) {
  alert(e.data);
}
w.onclose = function(e) {
  alert("closed");
}
```

Concurrence

Concurrence (Web Worker)

- HTML5 propose l'implémentation de tâche backgroud.
- C'est la notion de thread.

```
function testWorker() {
        if (typeof(Worker) !== undefined) {
                alert('Work OK');}
        else{
                alert('Work KO');}
        }
}
```

Web Worker

- La création d'un worker est possible:
  - Via un worker spécifique.
  - Via le script(s) principal.

```
var leworker = new Worker("echoWorker.js");
```

Web Worker

- Le créateur d'un worker peut recevoir les messages issus du worker.
- Pour recevoir les messages il est nécessaire de s'abonner via la fonction **onmessage**.

```
var leworker = new Worker("echoWorker.js");
Leworker.onmessage = function(e){
alert(e.message);}
```

Web Worker

- Il est possible d'émettre un message vers un 'worker'.

```
function fctSend() {
        leworker.postMessage("message pour worker");
}
```

Web Worker

- Le worker est un script indépendant
  - Il peut s'abonner à la réception de message de son créateur.

```
function onMessage(evt){
	var readObj = JSON.parse(evt.data);
	if (readObj.code == 2){
		close(); //arret worker
	}
}

//
this.onmessage = onMessage;
```

Web Worker

- Les workers sont isolés des ressources du navigateur.
  - Uniquement l'objet "navigator" est accessible.
- Les workers ne peuvent pas accéder aux fonctions et variables de la page.
- Il y a un mécanisme d'importation de script pour les workers.
  - Attention à l'url d'accés aux scripts.

```
importScripts("sc1.js");

importScripts("sc1.js", "sc2.js");
```

Web Worker

- Il est possible d'être notifié des erreurs non traités d'un Web Worker.
- Cette notification est réalisé en utilisant la méthode **onerror** du worker

```
var leworker = new Worker("echoWorker.js");
Leworker.onerror = function(e){
alert(e.message); }
```

Outils

Editeurs

- Une liste non exhaustive d'éditeurs.
- **Javascript**
  - Netbeans.
  - Eclipse.
  - …
- **HTML/CSS**
  - Netbeans.
  - Eclipse.

Mise au point

- Outils de mise au point (Javascript/HTML/CSS/HTML5)
  - Chrome
    - Outil intégré
  - Safari
    - Outil Intégré
- FireFox
  - FireBug

Vérification HTML5

- Voici un lien pour vérifier les capacités des navigateurs pour HTML5
  - http://html5test.com/

Compresseur

- **Compresseur de JavaScript**
  - JavaScript Compressor
    - http://javascriptcompressor.com/