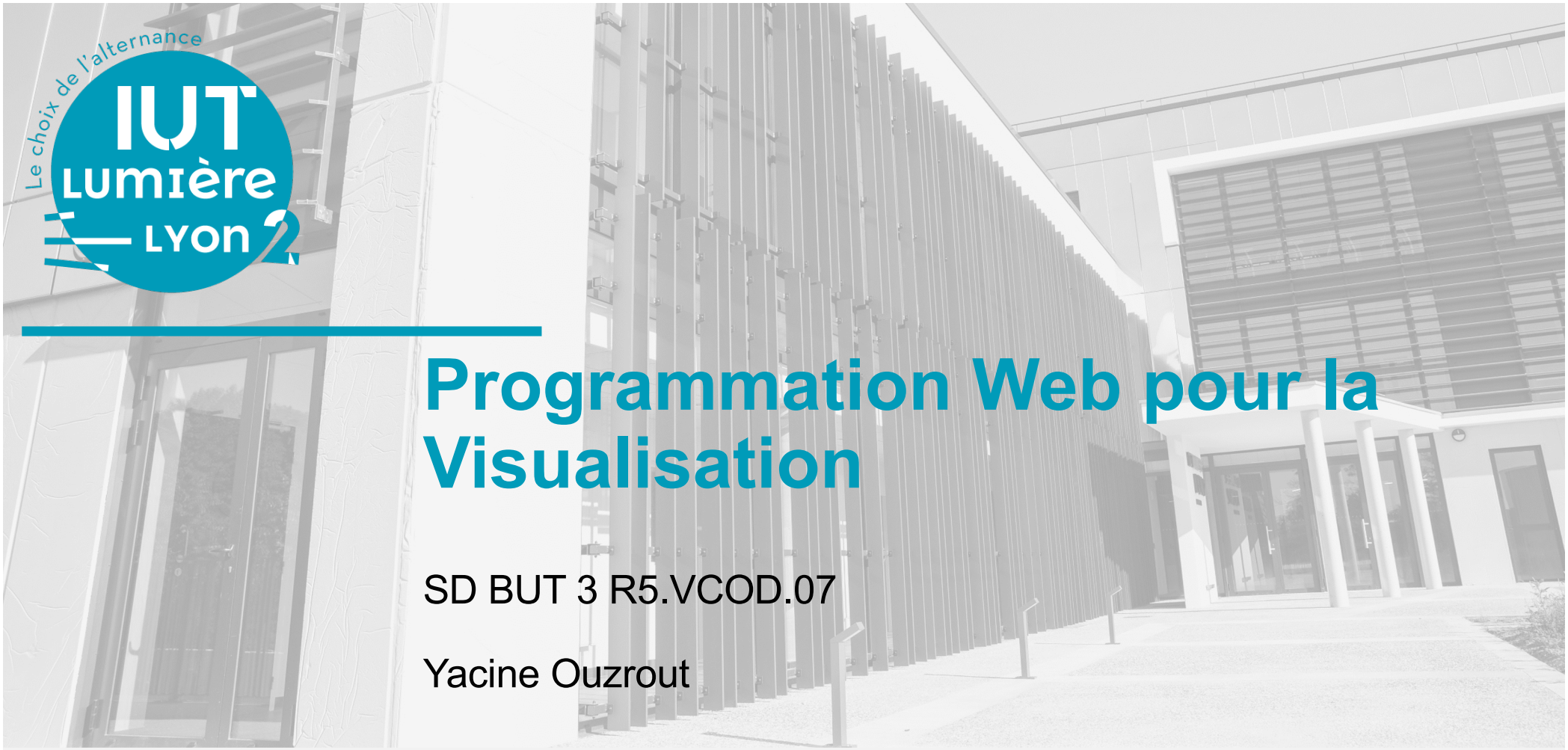

Programmation Web pour la Visualisation

SD BUT 3 R5.VCOD.07

Yacine Ouzrout

iut.univ-lyon2.fr @IUTLumiereLyon2

ORGANISATION DU COURS

7 TD et 10 TP

MCCC
- 1 TD ordinateur (0,5)
- 1 DS ordinateur (0,5)

- **Partie 1 : Approfondissement Javascript**
  - Programmation Objet en JS
  - Structuration des pages Web (DOM)
- **Partie 2 : Traitement des données Json**
  - Les formats de transfert données XML, Json...
  - Automatisation du traitement des données avec Json
  - Exploitation des données en JS
- **Partie 3 : Visualisation avec D3.js** https://observablehq.com/@d3/gallery
  - Les librairies JS pour la visualisation
  - Développement de graphiques avec D3.js
- **Partie 4 : Le javascript côté serveur**
  - Découverte de la plateforme NodeJS
  - Interaction avec une base de données
  - Développement d'une application web avec le Framework Express.js
  - API REST

PARTIE 1 : APPROFONDISSEMENT JS

Le choix de l'alternance
IUT Lumière Lyon 2

Notations JavaScript

- Similarités avec les langages
  - C, PHP, Java
- Commentaire
  - Sur une ligne : `//` … commentaire …
  - Sur plusieurs lignes : `/*` … commentaires … `*/`
- Séparateur d'instructions
  - Point-virgule instruction `;`
- Groupement d'instructions
  - Accolades `{` … instructions … `}`

Déclaration de variables

- Utilisation de l'instruction var *variable=valeur;*
  - Pas de typage (détection automatique par l'interpréteur)
  - Types atomiques : entier, réel, chaîne de caractères, booléen.
  - Nom de variable sensible à la casse.
  - Portée : déclaration en dehors de fonction ⇒ globale
    déclaration dans une fonction ⇒ locale
- const existe mais n'est pas standardisée !
  - Utiliser des majuscules

Déclaration de variables

```
<HTML>
      <HEAD> <TITLE> Exemple  </TITLE> </HEAD>
      <BODY>
      <SCRIPT LANGUAGE="JavaScript">
          var prenom_visiteur="Denis";
          var nom_visiteur="Dupont";
          var age_visiteur=29;

          // utilisation
          var accueil="Bonjour " + prenom_visiteur + " " + nom_visiteur;

          document.write(accueil);
      </SCRIPT>
      </BODY>
</HTML>
```

Exemple :

```
var total=0;
var factor=5;
var result=42;
var affiche="";
function compute(base,factor){
  result = base * factor;
  factor*=2;
  var total = result + factor;
  return total;
}
affiche+= "compute(5,4) =" + compute(5,4);
affiche+= " \ntotal = " +total+ " \n factor = " + factor + " \nresult = " +result;
alert(affiche);
```

Exemple :

```
var total=0;              // variable globale total
var factor=5;             // variable globale total
var result=42;            // variable globale result
var affiche="";           // variable globale affiche

function compute(base,factor){
        result = base * factor;  //result : 5 * 4 = 20
        factor*=2;               //factor * 2 = 8
        var total = result + factor; //variable locale total =  20 + 8 = 28
        return total;
}

affiche+= "compute(5,4) =" + compute(5,4); //affiche = "compute(5,4) = 28"
affiche+= " \ntotal = " +total+ " \n factor = " + factor + " \nresult = " +result;
//total = 0 et factor=5 et result =20"
alert(affiche);
```

Déclaration et création d'objets

- Existence d'objets prédéfinis
  - JavaScript intègre d'origine plusieurs type d'objets.
  - Déclaration : utilisation de var.
  - Création : utilisation du mot clé new, suivi du type d'objet.
- Exemple
  - Objet Date, très utile dans un environnement Internet.

```
...
// création d'un objet Date contenant la date du jour.
var date_jour = new Date();

// création d'un objet Date avec paramètres
var une_date = new Date(annee,mois-1,jour,heure,min) ;
...
```

Déclaration et création de tableaux

- Objet Array
  - Déclaration par l'utilisation de var.
  - Le premier élément du tableau est indexé à 0.
  - Il est possible de déclarer un tableau sans dimension fixée.
    La taille du tableau s'adapte en fonction du contenu
- Exemple

```
...
// création d'un tableau de 10 éléments de type
// basique (réel, entier, chaîne de caractères)
var un_tableau = new Array(10) ;

// création d'un tableau
var un_autre_tableau = new Array;
...
```

Utilisation de tableaux

- Accès aux éléments d'un tableau
  - Utilisation des crochets : [ ]

```
...
var tableau=new Array;
tableau[0]=10;
tableau[9]=5;
...
```

- Propriétés de l'objet

```
...
// obtention du nombre d'éléments de un_tableau
var dimension=tableau.length;
...
```

Tableaux associatifs

- Principe
  - L'indice est une chaîne de caractères

```
...
var tab=new Array;
tab["Lundi"]    ="semaine.html";
tab["Mardi"]    ="semaine.html ";
tab["Mercredi"] ="enfant.html";
tab["Jeudi"]    ="semaine.html";
tab["Vendredi"] ="semaine.html";
tab["Samedi"]   ="weekend.html";
tab["Dimanche"] ="weekend.html";
...
```

- Exemple
  - Chargement d'une page HTML en fonction du jour de la semaine

Tableaux d'objets

- Principe
  - Array permet de stocker des objets de n'importe quel type :
    - atomique : entier, réel, chaîne de caractères, …
    - prédéfini : Date, …
    - défini dans le code JavaScript, … cf. plus loin

- Exemple

```
var animaux=new Array;

// création de plusieurs instances d'Animal
var milou=new Animal("Milou","Chien");
var titi=new Animal("Titi","Canari");

// stockage d'instances d 'Animal dans un tableau
animaux[0]=milou;
animaux[1]=titi;
animaux[2]=new Animal("Rominet","Chat");
```

Tableaux multi-dimensionnels

- Principe
  - Array permet de stocker des objets, donc des tableaux.
- Exemple

```
...
var row0=new Array; row0[0]="O"; row0[1]="X"; row0[2]=" ";
var row1=new Array; row1[0]="X"; row1[1]="O"; row1[2]="X";
var row2=new Array; row2[0]=" "; row2[1]="0"; row2[2]="X";

var morpion=new Array;
morpion[0]=row0; morpion[1]=row1; morpion[2]=row2;
...
morpion[1][2]="X";
...
```

O	X	
X	O	X
	O	X

Exemple d'utilisation de tableaux

- Affichage de la date du jour

```
<HTML>
      <HEAD> <TITLE> Exemple Date </TITLE> </HEAD>
      <BODY>
      <SCRIPT LANGUAGE="JavaScript">
          var dt = new Date;
          var jour = dt.getDay();          // renvoi un jour [0..6]
          var numero = dt.getDate();       // renvoi le numéro dans le mois
          var mois = dt.getMonth();        // renvoi le mois [0..11]

          var tab_jour = new Array("Dimanche","Lundi","Mardi",
                 "Mercredi","Jeudi","Vendredi","Samedi");
          var tab_mois = new Array("Janvier","Février","Mars","Avril","Mai",
                 "Juin","Juillet","Août","Septembre",
                 "Octobre","Novembre","Décembre");

          document.write("Nous sommes le "+tab_jour[jour]+" "
                 +numero+" "+tab_mois[mois]);
      </SCRIPT>
      </BODY>
</HTML>
```

Nous sommes le Lundi 7 Octobre

Sortie écran

- Exemple

```
<HTML>
     <HEAD><TITLE> Exemple 1 </TITLE> </HEAD>
     <BODY>
     <SCRIPT LANGUAGE="JavaScript">
          var Les4saisons = new Array("printemps",
                              "été", "automne", "hiver");

          document.write("Voici les 4 saisons : <UL>");
          for (var i=0 ; i<4 ; i++)
          {
              document.write("<LI>", Les4saisons[i] );
          }
          document.write("</UL>");

     </SCRIPT>
     </BODY>
</HTML>
```

Voici les 4 saisons :

- printemps
- été
- automne
- hiver

Structures de contrôle

Test conditionnel : if … else …

- Permet de diriger l'exécution du script selon des conditions.
- Exemple

```
<SCRIPT language="JavaScript">
...
     if(age<18)  { alert("Vous devez être majeur");
                 }
     else        {alert( "C'est OK !");                    }
...
</SCRIPT>
```

Boucle itérative : for …

- Répéter des instructions un nombre déterminé de fois.
- Syntaxe
  - `for(initialisation ; condition ; opération )`
    `{ ... instructions ... }`
- Exemple

```
var somme=0;
for( var i=1 ; i<=10 ; i++ )
  { somme += i;  // équivalent à somme = somme +i;
  }
...
```

Pour incrémenter avec un pas de 2 ➔ i+=2

Pour décrémenter avec un pas de 2 ➔ i=-2

Boucle conditionnelle : while …

- Répéter des instructions tant qu'une condition est VRAIE.
- Syntaxe
  - `while(condition) { ... instructions ... }`
- Exemple

```
function demander()
{ var nb=0;
  while(nb<=100)
    { nb = prompt("Entrez un nombre supérieur à 100");
    }
  alert("Merci !");
}
```

Instructions Javascript

Sortie écran

- document.write(…);

Exemple

```
<HTML>
        <HEAD> <TITLE> Exemple 1 </TITLE> </HEAD>
        <BODY>
        <SCRIPT LANGUAGE="JavaScript">
                var bonjour = "Bonjour !";
                var question = "Comment allez-vous ? ";
                var phrase = bonjour + "<BR>" + question;

                document.write(phrase, "aujourd'hui ?");
        </SCRIPT>
        </BODY>
</HTML>
```

Fonctions

- Emplacement de la déclaration
  - Dans l'entête de la page
  - Utilisation de la syntaxe : function nom_fonction([param1, …])
- Corps de la fonction
  - Délimité par { … }
  - Contenu :
    - déclaration des variables locales, propres à la fonction,
    - instructions réalisées par la fonction,
    - instruction **return** pour renvoyer une valeur ou un objet
      (cette instruction n'est pas obligatoire ⇒ fonction qui ne renvoie pas de valeur)

Fonctions

- Appel de fonction
  - Peut avoir lieu à n'importe quel endroit de la page :
    - dans d'autres fonctions,
    - dans le corps de la page.
  - Utilisation de : nom_fonction([param1, …]);

```
<HTML>
   <HEAD><SCRIPT>// déclaration de fonction
         function bonjour(nom) {
            document.write("Bonjour ", nom);}
         </SCRIPT>
   </HEAD>
   <BODY>
         <SCRIPT> bonjour("M. Dupont");</SCRIPT>
  </BODY>
</HTML>
```

Fonctions

```
<HTML>
<HEAD>
<SCRIPT>// déclaration de fonctions
          function volumeSphere(rayon) {
                    return 4/3*Math.PI*Math.pow(rayon,3);
          }

          function calculerPrix(PrixUnitaire, NbArticles) {
                    return PrixUnitaire* NbArticles;
          }
</SCRIPT>
</HEAD>
<BODY>    <SCRIPT>// appels des fonctions
          document.write("Tu dois payer ",
                     calculerPrix(150,4)," Euros.<BR>");
          document.write("Le volume d'une sphère de rayon 1
                    est ", volumeSphere(1),"<BR>" );
          </SCRIPT>
</BODY>
</HTML>
```

Fonctions

Les fonctions fléchées

- Les fonctions fléchées sont une nouvelle syntaxe pour écrire des fonctions JavaScript.
- Elles font gagner du temps aux développeurs et simplifient la portée des fonctions

Fonctions classiques :

- ```
  function somme(n1,n2) {
      document.write(n1+n2);
  }
  ```

- maFonction = **function**(a, b) **{** return a+b; **}**

Fonctions fléchées :

- ```
  somme = (n1 , n2) =>
  document.write(n1 + n2);
  ```

- maFonction = (a, b) **=>** { return a+b; }

Ou

- maFonction = (a, b) **=>** a+b;

Déclenchement d'instructions JavaScript

- Programmation événementielle
  - JavaScript = langage réactif
  - L'interaction avec l'utilisateur est gérée via des événements
  - Événement = tout changement d'état du navigateur

- Production d'événement
  - Déclenché par l'utilisateur ou par le code JavaScript

Déclenchement d'instructions JavaScript

Événements JavaScript

- blur : le focus est enlevé d'un objet
- change : la valeur d 'un champ de formulaire à été modifiée par l'utilisateur
- click : un click souris est déclenché sur un objet
- focus : le focus est donné à un objet
- load : la page est chargée par le navigateur
- mouseover : la souris est déplacée sur un objet
- select : un champ de formulaire est sélectionné (par clic souris ou tabulation)
- submit : un formulaire est soumis
- unload : l'utilisateur quitte la page

Déclenchement d'instructions JavaScript

- Récupération des événements
  - Gestionnaire d'événement qui associe une action (fonction JavaScript) à la détection d'événement
- Événements détectables
  - Nom de l'événement précédé de on :
    onBlur, onChange, onClick, onFocus, onLoad,
    onMouseover, onSelect, onSubmit, onUnload
- Association événement - action
  - Dans le code HTML, identique à la déclaration d 'une propriété :

    <nom_élément attribut = propriété événement = "action" >

Déclenchement d'instructions JavaScript

```
<HTML>
     <HEAD><TITLE>Exemples de déclenchements</TITLE>
           <SCRIPT>
           function saluer() { alert("Bonjour M. Dupont!"); }
           </SCRIPT>
     </HEAD>
     <BODY>
           <H1>Exécution immédiate</H1>
           <SCRIPT> saluer(); </SCRIPT>

           <H1>Exécution sur événement onClick</H1>
           <FORM><INPUT type="button" name="Bouton"
                        value="Salut" onClick="saluer()">
           </FORM>

           <H1>Exécution sur protocole JavaScript:</H1>
           <A HREF="JavaScript:saluer()">pour saluer</A>
     </BODY>
</HTML>
```

JavaScript et fonctions mathématiques

Test de type

- Une chaîne est-elle un nombre ?
  - Utile pour la vérification de la saisie de champ de formulaire : saisie de quantités, de prix…
  - isNaN(string chaîne) renvoie :
    - TRUE si la chaîne n'est pas un nombre
    - FALSE sinon
- Exemple

```
var nombre="3.14";

if(!isNaN(nombre)) document.write(nombre, "est un nombre");
else document.write(nombre, "n'est pas un nombre");
```

```
3.14 est un nombre
```

isNaN() est une fonction qui vérifie si la valeur passée en paramètre est un nombre
isNaN veut dire is Not a Number

Conversion chaîne → nombre

Utilité

- Effectuer des opérations numériques sur des données initialement textuelles (cas des saisies de formulaire notamment)
- int parseInt(string chaîne) : conversion d'une chaîne en entier
- float parseFloat(string chaîne) : conversion d'une chaîne en réel

Exemple

```
var chaine = "3.14";
var entier = parseInt(chaine);
var reel = parseFloat(chaine);

document.write(entier);
document.write(reel);
```

```
3
3.14
```

Fonctions mathématiques

- Principe général
  - Appel des méthodes de l'objet Math

- Listes des méthodes
  - abs(x), acos(x), asin(x), atan(x), cos(x), ln(x), log(x), round(x), sin(x), sqrt(x), tan(x) : applique la fonction appropriée à x
  - ceil(x) : renvoie le plus petit entier supérieur ou égal à x
  - exp(x) : renvoie e (exponentielle) à la puissance x
  - floor(x) : renvoie le plus grand entier inférieur ou égal à x
  - max(x,y) : renvoie la plus grande des valeurs de x et y
  - min(x,y) : renvoie la plus petite des valeurs de x et y
  - pow(x,y) : renvoie x à la puissance y
  - random(x) : renvoie un nombre aléatoire compris entre 0 et 1

Fonctions mathématiques

- Exemple

```
document.write(Math.random());
document.write(Math.min(5,4));
document.write(Math.exp(1));
document.write(Math.ceil(2.2));
document.write(Math.floor(2.2));
document.write(Math.round(2.2));
document.write(Math.pow(2,3));
```

```
.8012453357071934
4
2.718281828459045
3
2
2
8
```

JavaScript et "langage à objets"

JS et Objets

- En JavaScript, la plupart des valeurs manipulées sont des objets
- Ils proviennent des fonctionnalités du langage, ex: les tableaux, ou ils sont fournis par les API du navigateur (window, document...)
- Il est possible de créer ses propres objets qui contiendront des propriétés (données ou fonctions)

Déclaration et création d'objets

- Deux types d'objets
  - Objets prédéfinis
  - Objets propres
- Création d'objets propres
  - Par appel d'une fonction qui va créer les propriétés de l'objet.
  - Utilisation de **this** pour faire référence à l'objet courant
- Exemple

```
var mon_chien=new CreerChien("Milou","Fox Terrier")

function CreerChien(le_nom,la_race)
{ this.nom=le_nom;
  this.race=la_race;
}
```

Déclaration et création d'objets

- Accès aux propriétés d'un objet
  - Utilisation de la notion pointée : objet.propriété
- Possibilité de parcourir toutes les propriétés d'un objet
  - Utilisation de la boucle : for (i in object) { ... object[i] ... }
  - i = nom de la propriété, object[i] = valeur de la propriété

- Exemple

```
document.write(mon_chien.nom);

// parcours des propriétés de l'objet navigator
var object = window.navigator;
for(i in object)
   { document.write(i+" = "+object[i]+""); }
```

Déclaration et création d'objets

```
<HTML>
  <HEAD>
  <SCRIPT>
     function CreerChien(le_nom,la_race)
     {  this.nom = le_nom;
        this.race = la_race;
     }
     var mon_chien = new CreerChien("Milou","Fox Terrier")
  </SCRIPT>
  </HEAD>
  <BODY>
  <SCRIPT>
     document.write("<b>"+ mon_chien.nom +" </b><br>");
           </SCRIPT>
  </BODY>
</HTML>
```

Déclaration et création d'objets

- Déclaration de méthodes
  - Association de fonctions dans la création de l'objet.
- Exemple

```
function CreerChien(le_nom,la_race)
{ this.nom=le_nom;
  this.race=la_race;
  this.Afficher=AfficherChien;
}

function AfficherChien()
{ document.write("Ce chien s'appelle "+this.nom +". C'est un "+this.race+".");
}

var mon_chien = new CreerChien("Milou","Fox Terrier")
document.write(mon_chien.Afficher());
```

Déclaration et création d'objets

- Alternative pour la déclaration de méthodes
  - Méthode générique, déclenchable sur un objet quelconque.
- Exemple

```
   function AfficherChien()
   {
     with(this)
     {
       document.write("Ce chien s'appelle "+this.nom
                      +". C'est un "+this.race+".");

     }
}
```