Université Sidi Mohamed Ben Abdllah

Faculté des sciences Dhar El Mahraz
Licence professionnelle SIGL

WEB DYNAMIQUE

JAVASCRIPT

RÉALISÉ PAR: PR. MAHRAZ MED ADNANE

ANNÉE UNIVERSITAIRE:2017/2018

INTRODUCTION

- JavaScript permet de dynamiser un site Web.
- Code JavaScript intégré aux pages HTML.
- Code interprété par le navigateur client ≠ code PHP (interprété du coté serveur).
- JavaScript est un langage événementiel (association d'actions aux événements déclenchés par l'utilisateur (passage de souris, clic, saisie clavier, etc...).

INTÉRÊTS DE JAVASCRIPT ?

- Supporté par les principaux navigateurs, c.-à-d., il ne nécessite pas de plug-in particulier.
- Accès aux objets contenus dans un document HTML
- Possibilité de mettre en place des animations sans l'inconvénient des longs temps de chargement nécessités par les données multimédia.
- Langage relativement sécurisé : il est impossible de lire ou d'écrire sur le disque client (impossibilité de récupérer un virus par ce biais).

INTÉGRATION DE JAVASCRIPT DANS HTML

Il existe **2 manières** pour insérer un code JavaScript dans une page HTML:

- JavaScript dans HTML

```
<html>
    <head>
        <title>Page HTML</title>
    </head>
    <body>
        <script>
            alert('bonjour');
        </script>
    </body>
</html>
```

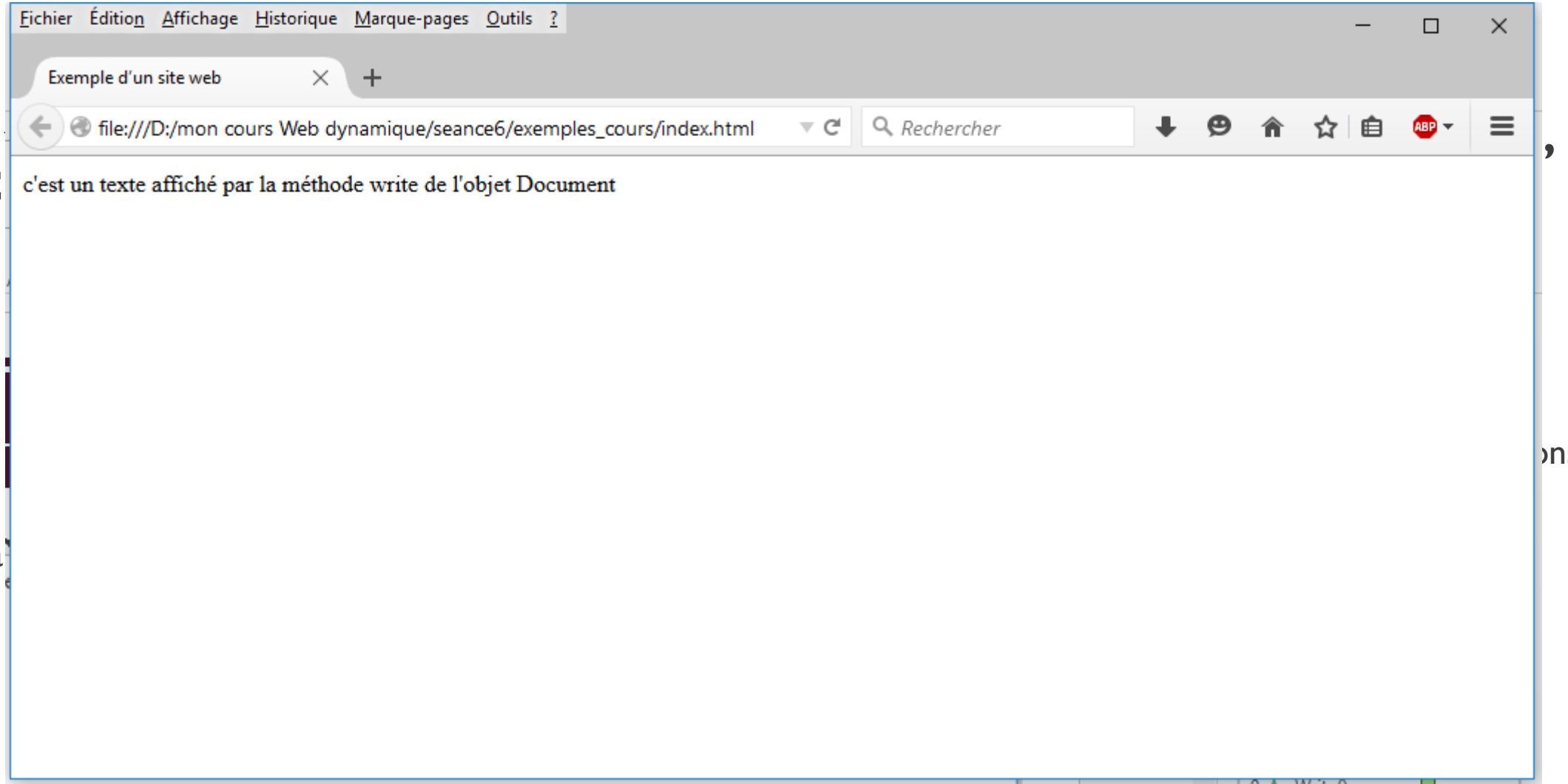
- JavaScript à l'extérieur du HTML

```
<html>
    <head>
        <title>Page HTML</title>
        <script src="monScript.js">
        </script>
    </head>
    <body>

    </body>
</html>
```

ENTRÉE ET SORTIE DE DONNÉES AVEC JAVASCRIPT

- 3
  C
  -
  -
  -
- La



ENTRÉE ET SORTIE DE DONNÉES AVEC JAVASCRIPT

```
<html>
   <head>
      <title> une page simple </title>
   </head>
   <body>
      Bonjour
      <script language='javascript'>
      alert('bonjour');
      document.write (prompt('quel est votre nom ?','Indiquer votre nom
      ici'));
      confirm('quel bouton allez-vous choisir ?');
      </script>
   </body>
</html>
```

DÉCLARATION DE VARIABLES

- Utilisation de l'instruction var *variable=valeur;*
  - Pas de typage (détection automatique par l'interpréteur)
  - Nom de variable **sensible à la casse**.
  - Portée :
    - déclaration en dehors de fonction ⇒ globale
    - déclaration dans une fonction ⇒ locale

DÉCLARATION DE VARIABLES

```
<html>
<head>
     <title>Exemple</title>
</head>
<body>
     <script>
          var nom;
          var texte;
          nom=prompt("quel est votre nom");
          texte="je connais votre nom, c'est "+"<g><i>"+nom+"</g></i>";
          document.write(texte);
     </script>
</body>
</html>
```

STRUCTURES DE CONTRÔLE

Test conditionnel : if … else …

```
<html>
<head>
     <title>Exemple</title>
</head>
<body>
     <script>
                    var age=prompt("quel est votre age");
                    if (age>=18){
                      alert("vous etes Majeur...");
                    }else{
                      alert("vous etes Mineur...");
                    }
     </script>
</body>
</html>
```

STRUCTURES DE CONTRÔLE

- Boucle itérative :

```
<html>
<head>
     <title>Exemple</title>
</head>
<body>
     <script>
                    var nb=prompt("Donnez un nombre");
                    var somme=0;
                    for(var i=1;i<=nb;i++){
                       somme+=i;
                    }
                    alert("La somme des nombres entre 0 est "+ nb +" est "+somme);
     </script>
</body>
</html>
```

STRUCTURES DE CONTRÔLE

- Boucle conditionnelle

```
while(condition) { ... instructions ... }
```

```
<html>
      <head>
                  <title>Exemple</title>
      </head>
      <body>
                  <script>
                     var nb=prompt("Donnez un nombre");
                     var somme=0, i=0;
                     while(i<=nb){
                           somme+=i; i++;
                     }
                     alert("la somme des nombres entre 0 est "+nb+" est "+somme);
                  </script>
      </body>
</html>
```

FONCTIONS

- syntaxe :

```
function nom_fonction ([param1, …]){
        //Corps de la fonction
}
```

- Corps de la fonction
  - Déclaration des variables locales, propres à la fonction,
  - Instructions réalisés par la fonction,
  - Instruction **return** pour renvoyer une valeur ou un objet (Facultative)

FONCTIONS ET PORTÉE DES VARIABLES

- La portée d'une variable déclarée dépend de l'endroit où elle est déclarée :
  - VARIABLE GLOBALE: déclarée en dehors de la fonction.
  - VARIABLE LOCALE: déclarée à l'intérieur d'une fonction aura une portée limitée à cette seule fonction.

```
<script>
     var nom="Mohamed"; //variable globale
     function afficher(){
          alert("Votre nom est: "+nom);
     }
     alert("Votre nom est: "+nom);
</script>
```

```
<script>
    var nom="Mohamed"; //variable globale
    function saisir(){
        var nom; //variable locale
        nom=prompt("Quel est votre nom:");
        Window.alert(window.nom);
        return nom;
    }
    alert("Votre nom est: "+saisir());
    alert("Votre nom est: "+nom);
</script>
```

FONCTIONS ANONYMES

- syntaxe :

```
function ([param1, …]){
    //Corps de la fonction
}
```

```
Var x= function ([param1, …]){
    //Corps de la fonction
}
```

- On peut attribuer cette fonction à une variable.
- Pour que cette fonction s'appelle automatiquement:

```
(function(){//Corps de la fonction})();
```

- Isoler les variables déclarées au sein de la fonction au monde extérieur.

TABLEAU DE DONNÉES (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: Sa taille s'adapte en fonction du contenu.

```
   // création implicite d'un tableau
var mon_tableau = ["Ali", 'Mohamed', "Sarah", 10, 6];

   // création d'un tableau de 10 éléments
var mon_tableau = Array(10);

   // création d'un tableau avec l'opérateur « new »
var mon_tableau = new Array(10);
var mon_tableau = new Array();
```

UTILISATION DE TABLEAUX

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

```
var tableau=new Array();
tableau[0]=10;
tableau[1]=5;
```

- La propriété Length

```
tableau.length
```

- Parcourir un tableau

```
// Parcourir un tableau sans connaître le nombre
d'éléments
var tableau= new Array(1, "a", 9) ;
tableau[200] = 12 ;
for (var i in tableau)
            alert("tableau[" + i + "] = "+tableau[i]);
```

TABLEAUX ASSOCIATIFS

L'indice est une chaîne de caractères

```
var tab=new Array();
tab["nom"]  ="Ben ali";
tab["prenom"]  ="Mohamed";
tab["age"]  =25;
tab["adresse"]  ="Fes";
...
```

```
...
alert("Votre nom est: "+tab["nom"]);
...
```

La propriété Length de l'objet Array() pour ce genre de tableau ne fonctionne pas.

TABLEAUX MULTI-DIMENSIONNELS

Array permet de stocker des objets, donc des tableaux.

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

O	X	
X	O	X
	O	X

DÉCLARATION ET CRÉATION D'OBJETS

- Deux types d'objets
  - Objets prédéfinis
  - Objets propres
- Création d'objets avec des initialiseurs d'objets (objets littéraux).

```
var obj = {
   propriété_1: valeur_1,
   propriété_2: valeur_2,
   …
   propriété_n: valeur_n
};
```

```
var obj = new Object();
obj.popriété_1=valeur_1;
obj.propriété_2=valeur_2,
   …
obj.propriété_n= valeur_n;
```

DÉCLARATION ET CRÉATION D'OBJETS

- Les objets sont créés de la même façon qu'avec new Object().
- les objets créés à partir d'une expression littérale seront des instances d'Object.
- Les objets peuvent également être créés en utilisant la méthode Object.create().

```
var Animal = {
type: "Invertébrés", // Valeur par défaut
afficherType : function() { // Une méthode pour afficher le type Animal
console.log(this.type); }
}
// On crée un nouveau type d'animal, animal1
var animal1 = Object.create(Animal);
animal1.afficherType(); // affichera Invertébrés
// On crée un nouveau type d'animal, animal2
var animal2 = Object.create(Animal);
Anim12["type"]= "poisson";
animal2.afficherType(); // affichera poisson
```

DÉCLARATION ET CRÉATION D'OBJETS

Création d'objets propres en utilisant un constructeur

- Par appel d'une fonction qui va créer les propriétés de l'objet.
- Utilisation de **this** pour faire référence à l'objet courant
- On crée une instance de l'objet avec **new**.

```
var Etudiant=new Etudiant("Mohamed", "Ben Ali", "1298742046");
function Etudiant(Le_nom,Le_prenom,Le_CNE){
            this.nom=Le_nom;
            this.prenom=Le_prenom;
            this.CNE=Le_CNE;
}
alert("Votre nom est: "+Etudiant.prenom);
```

DÉCLARATION ET CRÉATION D'OBJETS

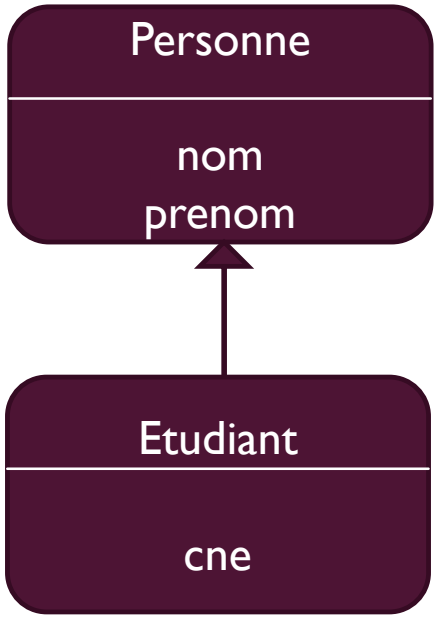
- Déclaration de méthodes
  - Association de fonctions dans la création de l'objet.

```
function Etudiant(Le_nom,Le_prenom,Le_CNE){
            this.nom=Le_nom;
            this.prenom=Le_prenom;
            this.CNE=Le_CNE;
            this.afficher=affiche_Etudiant;
}
function affiche_Etudiant(){
            document.write("Votre nom et prénom est: "+ Etudiant.nom+"
            "+Etudiant.prenom+"<br/>" );
            document.write("Votre CNE est: "+ Etudiant.CNE );
}
var Etudiant=new Etudiant("Mohamed", "Ben Ali", "1298742046");
Etudiant.afficher();
```

LE MODÈLE OBJET JAVASCRIPT

- JavaScript est un langage objet basé sur des prototypes et non pas sur des classes.
- JavaScript possède que des objets.
- Les objets *prototypiques* agissent comme un modèle sur lequel on pourrait obtenir des propriétés initiales pour un nouvel objet.
- Un objet peut être associé comme le *prototype* d'un autre objet (le second objet partage les propriétés du premier).

LE MODÈLE OBJET JAVASCRIPT




```
function Personne(){
        this.nom="Mahraz";
        this.prenom="Mohamed";
}
function Etudiant(){
        this.CNE="22222222";
}
```

```
Etudiant.prototype=new Personne()
Var p1=new Personne();
var e1=new Etudiant();
e1.__proto__.nom="toto";
Etudiant.prototype.nom="dodo";
```

On peut modifier la propriété nom du constructeur Etudiant pour que les objets instanciés changent leurs valeurs