

Technologies et Développement Web

Cours 4 – JavaScript

Objectifs du module

- Faire un tour d'horizon du langage CSS, ses propriétés et ses règles
- Appliquer CSS sur des pages web

Plan du module

- La Structure
- Le Texte
- Les Listes
- Les Liens
- Les Images
- Les Tableaux
- Les Formulaires
- Organisation
- Audio et Vidéo
- Nouveautés HTML 5

Section 1 : Introduction

Pourquoi JavaScript ?

- HTML est un langage statique
- Les capacités dynamiques de CSS sont très limitées
- Ne répondent pas à certains besoins : répondre aux interactions, traitement, boucles,...
- JavaScript a été proposé dans ce contexte
- JS est un langage interprété par le navigateur
- JS a été créé par Netscape Communications

Que peut faire JS ?

- Afficher des messages
- Valider les formulaires
- Faire des calculs
- Faire des animations
- Déetecter le navigateur et ses capacités
- Interagir avec les serveur à travers les services web et AJAX

Intégrer du JS dans une page web

- JS script est intégré à travers la balise «script »
- Le script peut être à l'intérieur du corps de la page, dans l'entête ou dans un fichier externe.
- Les fichiers JavaScript ont généralement l'extension « js »

Déclarations JavaScript

```
<!-- script interne -->
<script type="text/javascript">
    function appelScriptInterne() {
        alert('salam interne');
    }
</script>
<!-- script externe -->
<script src="ScriptIntro.js"></script>
```

Introduction au DOM

- JS interagit avec la page à travers une structure appelée « DOM » accessible via une variable appelée « document »
- À travers le DOM, on peut accéder à un ou plusieurs éléments. Pour accéder à un élément par son « id », utiliser « GetElementById »
- Si un élément HTML a un « id », alors, une variable est automatiquement créée pour lui
- Voir « DomIntro.HTML »

Section 2 : Syntaxe

Instructions

- La syntaxe JavaScript ressemble à la famille C, C++, Java ou C# avec des spécificités
- Un script JS est composé de plusieurs instructions
- Les instructions peuvent être des **déclarations**, des **opérations** ou ;
- Les instructions se terminent par « ; »
- Une instruction peut se trouver sur plusieurs lignes

Variables

- Les variables sont déclarées avec le mot clé **var**
- Le type est automatiquement déduit par l'interpréteur
- Quand c'est possible, le moteur JS effectue automatiquement les conversions
- Pour convertir une variable, utiliser parseInt et parseFloat

Fonctions

- Les fonctions sont déclarées avec le mot clé **function**
- Les instructions sont délimitées par des accolades
- Le type de retour est automatiquement déduit par le mot clé « **return** » sinon pas de type de retour
- Pour passer l'instance de l'objet concerné, utiliser « **this** »

Commentaires

- JS peut inclure des commentaires sur une seule ligne ou sur plusieurs lignes
- Les commentaires sur une lignes sont déclarés avec « `//` »
- Les commentaires sur plusieurs lignes se trouvent entre « `/*` » et « `*/` »
- Voir « [Syntax.html](#) »

Exemple

```
<script type="text/javascript">

    function somme(x, y) {
        return x + y;
    }

    /* cette fonction sert à faire l'addition de deux
    nombres */

    function calculer() {
        var nbr1 = parseInt(document.getElementById('txt1').value); // récupérer la valeur et la
convertir en int
        var nbr2 = parseInt(document.getElementById('txt2').value);
        var resultat = somme(nbr1, nbr2);
        alert(resultat);
    } </script>
```

Section 3 : Fenêtre et DOM

Introduction

- La fenêtre est composée de trois objets qui peuvent être accédés directement : « location », « history » et « document »



Historique (history)

- Permet d'accéder à l'historique du navigateur
- Permet de personnaliser l'accès à l'historique dans son site web sans utiliser les boutons du navigateur

history (fonctions principales)

Fonction	Description
back()	Accède à l'URL précédente
forward()	Accède à l'URL suivante
go()	Accès à l'URL par delta. go(-2) est équivalent à deux clics sur le bouton précédent

Location (location)

- Donne des informations sur le site sur lequel pointe la fenêtre
- Donne des informations telles que le port, le protocole, ...
- Possède des méthodes qui permettent de naviguer vers certaines urls

Location (fonctions et propriétés principales)

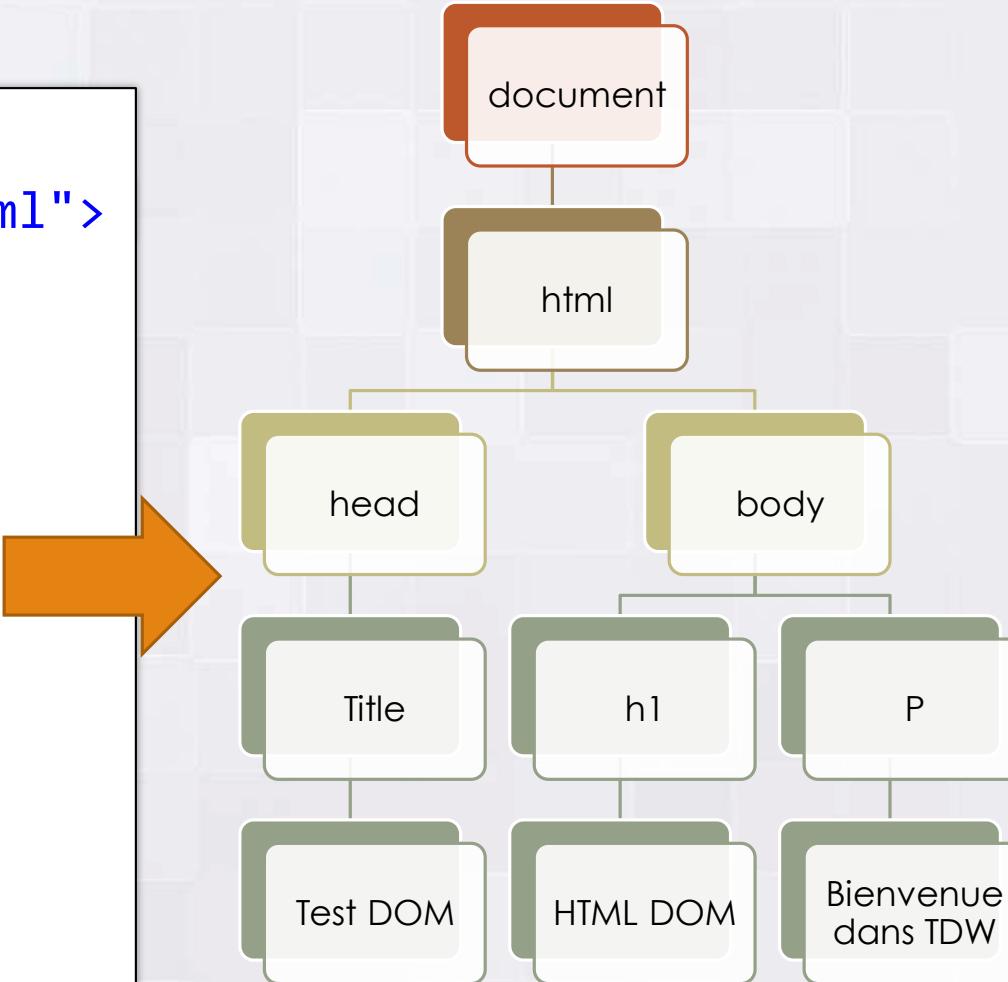
Fonction	Description
host	Hôte
href	Adresse en cours
protocol	Protocole
search	Accède aux paramètres de l'url
reload()	Recharge la page
assign(url)	Navigue vers une nouvelle url

DOM (Document Object Model)

- Un document HTML est structuré sous forme d'une arborescence appelée DOM
- Chaque élément de cette arborescence est appelé « **nœud** »
- Quand un nœud est un sous-élément d'un autre nœud, il est appelé « **enfant** » et son conteneur « **parent** »
- Deux nœuds de même niveaux sont appelés « **frères** » (siblings)
- Le texte est une forme de nœud appelé « **texte simple** »

DOM - Exemple

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Test DOM</title>
</head>
<body>
    <h1>HTML DOM</h1>
    <p>Bienvenue dans TDW</p>
</body>
</html>
```



Propriétés / Méthodes des nœuds

Fonction	Description
nodeName	Nom du nœud
nodeType	1 balise normale, 3 texte simple, 9 document
nodeValue	Valeur d'un nœud texte
innerHTML	HTML à l'intérieur du nœud
innerText	Texte brut à l'intérieur du noeud
id	Id de l'élément
className	Nom de la classe de l'élément
firstChild	Premier enfant
lastChild	Dernier enfant
style	Accède au style de l'enfant

Propriétés / Méthodes des nœuds - suite

Fonction	Description
childNodes	Tableau contenant tous les enfants
previousSibling	Frère précédent
nextSibling	Frère suivant
appendChild(elt)	Ajoute un nouvel enfant au parent
insertBefore(elt, eltAvant)	Insère un élément avant un autre élément
replaceChild(nouveau, ancien)	Remplace un élément par un nouvel élément
removeChild(enfant)	Supprime l'enfant
hasChildren()	Indique si l'élément possède des enfants ou non
cloneNode()	Clone un élément
attributes	Tableau des attributs de l'élément

Style d'un noeud

- Toutes les propriétés CSS sont accessibles via la propriété « **style** » d'un nœud
- Par exemple : `noeud.style.backgroundColor = 'yellow';`

Propriétés / Méthodes de l'objet document

Fonction	Description
getElementById(id)	Renvoie un élément par son id
getElementsByTagName(balise)	Renvoie la liste des éléments du document dont la balise est « balise »
createTextNode(texte)	Crée une balise texte
createElement(balise)	Crée une balise HTML
getElementsByClassName(className)	Renvoie la liste des éléments par nom de classe

Démo : FenetreDOM.HTML

<http://localhost:11875/JS/> [aller à l'url](#)

Hello World !

[InnerHTML](#) [InnerText](#)

[retour](#)



Section 4 : Variables et Types

Déclaration de variables

- Les variables sont déclarées par le mot clé « var »
- Il existe deux types de variables : globales (déclarées dans tout le script) et locales (déclarées dans une fonction)
- « var » est optionnel pour les variables globales (la déclaration se fait par une affectation mais la déclaration en utilisant « var » est une bonne pratique pour la lisibilité du script)

Opérateurs

Opérateur	Description
+	Concaténation de chaînes de caractères
+	Addition
-	Soustraction
*	Multiplication
/	Division
%	Modulo
++	Incrémantion
--	Décrémantion
=	Affectation

Opérateurs logiques

Opérateur	Description
<code>==</code>	Comparaison
<code><</code>	Inférieur
<code>></code>	Supérieur
<code><=</code>	Inférieur ou égal
<code>>=</code>	Supérieur ou égal
<code>!=</code>	Different
<code>&&</code>	Et logique
<code> </code>	Ou logique
<code>!</code>	Négation

Précédence des opérateurs

- Les opérateurs en JS sont les mêmes que Java, C ou C++
- Pour forcer la précédence ou rendre le code plus lisible utiliser les parenthèses
- Par exemple : `var x = (a + b) * c;`

Types

- Le type ne se déduit pas à la déclaration mais l'affectation
- Rien n'empêche une variable de changer de type (contrairement aux autres langages)
- Quand JS le peut, la conversion entre types est automatique
- Pour forcer la conversion, utiliser ***parseInt*** ou ***parseFloat***

Types de base dans JS

Type	Description	Exemples
Nombre	Entier ou réel	2, 88 ou -15.23
Booléen	Vrai ou faux	true ou false
Chaînes de caractère	Texte	« Bonjour » ou 'au revoir'
null	Variable indéfinie	

Chaines de caractères

- Les chaines de caractères peuvent être déclarées directement :
- var s = « Bonjour » ou var t = 'bonjour'
- L'opérateur new peut être utilisé
- var s = new String('bonjour')
- Une chaine de caractères est un

Propriétés / Fonctions des chaînes de caractères

Type	Description
length	Longueur de la chaine
toUpperCase()	Convertit en majuscules
toLowerCase()	Convertit en minuscules
substring(départ,fin)	Extrait une sous-chaîne de caractère à partir d'une autre
charAt(indice)	Renvoie le caractère à l'indice donné
indexOf(sousChaine)	Cherche une sous-chaine dans une chaine
split(séparateur, max_longueur)	Fractionne une chaine de caractères en un tableau de caractères

Tableaux

- Il existe plusieurs façons de déclarer des tableaux
- `var tableau = new Array(3);` → déclare un tableau de 3 éléments
- `var tableau = new Array(11,22,33);` → déclare et initialise un tableau
- `var tableau = [11,22,33];` → déclare et initialise un tableau

Propriétés / Fonctions des tableaux

Type	Description
length	Longueur du tableau
indexOf	L'indice d'un élément, -1 si pas trouvé
reverse()	Inverse le tableau
sort(callback)	Trie le tableau. Pour les tableaux numériques, il faut fournir une fonction de tri
foreach(callback)	Exécute la fonction callback pour chaque élément du tableau

Démonstration (Variables.html)

19	1919 57 1938
df fg gfg	9 fg DF FG GFG
Tri	
<input type="text"/>	
Tri 2	
9,11,19	

Section 5 : Fonctions et Objets

Fonctions

- Une fonction est déclarée par le mot clé « **function** »
- Une fonction peut avoir 0,1 ou plusieurs **paramètres**
- Si une fonction renvoie un résultat, on utilise le mot clé « **return** »
- L'implémentation de la fonction se fait en mettant les instructions entre accolades

Appel de Fonctions

- L'appel d'une fonction se fait en utilisant son nom et les paramètres entre parenthèses
- Si un paramètre n'est pas fournir, l'interpréteur lui donne la valeur « `undefined` »
- Utiliser le paramètre `||` ou affecter des valeurs par défaut

Exemple

```
function calcul(liste, x, y) {  
    var elt = document.createElement('option');  
    x = x || 25; // si x n'est pas fourni, prendre 25 comme va par défaut  
    y = y || 30; // si y n'est pas fourni, prendre 30 comme val par défaut  
    elt.text = x + y;  
    elt.value = x + y;  
    liste.add(elt);  
}  
  
function testAppels() {  
    var liste = document.getElementById('liste');  
    calcul(liste, 11, 12); // fournit tous les paramètres  
    calcul(liste, 11); // 1er par par défaut  
    calcul(liste); // prend les deux paramètre par défaut  
    calcul(liste, 1, 2, 6); // dernier appel ignoré  
}
```

Objets

- Tout dans JavaScript est objet : nombre, chaîne et tout autre élément
- Quoique l'objet est tout le temps utilisé, JS n'est pas orienté objet, la notion de classe n'existe pas
- Un objet est caractérisé par un état représenté par ses propriétés
- Un objet est caractérisé par son comportement représenté par ses méthodes

Déclaration d'objets

- JS permet de déclarer les objets de 3 manières
- Une déclaration en utilisant « new Object() »
- Une déclaration directe
- Une déclaration en utilisant les fonctions

Déclaration des objets avec l'opérateur « new »

- Créer une instance d'un objet en utilisant « new Object »
- Affecter les propriétés de l'objet
- Affecter les méthodes de l'objet

Déclaration des objets avec l'opérateur « new », Exemple

```
var module = new Object();

    module.Nom = « ig1 »;
    module.Coefficient = 5;
    module.credits = function () {
        return this.Coefficient * 15;
    };
}
```

Déclaration directe des objets

- JS permet de déclarer directement des objets sans passer par l'opérateur « new »
- Cette syntaxe décrit un objet sous forme d'accolades, de propriétés et de méthodes

Déclaration directe des objets, Exemple

```
var employe = {  
    nom: « mohammed amine »,  
    prenom: « mostefai »,  
    echelon: 1,  
    salaire: function (section) {  
        return this.echelon * section * 15000;  
    }  
};
```

Déclaration des objets en utilisant les fonctions

- Une fonction permet de déclarer un objet
- L'objet peut ensuite être directement instancié en utilisant l'opérateur « new »

Déclaration des objets en utilisant les fonctions, exemple

```
function Etudiant(nom, nigl, ntdw) {  
    this.nom = nom;  
    this.noteIGL = nigl;  
    this.noteTDW = ntdw;  
    this.Moyenne = function () {  
        return (this.noteIGL + this.noteTDW) / 2;  
    };  
}  
  
var e = new Etudiant("amine", 15, 14);
```

Utilisation des objets

- Pour accéder à une propriété ou une méthode d'un objet, utiliser le nom de l'objet et le nom de la propriété / méthodes séparés par « . »
- Le mot clé « **with** » permet de simplifier l'utilisation d'un objet en accéder directement à ses propriétés

Utilisation des objets - exemple

```
var e = new Etudiant("amine", 15, 14);  
e.noteIGL = 16;  
alert(e.Moyenne());  
with (e) {  
    noteIGL = 11;  
    noteTDW = 19.5;  
    alert(Moyenne());  
}
```

Extension d'un objet

- JS permet d'étendre des définitions d'objets ou des types déjà existants en ajoutant de nouvelles propriétés ou de nouvelles méthodes
- Pour étendre un objet, utiliser le mot clé « **prototype** »

Extension d'un objet - Exemple

```
function testerExtension() {  
    var s = new String('tdw');  
    alert(s.Doubler());  
}  
  
String.prototype.Doubler = function () {  
    return this + this;  
}
```

Objets prédéfinis

- L'objet « Math » fournit des utilitaires mathématiques
- L'objet « String » gère les chaînes de caractères
- L'objet « Date » gère la date et l'heure

L'objet « Math », méthodes principales

Fonction	Description
round(x), ceil(x) ou floor(x)	Arrondit x au plus proche, plus proche supérieur ou plus proche inférieur respectivement
random()	Génère un nombre aléatoire de 0 à 1
sin(x), cos(x)	Sinus, cosinus
abs(x)	Valeur absolue de x
log(x)	Logarithme
pow(x,y)	x puissance y
sqrt(x)	Racine de x

L'objet « Date », méthodes principales

Fonction	Description
getDay(), getMonth(), getFullYear()...	Renvoie des informations partielles telles que le jour, le mois ou l'année
getTime	Renvoie le nombre de millisecondes depuis la date en question et le 1 ^{er} Janvier 1970
toLocaleDateString()	Convertit en chaîne de caractères selon les options régionales
toTimeString()	Extrait la portion de temps à partir de la date

Démonstrations (FonctionsObjets.html)

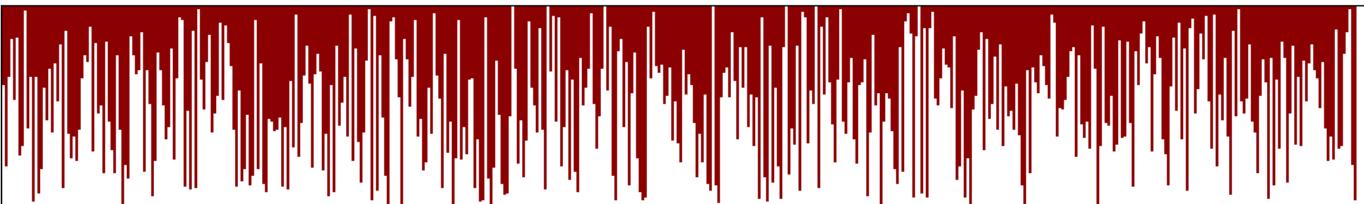
Déclaration et utilisation d'objets

[Tester Module](#)
[Tester Employé](#)
[Tester Etudiant](#)

Extension d'un objet

[Tester Extension](#)

Nombres aléatoires



[Générer barres](#)

Gestion des dates

12/05/2013 [extraireDate](#)

Section 6 : Blocs de contrôle

L'instruction conditionnelle

- L'instruction conditionnelle est « if »
- La condition est une expression booléenne entre parenthèse
- Un bloc « else » exprime le sinon. Il est facultatif.

Syntaxe

```
if (condition)
{
    I1();
    I2();
}
else
{
    I3();
    I4();
}
```

Exemple

```
if (v % 2 == 0)
    alert('nombre pair');

else
    alert('nombre impair');
```

Conditions multiples

- Lorsque plusieurs conditions sont requises, il est nécessaire de faire plusieurs combinaisons de if/ else qui réduit la lisibilité du code.
- JS dispose de l'instruction « switch » qui permet de simplifier le multi-branchement conditionnel

Syntaxe switch

```
switch (expression) {  
    case val1: traitement1();  
        break;  
    case val2: traitement2();  
        break;  
    case val3: traitement3();  
        break;  
    ...  
    default:  
        traitementAlternatif();  
}
```

Exemple switch

```
switch (v.length) {  
    case 0: alert('vide !');  
        break;  
    case 1: alert('un seul caractère');  
        break;  
    case 2: alert('caractère double');  
        break;  
    default:  
        alert('plus que deux !');  
}
```

Boucles for

- Une boucle « **for** » est utilisée lorsque le nombre d'itérations est à priori connu
- Elle est composée de trois compartiments : déclaration et valeur initiale, condition et instruction d'incrément

Syntaxe for

```
for (val_init; condition; increment)  
{  
    traitement();  
}
```

Exemple for

```
for (i = 0; i < 10; i++)  
    v = v + getValeur();
```

Boucles while et boucles do/while

- Une boucle « **while** » est utilisée lorsqu'on ne connaît pas à priori le nombre d'itérations. Elle est définie par une seule condition.
- La boucle « **do/while** » est similaire à while sauf que les instructions sont exécutée au moins une fois.

Syntaxe while

```
while (condition)
{
    traitement();
}
```

Syntaxe do/while

```
do {  
    traitements();  
} while (condition);
```

Exemple while

```
while ((somme < 23) && (i < tableau.length))  
    somme = somme + getValeur(tableau[i]);
```

Instructions spéciales « boucles »

- L'instruction « **break** » permet de sortir d'une boucle avant que la condition de terminaison ne soit satisfaite.
- L'instruction « **continue** » permet d'interrompre l'exécution de la boucle pour l'incrément en cours et passe au prochain incrément

Parcourir un objet

- L'instruction « **for ..in** » permet de parcourir un objet en obtenant toutes les propriétés de cet objet
- Pour obtenir la valeur d'un objet, utiliser l'opérateur d'indexation

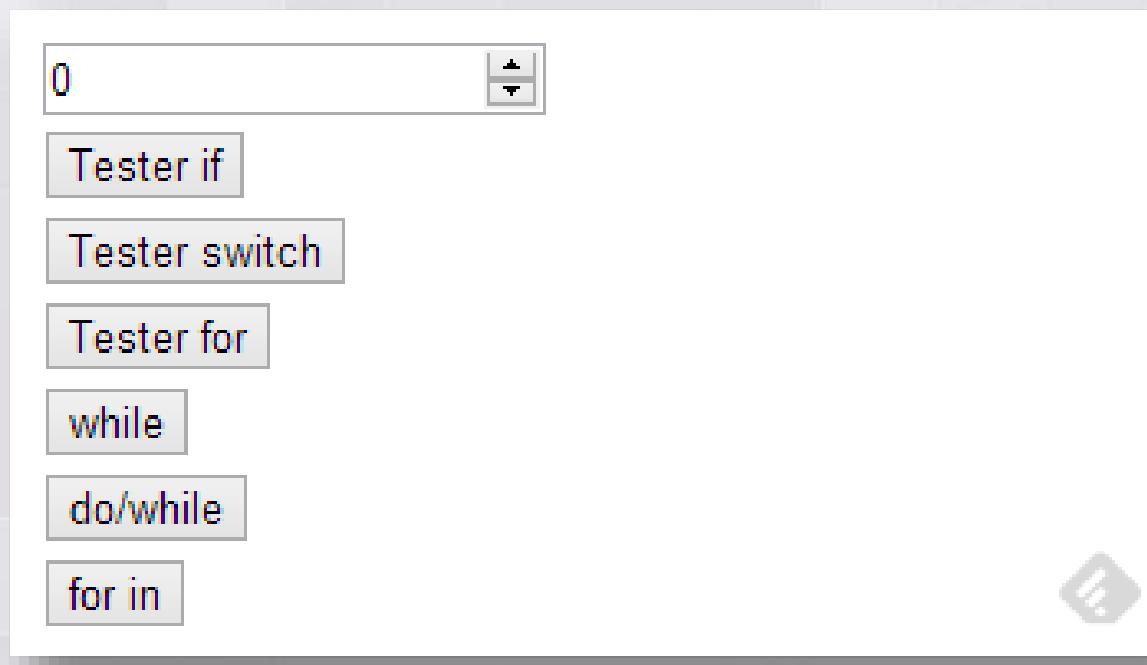
for..in Syntaxe

```
for (variable in objet) {  
    traitement();  
}
```

Exemple for..in

```
for (x in etudiant) {  
    alert(x);  
    alert(etudiant[x]);  
}
```

Démonstration – Blocs.html



Section 7 : Evènements et interactions

Introduction aux évènements

- Les évènements représentent des fonctions appelées en réaction à une action de l'utilisateur : souris, clavier,....
- La fonction répondant à un évènement est appelée « gestionnaire d'évènement »
- Les évènements sont par convention préfixés par « on ». Par exemple : « onclick »

Affectation d'un gestionnaire d'évènement

- L'affectation d'un gestionnaire d'évènement à un objet se fait en renseignant certains attributs de cet objet correspondant à l'évènement
- La valeur de l'attribut correspond à l'appel d'une fonction JS

Affectation de Gestionnaire - Exemple

```
<div class="vignette" onmouseover="survoler(this)"  
onmouseout="sortir(this)">Survolez-moi</div>
```

Affectation dynamique de gestionnaire d'évènements

- Les gestionnaires d'évènement peuvent être affectés dynamiquement (par code)
- Ceci se fait simplement en affectant une fonction au gestionnaire en utilisant l'opérateur d'affectation

Affectation dynamique de Gestionnaire - Exemple

```
function survoler(objet) {
    objet.style.backgroundColor = 'yellow';
}
function sortir(objet) {
    objet.style.backgroundColor = 'lightgreen';
}
function click() {
    alert('la div a été cliquée');
}
function affecterEvenement() {
    var objet = document.getElementById('myDiv');
    objet.onclick = click;
    objet.onmouseover = function (e) { survoler(e.target)};
    objet.onmouseleave = function (e) { sortir(e.target); }
}
```

L'objet « évènement »

- Lorsqu'un évènement a lieu, un objet de type « **event** » est passé au gestionnaire
- L'objet contient des propriétés relative au contexte de l'évènement (par exemple coordonnées)

L'objet « event » - Propriétés

Propriété	Description
modifiers	Touches spéciales (Par exemple Ctrl ou Alt)
button	Bouton gauche ou droit de la souris (0 ou 2)
pageX, pageY	Coordonnées
cancelable	Indique si l'évènement peut être annulé
target	Objet relatif à l'évènement
charCode	Relatif aux évènements clavier, code caractère
keyCode	Relatif aux évènements clavier, code touche
altKey	Indique si la touche « alt » est pressée
ctrlKey	Indique si la touche « ctrl » est touchée

L'objet « event » - Méthodes

Méthode	Description
preventDefault()	Annule l'évènement s'il est annulable

Evènements de la souris

Evènement	Description
onclick	Clic simple de la souris
ondblclick	Clic double
onmousedown	Clic de la souris sur un élément
onmouseup	Quand l'utilisateur relache la souris
onmousemove	Quand la souris est déplacée sur un élément
onmouseover	Quand la souris entre dans l'espace d'un élément
onmouseout	Quand la souris sort de l'espace d'un élément

Evènements des formulaires

Evènement	Description
onchange	Est provoqué au fur et à mesure que le contenu d'un contrôle est changé
oninput	Provoqué au fur et à mesure du changement du contenu dans une zone de texte
onfocus	Provoqué lorsqu'un contrôle reçoit le focus
onblur	Provoqué lorsqu'un contrôle perd le focus
onsubmit	Lorsque le formulaire est envoyé au serveur

Evènements du clavier

Evènement	Description
onkeydown	Une touche est entrain d'être appuyée
onkeypress	Une touche est appuyée
onkeyup	Une touche est relachée

Evènements de la fenêtre

Evènement	Description
onload	Appelé lorsque la page est chargée
onresize	Lorsque la fenêtre est redimensionnée
onerror	Lorsqu'une erreur se produit lors du chargement d'une image
onabort	Lorsque le chargement d'une image est annulé

Les minuteurs

- Les minuteurs sont des gestionnaires d'évènements qui sont déclenchée à des intervalles de temps définies en millisecondes
- Les minuteurs (timers) déclenchent des actions périodiques sur la page
- La création d'un minuteur se fait en utilisant « **setInterval** »
- L'arrêt d'un minuteur se fait en utilisant « **clearInterval** »

Interaction avec les fenêtres

Méthode	Description
open()	Ouvre une nouvelle fenêtre
close()	Ferme la fenêtre
moveTo()	Déplace la fenêtre
resiezTo()	Redimensionne la fenêtre
moveBy()	Déplace la fenêtre par rapport à la position en cours
resizeBy()	Redimensionne par rapport à la taille actuelle

Boîtes de dialogue

Méthode	Description
alert(msg)	Affiche un message sous forme de dialogue
confirm(msg)	Message de confirmation. Renvoie true or false selon la réponse de l'utilisateur (oui / non)
prompt(msg,val_defaut)	Demande à l'utilisateur d'entrer une valeur

Démonstration (Evenements.html)

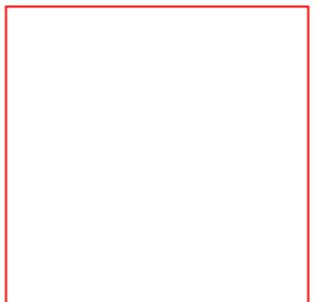
Page chargée

379,238

Survolez-moi

Survolez-moi

Survolez-moi



Affecter évènement

11:29:18 Arrêter



Ouvrir dans une nouvelle fenêtre



Section 8 : JQuery

Introduction

- Bibliothèque JavaScript adoptée par la majorité des industriels : IBM, Netflix, Amazon, Dell, Twitter et Microsoft
- JQuery facilite la manipulation du DOM en appliquant des connaissances HTML, CSS ou JS
- JQuery permet de créer des scripts non obstrusifs
- Référence : <http://api.jquery.com/>
- Téléchargement dernière version : <http://jquery.com/>

Avantages

- Economie de code (faire plus de choses avec moins de code)
- Code non obstrusif (séparation de la structure du traitement)
- Compatibilité avec tous les navigateurs
- Taille insignifiante du code

Blocs fonctionnels

Sélection,
Manipulation et
Création
d'éléments

Fonctions
Utilitaires

Fonction
d'initialisation

Extension avec
les plugins

Principe de base

- Pour manipuler JQuery, il faut passer par un objet « JQuery »
- Pour rendre la syntaxe plus compacte, « \$ » est un alias sur cet objet
- Le traitement initial et l'affectation des évènements se fait dans la fonction d'initialisation

Fonction d'initialisation - Syntaxes

```
// syntaxe 1
$(function () {
    // traitement
});

// syntaxe 2
$(document).ready(function () {
    // traitement
});
```

Sélecteurs JQuery

- Les sélecteurs JQuery permettent de sélectionner un ou plusieurs éléments HTML
- Les sélecteurs renvoient des **ensembles enveloppés**
- Pour identifier les éléments, JQuery utilise une syntaxe proche de celle des sélecteurs CSS avec quelques petites spécificités

Exemples de sélecteurs

Sélecteur	Description
*	Tous les éléments
div	Tous les div
#monId	Sélectionne l'élément dont l'id est « monId »
p span.maClasse	Tous les span dont la classe est « maClasse » et qui sont descendants d'un paragraphe
div,p	Tous les div et tous les paragraphes
li:first	Premier li
li:odd	Li pairs
li:nth-child(3n+1)	1 ^{er} , 4 ^{ème} ,...

Exemples de sélecteurs - Suite

Sélecteur	Description
eq(n)	Sélectionne le nème élément (1 ^{er} démarre par 0)
lt(n)	Les n premiers, après le nème, les éléments sont ignorés
gt(n)	Les derniers après le nème élément
input[type=checkbox]	Toutes les cases à cocher
input[type=checkbox][checked]	Toutes les cases à cocher qui sont cochées
:animated	Les éléments qui sont en cours d'animation
:enabled	Éléments activés
:disabled	Éléments désactivés

Exemples de sélecteurs - Suite

Sélecteur	Description
:has(element)	Les éléments qui ont un descendant « élément »
:not	Inverse un ensemble

Création d'éléments HTML

- L'alias \$ permet aussi de créer des balises HTML
- Par exemple `$(« <p>Mon par</p> »)` ou `$(« <div> »)`

Ajout d'éléments DOM

Sélecteur	Description
appendTo(elt)	Ajoute des éléments à un ensemble
insertAfter(elt)	Insère après
insertBefore(elt)	Insère avant
html(elt)	Change le HTML interne d'un élément

Gestion d'un ensemble enveloppé

Fonction	Description
size()	Renvoie le nombre d'éléments d'un ensemble
get(n)	Renvoie l'nème élément de l'ensemble
eq(n)	Renvoie un ensemble enveloppé contenant au plus le nème élément
first()	Renvoie un ensemble enveloppé contenant au plus le premier élément
last()	Renvoie un ensemble enveloppé contenant au plus le dernier élément
toArray()	Convertit l'ensemble en un tableau Javascript
index(elt)	Cherche elt dans l'ensemble puis renvoie sa position ou -1 s'il n'est pas trouvé
add(elt)	Ajoute des éléments à l'ensemble
filter(expression)	Filtre les éléments à filtrer
slice(debut,fin)	Renvoie un sous-ensemble de l'ensemble original

Gestion d'un ensemble enveloppé - Suite

110

Fonction	Description
each(fonction)	Fonction appelée pour chaque fonction
children(elts)	Renvoie tous les descendants des éléments d'un ensemble
closest(elts)	Renvoie les éléments les plus proches
find(selecteur)	Cherche les éléments conformes au sélecteur
is(selecteur)	Indique si un élément de l'ensemble est conforme au sélecteur

Gestion des éléments d'un ensemble

Fonction	Description
attr(nom_attribut)	Donne la valeur de l'attribut du premier élément
attr(nom,valeur)	Change la valeur d'un attribut
data(nom)	Renvoie une donnée stockée sur un élément
data(nom,valeur)	Change la valeur d'une donnée stockée sur un élément
removeData(nom)	Supprime une donnée stockée
addClass(nom)	Ajoute une classe CSS à un élément
removeClass(nom)	Supprime une classe CSS d'un élément
toggleClass(nom)	Ajoute ou supprime une classe CSS d'un élément
hasClass(nom)	Indique si un élément de l'ensemble a une classe
css(nom,valeur)	Applique directement un style css à un élément

Gestion des éléments d'un ensemble - Suite

Fonction	Description
width(), width(valeur), height(), height(valeur)	Renvoie ou change la hauteur ou la largeur des éléments d'un ensemble
text() ou text(valeur)	Renvoie ou change le texte d'un élément
wrap(html)	Envelopper les éléments dans un conteneur
remove(selecteur)	Supprime les éléments de la page
empty()	Vide les éléments
clone()	Clone les éléments
val() ou val(valeur)	Renvoie ou modifie la valeur d'un contrôle de formulaire
hide()	Cacher les éléments
show()	Montre des éléments précédemment cachés
toggle()	Cache / Montre l'élément

Gestion des évènements

- JQuery permet de simplifier l'affectation des évènements aux éléments et élimine tous les problèmes d'incompatibilité
- JQuery permet d'affecte un évènement de plusieurs façons et de le désaffecter
- JQuery peut même affecter des évènements à des éléments qui n'existent pas encore

Fonctions de gestion des évènements

Fonction	Description
click, hover	Des alias pour le clic ou le survol de la souris
bind(nom, fonction)	Associe une fonction à un évènement
unbind(nom)	Supprime un gestionnaire d'évènement
one(nom, fonction)	Gestionnaire d'évènement qui ne s'exécute qu'une seule fois
on(evt,selecteur,fonction)	Attache un évènement aux éléments sélectionnés (s'applique aussi aux éléments créés dynamiquement)
trigger(evt,donnees)	Déclenche manuellement un évènement

Animations

- Certaines fonctions comme « show » ou « hide » peuvent être animées en ajoutant une durée d'exécution
- La fonction « animate » permet de changer des propriétés d'une manière animée
- Il est intéressant de combiner Jquery avec les animations et les transformations CSS3

Fonctions d'animation

Fonction	Description
show(), hide(), toggle()	Ces fonctions peuvent être animées
fadeIn(), fadeOut()	Cache / Montre en utilisant les opacités
animate()	Change une ou plusieurs propriétés en les animant

Introduction à AJAX

- AJAX ou Asynchronous Javascript And XML permet d'interroger le serveur sans avoir à rafraîchir toute la page
- Le web 2,0 est techniquement basé sur AJAX

Principes AJAX

Requêtes HTTP
(Post / GET)

Format
d'échange
(XML, JSON)

Rechargement
Partiel

Fonctions
callback

AJAX avec JQuery

- JQuery facilite l'intégration d'AJAX dans les applications web
- La fonction AJAX est une fonction utilitaire (\$.ajax)
- Les deux alias \$.get et \$.post permettent de faire des requêtes AJAX en mode GET ou POST

Fonctions ajax

Fonction	Description
<code>\$.get(url,callback,type)</code>	Requête AJAX en mode GET
<code>\$.post(url,callback,type)</code>	Requête AJAX en mode POST
<code>\$.getJSON(url)</code>	Demande des données en format JSON à parti du serveur
<code>\$.ajax(options)</code>	Fonction détaillée qui gère tous les paramètres AJAX

Plugins JQuery

- Les plugins permettent d'étendre les fonctionnalités de Jquery
- Il existe plusieurs catégories : effets, diaporama, utilitaire,...
- L'appel d'une fonction d'un plugin se fait toujours en passant par l'alias Jquery (\$)

- Librairie permettant d'ajouter des contrôles dynamiques côté client
- Pour plus de détails, consulter : <http://jqueryui.com/>

Démonstration

Bienvenue dans le tutoriel !

Entrez le sélecteur : `vignette:nth-child(2n)` Entrez le filtre : Appliquer Style Iterer Envelopper Supprimer Cacher/Montrer Ajouter vignette 1000 Cacher/Montrer animé Animer



appliqué sur 4 éléments

Entrez le code HTML : Appliquer HTML



Bibliographie

- Java Script Reference, W3C Schools
- Sam's Teach Yourself, HTML, CSS, and JavaScript, 2012
- JQuery in Action, Bear Bibeault, Yehuda Katz, Mann, 2010
- JQuery reference (www.JQuery.com)