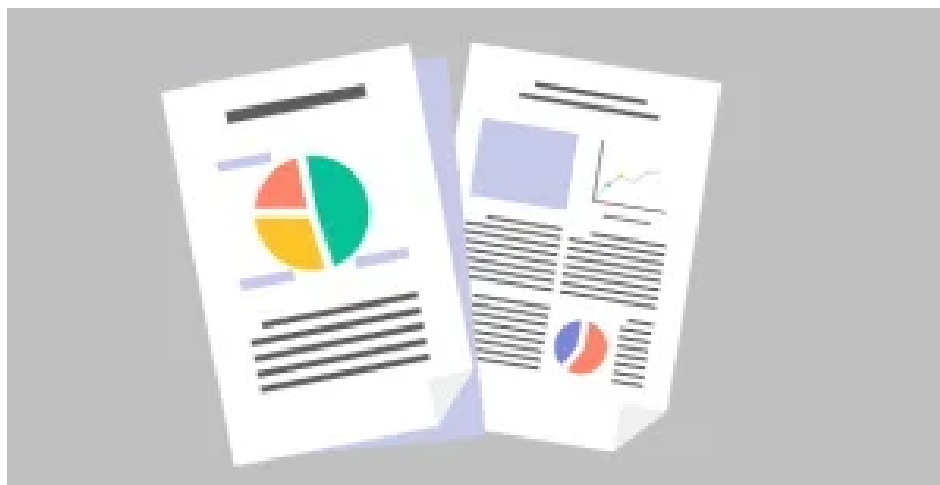


🏠 Cours gratuits » Cours informatique » Cours développement web » Cours JAVASCRIPT

Cours détaillé en JavaScript



3.7 étoiles sur 5 a partir de 14 votes. ★★★★★

Votez ce document: ☆☆☆☆☆

Cours détaillé en JavaScript enjeux et pratique

JavaScript

Quid ?

Langage de programmation lié à HTML.

Code JavaScript intégré aux pages HTML.

Code interprété par le navigateur client (interprétation dépendante du type et de la version de navigateur).

≠ code PHP (interprété du côté serveur).

JavaScript est un langage à prototype (par opposition à un langage basé sur les classes et sous classes pour réaliser l'héritage) Tout objet JavaScript est doté d'une propriété prototype, qui représente le modèle sur lequel l'objet en question se base. événementiel (association d'actions aux événements déclenchés par l'utilisateur (passage de souris, clic, saisie clavier, etc...)).

Les objets

Un objet est un élément nommé ayant des

Propriétés : paramètres que vous vérifiez et modifiez.

Méthodes : actions que l'objet est capable d'effectuer.

Événements : choses qui arrivent à l'objet, auxquelles celui-ci peut répondre automatiquement par une action.

Les événements

Action Event JavaScript DOM Réaction

L'utilisateur place la souris sur l'objet

Identification d'une action concernant l'objet

Indique à l'objet ce qu'il doit faire

Localise l'objet dans la page WEB La source de l'image

Change

Les événements

```
im1 = new Image();
```

```
im2 = new Image();
```

```
im1.src = "image1.gif";
```

```
im2.src = "image2.gif";
```

```
function changeImage (nomIm,src)
```

```
{
```

```
document.images[nomIm].src = src;
```

```
}
```

```
<A HREF="#"
```

```
onMouseover="changeImage('image1',im2.src)"
```

Articles similaires

[Comment rédiger un rapport journalier/mensuel d'avancement des travaux de chantier ?](#)

[Exercice HTML: Réalisation d'un Formulaire de QCM](#)


[Exercice HTML: Formulaire Html avec Fonction Javascript](#)

[Cours de soutien scolaire bénévole - Informations et conseils](#)

[Cours particuliers : une nouvelle école informelle ?](#)


[Gestion locative : quels sont les meilleurs logiciels ?](#)

Documents similaires

 [Introduction à Javascript cours](#)

Introduction à Javascript cours

 [Cours de JavaScript](#)
Cours de JavaScript

 [Cours Complet de JavaScript](#)

Cours Complet de JavaScript

 [Cours JavaScript : Les formulaires](#)

Cours JavaScript : Les formulaires

 [Cours interactions Html et JavaScript](#)

Cours interactions Html et JavaScript

 [Cours et exercices JavaScript](#)

Cours et exercices JavaScript

```
document.getElementById("lien1").addEventListener('mouseover',verifie,false);
```

Contactez-nous

Web 2.0

A propos de nous

JavaScript

On recrute

Intérêts de JavaScript ?

Rechercher dans le site

Supporte (par défaut) par les principaux navigateurs, ne nécessite pas de plug-in particulier.

Politique de confidentialité

Accès aux données personnelles sans plugin Flash (HTML (+ possibilité de les manipuler relativement facilement)).

Droit d'auteur/Copyright

Possibilité de mettre en place des animations sans l'inconvénient des longs temps de chargement nécessités par les données multimédia.

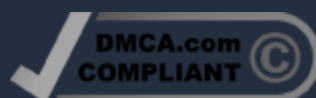
Plan du site

Langage relativement sécurisé : il est impossible de lire ou d'écrire sur le disque client (impossibilité de récupérer un virus par ce biais).

Arrivée d'AJAX

JavaScript

Difficultés d'utilisation de JavaScript ?



Comme pour HTML, il n'y a pas de standard pour l'accès aux différents objets d'un document (dépendant du navigateur).

Si le script ne fonctionne pas, la page est, le plus souvent, inutilisable. Attention au menu non visible !

Les utilisateurs peuvent empêcher l'exécution de code JavaScript, souvent en raison des erreurs générées par les scripts, ou encore en raison de la nature de l'interaction (apparition de nouvelles fenêtres, ...).

Lenteur d'exécution des scripts, ainsi que pour les scripts complexes, un certain délai avant le démarrage

Intégration de JavaScript dans HTML

Élément SCRIPT

```
<HTML>
```

```
...
```

```
<SCRIPT language="JavaScript">
```

```
<!-- // Masquage du code pour les navigateurs
```

```
// ne supportant pas JavaScript
```

```
... code JavaScript ...
```

```
//--> // Fin de la partie cachée
```

```
</SCRIPT>
```

```
...
```

```
</HTML>
```

Intégration de JavaScript dans HTML

Place de l'élément SCRIPT

Possibilité d'intégrer du code JavaScript :

dans l'entête de la page.

dans le corps de la page.

Intégration dans un événement d'un objet de la page

Sous la forme d'un couple attribut-valeur HTML :

Attribut = événement déclencheur

Valeur = code JavaScript déclenché

```
...
```

```
<FORM name="formulaire" onSubmit="maFonction()">
```

```
...
```

Web 2.0

JavaScript à l'extérieur du HTML

```
<head>
```

```
<title>Vincennes-Sud</title> <meta http-equiv="Content-Type" content="text/html; charset=iso8859-15">
```

```
<meta http-equiv="Content-Style-Type" content="text/css"> <meta http-equiv="Content-Language" content="fr">
```

```
<link rel="stylesheet" type="text/css" media="all" href="stylemenugauche.css"> <link rel="shortcut icon" href="/%7Edupont/Images/icone.jpg"> <link  
rel="stylesheet" type="text/css" media="all" href="style.css">
```

```
<script LANGUAGE="JavaScript SRC="Agenda/mon_agenda.js"></script>
```

```
<script LANGUAGE="JavaScript" SRC="Scripts/gestion_rdv.js"></script>
```

```
<script LANGUAGE="JavaScript" SRC="Scripts/affiche_calendrier.js"></script>
```

```
<script LANGUAGE="JavaScript" SRC="Scripts/photo.js"></script>
```

```
</head>
```

```
<script language="JavaScript" src="Scripts/agenda.js"> </script>
```

Langage

Notations JavaScript

Similarités avec les langages

C, PHP, Java

Commentaire

Sur une ligne : // ... commentaire ...

Sur plusieurs lignes : /* ... commentaire

... */

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>
```

Web 2.0

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;  
  
} // compute  
  
affiche+= 'compute(5,4) = ' + compute(5,4);  
  
affiche+= '\ntotal = ' +total+ '\n factor = ' + factor +  
  
'\nresult = ' +result;  
  
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 : []

Propriétés de l'objet

...

```
var tableau=new Array;
```

```
tableau[0]=10;
```

```
tableau[9]=5;
```

...

...

// obtention du nombre d'éléments de un_tableau

```
var dimension=tableau.length;
```

...

Tableaux associatifs

Principe

L'indice est une chaîne de caractères

Exemple

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

...

```
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";
```

...

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]="O"; 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
```

Sauvegarder html 24

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>
```

Sauvegarder html 25

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>
```

Structures de contrôle

Test conditionnel : if ... else ...

But

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

Exemple

```
<SCRIPT language="JavaScript">
```

```
...
```

```
if(age<18) { alert("Vous devez être majeur");
```

```
window.location="mineur.php3";
```

```
}
```

```
else { window.location="majeur.php3";
```

```
}
```

```
...
```

```
</SCRIPT>
```

Boucle itérative : for ...

But

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;

}

...
```

Notations abrégées à la C

i++

+=

Boucle conditionnelle : while ...

But

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 du langage

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>
```

ex2

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 réalisés par la fonctions, 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 fonction
```

```
function bonjour(nom) {
```

```
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

```
<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>
```

JavaScript *console

```
<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>
```

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 a été modifiée par l'utilisateur

click : un clic 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 événementj = "actionj
```

```
" >
```

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

```
<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 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

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
```

```
2.718281828459045
```

JavaScript et "langage à objets"

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>");
```

```
// parcours des propriétés de l'objet navigator
```

```
var object=window.navigator;
```

```
for(i in object)
```

```
for(i in object)
```

```
{ document.write(i+" = "+object[i]+" <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+" .");
```

```
}
```

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+" .");
```

```
}
```

```
}
```

```
...
```

Modèle Objet de Document

Définition

Le modèle d'objet du document donne une représentation en mémoire des objet du document.

Un objet est un élément HTML qui a été défini par une ID ou un nom.

Le DOM est l'adresse par laquelle vous pouvez localiser un objet de la page HTML.

Les objets de la page peuvent être identifié par un attribut de nom ou d'ID qui lui donne son adresse unique et en fait un objet.

Adresse de l'objet

Le DOM décrit le chemin partant de la fenêtre du navigateur pour descendre jusqu'aux objet de la page Web.

Le DOM est structuré comme un arbre est suit de près la structure hiérarchique du code HTML.

L'arbre contient des nœuds, les nœuds peuvent avoir des fils, et tous les nœuds ont un parent (sauf la racine).

Exemple

```
<html>

<head>

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

<title>Exemple</title>

</head>

<body>

<h1>Le DOM </h1>

<p>un texte dans le <i>dom.</i></p>

</body>

</html>
```

```
<html>

<head>

<title>Exemple</title>

</head>

<body>

<h1>Le DOM </h1>

<p>un texte dans le <i>dom.</i></p>

</body>

</html>
```

Un arbre ?

Document

head body ...

text text

...

h1 p

text i

text

HTML html

Ce que nous prenons pour des espaces sans signification se retrouve dans le DOM.

Les nœuds texte figurant avant

H1, entre h1 et P, et après P représentent les lignes vides entre ces éléments.

Inspector DOM : inclus dans firefox et mozilla

Pour obtenir les infos DOM

Vous pouvez télécharger la barre d'outils pour Internet Web Developer

...

Node

C'est une des interfaces incontournable du modèle.

Les différents types de nœuds reflètent les différentes catégories de balisage d'un document : éléments, attributs, commentaires, textes.

Tous ces types de nœuds partagent un ensemble de propriétés et fonctions héritées de leur type générique : le nœud.

Ainsi, toutes les interfaces sur les nœuds disposent des propriétés et méthodes de Node.

Propriétés de parcours du DOM

```
<html>
```

```
<head>
```

```
<meta content="text/html; charset=ISO-8859-1"
```

```
http-equiv="content-type">
```

```
<title></title>
```

```
</head>
```

```
<body>
```

```
<h1 id="header">Les légumes<br></h1>
```

```
<ul>
```

```
<li id="a">Artichaut</li>
```

```
<li id="n">Aubergine</li>
```

```
<li id="c">carotte </li>
```

```
<li id="m">Mangue</li>
```

```
<li id="p">pomme de terre </li>
```

. .

```
</ul>
```

```
</body>
```

```
</html>
```

...

Propriétés de noeud

nodeName : le nom (par exemple #text)

nodeType : valeur numérique souvent à comparer avec les constantes

1 = Node.ELEMENT_NODE

2 = Node.ATTRIBUTE_NODE

3 = Node.TEXT_NODE

nodeValue : la valeur !

Parcourt dans le DOM

```
function getInnerText(node) {
```

```
var result = "";
```

```
if (Node.TEXT_NODE == node.nodeType)
```

```
return node.nodeValue;
```

```
if (Node.ELEMENT_NODE != node.nodeType)
```

```
return "";
```

```
for (var index = 0; index < node.childNodes.length; ++index)
```

```
result += getInnerText(node.childNodes.item(index));
```

```
return result;
```

```
} // getInnerText
```

DOM : objet Window

Propriétés frames[] : tableau de frames

frames.length : nombre de frames self : fenêtre courante

opener : la fenêtre (si elle existe) qui a ouvert la fenêtre courante

parent : parent de la fenêtre courante, si la fenêtre courante est une sous-partie d'un frameset top : fenêtre principale (qui a créé toutes les fenêtres)

status : message dans la barre de statut

defaultstatus : message par défaut de la barre de statut

name : nom de la fenêtre

DOM : objet Window

Méthodes

alert(string) : ouvre une boîte de dialogue avec le message

passé en paramètre

confirm : ouvre une boîte de dialogue avec les boutons

OK et cancel

blur() : enlève le focus de la fenêtre

focus() : donne le focus à la fenêtre

prompt(string) : affiche une fenêtre de saisie

scroll(int x, int y) : positionnement aux coordonnées (x,y)

open(URL, string name, string options) :

ouvre une nouvelle fenêtre contenant le document identifié par l'URL close() : ferme la fenêtre

DOM : objet Document

Propriétés

title : titre du document

location : URL du document

lastModified : date de dernière modification

referrer : URL de la page d'où arrive l'utilisateur

bgColor : couleur de fond

fgColor : couleur du texte

linkColor

vlinkColor couleurs utilisées pour les liens hypertextes

alinkColor

DOM : objet Document

Propriétés

forms[] : tableau des formulaires de la page

forms.length : nombre de formulaire(s) de la page

links[] : tableau des liens de la page

links.length : nombre de lien(s) de la page

anchors[] : tableau des ancres internes (<A NAME=

...>)

anchors.length : nombre de d'ancre(s) interne(s)

images[]

applets[] tableaux des images, applets et plug-ins

embeds[]

Remarque : les tableaux contiennent les éléments dans l'ordre de leur apparition dans le code HTML

DOM : objet Document

Méthodes

write(string) : écrit une chaîne dans le document

writeln(string) : idem + caractère de fin de ligne

clear() : efface le document

close() : ferme le document

Sauvegarder html 84

DOM : objet Form

Propriétés

name : nom (unique) du formulaire

method : méthode de soumission (0=GET, 1=POST) action : action déclenchée par la validation du formulaire

target : fenêtre de destination de la réponse (si elle existe)

elements[] : tableau des éléments du formulaires

length : nombre d'éléments du formulaire

DOM : objet Form

Méthodes

submit() : soumet le formulaire reset() : ré-initialise le formulaire

Événements

onSubmit(method) : action à réaliser lorsque le formulaire est soumis

onReset(method) : action à réaliser lorsque le formulaire est ré-initialisé

DOM : objet Navigator

Propriétés

appName : nom de code interne du navigateur

appName : nom réel du navigateur

appVersion : version du navigateur

userAgent : objet complexe contenant des détails sur :

l'appName,

l'appVersion le système d'exploitation utilisé plugins[] : tableau des plugins installés chez le client

mimeType[] : tableau des types MIME supportés par le navigateur html 87

DOM : objet Navigator

Méthodes

javaEnabled : retourne TRUE si le navigateur supporte Java (et que l'exploitation de Java est actif)

Exemples d'utilisation du DOM

Exemple de formulaire

Accès à l'objet correspondant au formulaire précédent

3 possibilités :

```
<FORM name="general">
```

```
<INPUT type="text" name="champ1" value="Valeur initiale">
```

```
</FORM>
```

```
document.forms["general"]
```

```
document.forms[0] //si vous voulez vous pendre
```

```
document.general
```

Exemples d'utilisation du DOM

Accès aux éléments du formulaire

3 possibilités :

Accès aux propriétés d'un élément

```
document.forms["general"].elements["champ1"]
```

```
document.forms["general"].elements[0] //voir remarque
```

```
document.forms["general"].champ1
```

```
document.forms["general"].elements["champ1"].value
```

Exemples d'utilisation du DOM

Appeler une méthode sur un objet

Pour donner le focus, par exemple :

Associer une action à un événement

```
<FORM name="changer">
```

```
<INPUT type="text" name="zonetexte" value="Valeur initiale">
```

```
<INPUT type="button" value="Changer la valeur"
```

```
onClick='document.forms["changer"].elements
```

```
["zonetexte"].value="NOUVEAU" '>
```

```
</FORM>
```

```
document.forms["general"].elements["champ1"].focus();
```

Exemples d'utilisation du DOM

```
<HTML>
```

```
<HEAD> <SCRIPT></SCRIPT> </HEAD>
```

```
<BODY>
```

```

<FORM name="changer">

<INPUT type="text" name="zonetexte"

value="Valeur initiale">

<INPUT type="button"

value="Changer la valeur"

onClick='document.forms["changer"].elements

["zonetexte"].value="NOUVEAU" '>

</FORM>

</BODY>

</HTML>

```

JavaScript

Arrivée l'arrivée d'AJAX, la création d'objet DOM a redoublé d'intérêt.

Voir cours AJAX.

Sauvegarder html 93

Exemple pour tout regrouper !

```

<html>

<head>

<link rel='stylesheet' type='text/css' href='hello.css' />

<script type='text/javascript' src='hello.js'></script>

</head>

<body>

<p id='hello'>hello</p>

<div id='empty'></div>

</body>

```

hello.js

```

window.onload=function() {

var hello=document.getElementById('hello');

hello.className='declared';

var empty=document.getElementById('empty');

addNode(empty,"reader of");

addNode(empty,"Ajax in Action!");

var children=empty.childNodes;

for (var i=

```

```
0;i<children.length;i++){  
  
    children[i].className='programmed';  
  
}  
  
empty.style.border='solid green  
  
2px';  
  
empty.style.width="200px";  
  
}  
  
function addNode(el,text){  
  
    var childEl=document.createElement("div");  
  
    el.appendChild(childEl);  
  
    var txtNode=document.createTextNode(text);  
  
    childEl.appendChild(txtNode);  
  
}
```

hello.js

```
window.onload=function() {  
  
    var hello=document.getElementById('hello');  
  
    hello.className='declared';  
  
    var empty=document.getElementById('empty');  
  
    addNode(empty,"reader of");  
  
    addNode(empty,"Ajax in Action!");  
  
    var children=empty.childNodes;  
  
    for (var i=0;i<children.length;i++){  
  
        children[i].className='programmed';  
  
    }  
  
    empty.style.border='solid green 2px';  
  
    empty.style.width="200px";  
  
}  
  
function addNode(el,text){  
  
    var childEl=document.createElement("div");  
  
    el.appendChild(childEl);  
  
    var txtNode=document.createTextNode(text);  
  
    childEl.appendChild(txtNode);
```

```
}
```

Syntaxe

Object document.getElementById(String id)

Description

Retourne un objet HTML à partir de son id, défini dans la propriété id de la balise de l'objet.

hello.js

```
window.onload=function() {  
  
var hello=document.getElementById('hello');  
  
hello.className='declared';  
  
var empty=document.getElementById('empty');  
  
addNode(empty,"reader of");  
  
addNode(empty,"Ajax in Action!");  
  
var children=empty.childNodes;  
  
for (var i=0;i<children.length;i++){  
  
children[i].className='programmed';  
  
}  
  
empty.style.border='solid green 2px';  
  
empty.style.width="200px";  
  
}  
  
function addNode(el,text){  
  
var childEl=document.createElement("div");  
  
el.appendChild(childEl);  
  
var txtNode=document.createTextNode(text);  
  
childEl.appendChild(txtNode);  
  
}
```

Syntaxe

Object document.createElement(String id)

Description

Créer un nouvel élément HTML en prenant le type de balise en argument.

hello.js

```
window.onload=function() {  
  
var hello=document.getElementById('hello');  
  
hello.className='declared';
```

```

var empty=document.getElementById('empty');

addNode(empty,"reader of");

addNode(empty,"Ajax in Action!");

var children=empty.childNodes;

for (var i=

0;i<children.length;i++){

children[i].className='programmed';

}

empty.style.border='solid green

2px';

empty.style.width="200px";

}

function addNode(el,text){

var childEl=document.createElement("div");

el.appendChild(childEl);

var txtNode=document.createTextNode(text);

childEl.appendChild(txtNode);

}

```

Syntaxe

Object el.appendChild(fil)

Description

ajoute un n

œud fil au noeud el.

JavaScript Et formulaires

Rappel sur les formulaires

Langage HTML

Déclaration de formulaire : <FORM ...> ... </FORM>

Élément(s) d'un formulaire : <INPUT ...>

But

Interaction avec l'utilisateur sous la forme d'une saisie d'informations.

Intérêt de JavaScript

Augmenter l'interaction du côté client, par exemple pour :

assister et guider le visiteur, contrôler la saisie, réaliser des traitements (calcul, ...), envoyer des résultats au serveur.

Événements associés à <FORM ...>

onSubmit = " ... "

Détecte la soumission du formulaire

onReset = " ... "

Détecte la réinitialisation du formulaire

Éléments <INPUT ...>

Propriétés

name : nom du champ

type : type du champ (text, button, radio, checkbox, submit, reset)

value : valeur textuelle

size : taille du champ

maxlength : taille maximale d'un champ texte

checked : case à cocher / bouton radio coché ou non

disabled : grisé (modification impossible)

readOnly : lecture seule class : nom de la classe de style

style : nom du style

Sauvegarder html 102

Élément <INPUT type="text" ...>

Propriétés acceptées

name, value, defaultvalue, size, maxlength, disabled, readOnly, class, style

Méthodes acceptées focus, blur

Événements acceptés

onFocus, onBlur, onChange

Sauvegarder html 103

Éléments <INPUT ...>

Méthodes

focus : donne le focus (curseur)

blur : enlève le focus

click : simule un click (sur un bouton)

Événements

onFocus : détecte la prise de focus

onBlur : détecte la perte de focus

onClick : détecte un click

onChange : détecte les changements