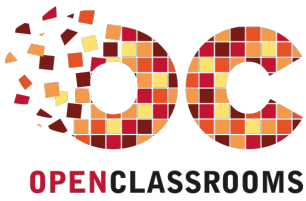
Les Bookmarklets ou comment modifier la page d'un site depuis la barre d'adresse

Par Xavier MONTILLET (xavierm02)



www.openclassrooms.com

Licence Creative Commons 6 2.0
Dernière mise à jour le 28/04/2011

Sommaire

Les Bookmarklets ou comment modifier la page d'un site depuis la barre d'adresse

Par  Xavier MONTILLET (xaviern02)

Mise à jour : 28/04/2011
Difficulté : Facile

(cc) BY-NC-SA

Vous avez déjà voulu voir le code source généré par JavaScript sans ouvrir un débugger ?
Pré-remplir des champs en un clic ?
Connaître la valeur des champs cachés sur une page ?
Ajouter quelque chose dans une page d'un site qui ne vous appartient pas pour le personnaliser sans devoir copier son code source et rendre tous ses liens absolus ?
Ou vous avez tout simplement soif de connaissances ?
Si oui, ce tutoriel est pour vous

Prérequis

La compréhension de ce tutoriel nécessite de connaître les bases du JavaScript et, si vous avez appris le JavaScript ailleurs que sur le SdZ, je vous conseille de lire le tutoriel sur les bonnes pratiques en JavaScript.

Il peut également être utile de connaître le DOM et le DHTML car ces parties du JavaScript sont souvent utilisées dans les bookmarklets et seront donc utiles pour les exemples.
De même pour les closures qui sont presque obligatoires pour certaines utilisations des bookmarklets (voir chapitre 3).
Sommaire du tutoriel :

Introduction

Vous avez sans doute déjà vu du JavaScript dans le `href` de "liens" du genre :

Code : HTML

```
<a href="javascript:fonction_javascript();">Lien</a>
```

Ce n'est pas bien ! Il ne faut pas le faire !
Voir le tutoriel sur les bonnes pratiques de nod_.

Eh bien on peut aussi en mettre **directement dans la barre d'adresse** et donc aussi **stocker le code comme favoris/signets dans son navigateur**.
On peut donc faire plein de trucs sympathiques, dont quelques exemples sont donnés dans la dernière partie de ce tutoriel, en cliquant simplement sur un favori/signet dans lequel on aura auparavant enregistré du code JavaScript !
C'est pour cela que ces morceaux de JavaScript précédés de `javascript:` sont appelés : **bookmarklets**, **applisignets**,

marque-pages scriptés ou encore **favelets**.

Les emplois de ces bookmarklets sont très variés :

- Pour les développeurs :
  - Afficher le code généré par le JavaScript
  - Afficher le contenu d'une variable, tester quelque chose sans avoir à se créer une page avec des balises `<script>` etc.
  - Tester un code directement sur un site pour savoir s'il est envisageable de développer un *userscript* pour ce site
  - Connaître la valeur d'un champ caché
- Pour les utilisateurs lambda :
  - Éviter d'attendre sur les sites comme RapidShare et autres
  - Tricher dans les jeux faits en JavaScript
  - Remplir des champs si on doit souvent remplir le même formulaire
  - Connaître l'url d'une vidéo (par exemple sur Youtube) sans chercher dans le code source (si quelqu'un d'autre ou vous même avez pris la peine de regarder le code source pour dire au JavaScript comment y accéder avant)
- Autres...

Des exemples concrets sont donnés dans la dernière partie de ce tutoriel.

[TP] Rajouter des smileys

L'objectif

L'objectif est de pouvoir utiliser les images présentes dans le pense-bête (montrées par la flèche rouge) comme les smileys du SdZ montrés par la flèche verte).

Le contenu des onclick

Énoncé

Nous allons dans un premier temps chercher ce que nous allons mettre dans le `onclick` des images.

De quoi avons-nous besoin ?

- Tout d'abord, il faut que vous ayez activé le pense-bete. Si ce n'est pas le cas, utilisez ce bookmarklet qui va ouvrir la page et cocher la case pour vous (il ne vous restera qu'à valider):
  Secret (cliquez pour afficher)

  Code : JavaScript

```
javascript:
window.open("infos-286.html").onload=function() {
 this.document.getElementById("postit_visible").checked="true"
;
};
void 0;
```

- Ensuite, il faut des images dans le pense-bete... Si vous n'en avez pas, utilisez ce bookmarklet :
  Secret (cliquez pour afficher)

  Code : JavaScript - Code pour ceux qui n'ont pas d'images dans leur pense-bête

```
javascript:
```

```
(function(){
 function i(n){
  return '<img src="http://princessedesanes.free.fr/gifs/kirby/kirbysstar
/>';
 };
 document.getElementById('text_notepad').innerHTML=i('marche')+i('court')
;
})();
```

- Et enfin, il nous faut aussi l'id du textarea : `text1`
- Le code JavaScript que le SdZ utilise pour insérer les smileys dans le textarea :
  `balise(':p','','text1');parse('text1','prev_text1'); return false;`
  Le `return false;` sert normalement à annuler une action (comme un clic sur un lien) mais là, aucune action de ce genre n'a besoin d'être annulée... donc on va s'en passer...
  La fonction `parse()`, on n'a pas besoin d'y toucher.
  Ce que l'on va devoir changer, c'est les arguments de la fonction `balise()`.

A vous de jouer !

Indices

Secret (cliquez pour afficher)

`balise(':p','','text1');` insère `:p` dans le textarea... c'est donc le premier argument qu'il nous faut changer.

Ce code sera dans le onclick de chaque image. Il faut donc qu'une variable ou qu'un attribut propre à l'image soit utilisé.

Comme ce code est dans le onclick, on peut utiliser `this` pour référer à l'image... et on peut donc facilement accéder à ses attributs.

On peut accéder au `src` de l'image avec `this.src`.

Il ne faut pas oublier les balises zcode `[image] [/image]`.

Correction

Secret (cliquez pour afficher)

Code : JavaScript

```
balise('<image>'+this.src+'</image>','','text1');parse('text1','prev_text1');
```

On inclut dans le textarea la balise `[image] [/image]` et l'url de l'image au milieu.

Rajouter les onclick sur toutes les images

Énoncé

De quoi avons-nous besoin ?

- L'id du pense-bête : `text_notepad`
  On peut donc accéder au pense-bête par `document.getElementById('text_notepad')` et aux images par `document.getElementById('text_notepad').getElementsByTagName('img')`.
- Une boucle qui rajoute le onclick à chaque image.

C'est à vous  (N'essayez pas de mettre directement le code dans un bookmarklet. Faites comme un code normal avec sauts de

lignes, indentation et tout ce qui va bien.)

Indices

Secret (cliquez pour afficher)

Une boucle `for()` peut être un bon choix.

Code : JavaScript

```
var images =
document.getElementById('text_notepad').getElementsByTagName('img');
for(var i=0,l=images.length;i<l;i++) {
 //on fait quelque chose :)
}
```

Pour mettre un onclick sur l'image on fait :

Code : JavaScript

```
img.onclick = function() {
 //ce qui se passe quand on clic sur l'image
};
```

Il existe d'autres façons de le faire mais c'est, de celles qui sont propres et comprises par tous les navigateurs, la plus simple.

Correction

Secret (cliquez pour afficher)

Code : JavaScript

```
var images =
document.getElementById('text_notepad').getElementsByTagName('img');
for(var i=0,l=images.length;i<l;i++) {
 images[i].onclick=function() {

balise('<image>'+this.src+'</image>','','text');parse('text','prev_text');
 };
}
```

Et voilà, on peut insérer des smileys facilement 
Pour tester, vous pouvez, par exemple, aller poster un commentaire 

La transformation en bookmarklet

Maintenant, il ne reste qu'à faire tenir ça sur une seule ligne et à rajouter `javascript:`.
Et voilà le travail :

Code : JavaScript

```
javascript:var images=document.getElementById('text_notepad').getElementsByTagNan
i=0,l=images.length;i<l;i++){images[i].onclick=function(){balise('<image>'+this.s
```

Améliorations

Même si ce bookmarklet est magnifique et tout et tout (et en plus, il marche !), il est encore possible de l'«améliorer».

On peut se passer de certains `;` (ceux qui se trouvent avant des `}` fermantes et raccourcir le nom des variables, ce qui est bien pratique pour les navigateurs qui limitent le nombre de caractères (je ne donnerai pas de nom).

Code : JavaScript

```
javascript:var s=document.getElementById('text_notepad').getElementsByTagName('in
i=0,l=s.length;i<l;i++){s[i].onclick=function(){balise('<image>'+this.src+'</imag
```

Le problème avec notre code, c'est que l'on garde une variable globale...
On va donc utiliser une fonction anonyme que l'on va faire s'exécuter directement.
Ainsi, la variable (`images` ou `s`) sera locale et sera donc supprimée à la fin de l'exécution et on libérera un peu de mémoire
(De plus, mettre le code dans une closure comme cela permet de ne pas du tout agir sur les variables de la page et donc de ne pas perturber le fonctionnement des scripts)

Code : JavaScript

```
javascript:(function(){var s=document.getElementById('text_notepad').getElementsB
i=0,l=s.length;i<l;i++){s[i].onclick=function(){balise('<image>'+this.src+'</imag
```

Si vous ne comprenez pas ce dernier code, vous pouvez aller voir le tutoriel sur les closures de Darkodam.

P.S. : Je ne vous l'ai pas dit avant parce que sinon, vous n'auriez sans doute pas cherché à comprendre comment on compresse, mais il existe des sites qui le font pour vous comme par exemple http://closure-compiler.appspot.com/home

Afin de garder les codes lisibles, je ne mettrai plus les codes sous forme de bookmarklets exception faite des codes d'une seule ligne.
À vous de rajouter le préfixe `javascript:`.
Pour ce qui est du fait de tout mettre sur une ligne, cela peut être faire en copiant un code sur plusieurs lignes et en le copiant directement dans la barre d'adresse (le navigateur se chargera de supprimer les retours à la ligne).
Attention toutefois aux commentaires `//` qui au lieu de mettre en commentaire seulement la ligne qui suit mettrait toute la suite du code en commentaire.

Intercepter les valeurs renvoyées

Le problème

Certains d'entre vous ont dû être très déçus en essayant de continuer tout seuls ; en essayant par exemple de changer la couleur de fond du **`<body>`** avec : `javascript:document.body.style.backgroundColor="pink";` (ou toute autre assignation avec le signe =) et en se retrouvant avec `pink` (ou la valeur assignée) écrit sur leur page... qui est devenue complètement blanche... Ils ont paniqué, rafraîchi la page et vu que ça redevenait normal.

Que s'est-il passé ?

En gros, quand on fait `document.body.style.backgroundColor="pink";` , le JavaScript renvoie "pink" car quand on affecte une valeur à une variable, la valeur renvoyée est la valeur affectée.
D'où :

Code : JavaScript

```
var a,b,c;
```

```
a=b=c=1;
alert(a);//1
alert(b);//1
alert(c);//1
```

On peut vérifier en faisant :

Code : JavaScript

```
javascript:alert(document.body.style.backgroundColor="pink");
```

Et là, ô miracle, le fond devient rose.
Et le `alert()` affiche bien pink.
Il faut donc utiliser une fonction pour qu'elle récupère ce qu'aurait dû renvoyer le JavaScript et que rien ne soit renvoyé.

Ne rien renvoyer au navigateur

Nous savons que quand on assigne une valeur, cette valeur est renvoyée.

Code : JavaScript

```
javascript:document.body.style.backgroundColor="red";
//renvoie "red"
```

Ce qui aboutit à l'affichage de ce qui est renvoyé...

Nous savons aussi qu'en le mettant dans un `alert()` , le code est exécuté normalement.

Code : JavaScript

```
javascript:alert(document.body.style.backgroundColor="yellow");
//le alert() capture la valeur renvoyée
```

Nous devons donc savoir ce que renvoie le `alert()` pour renvoyer la même chose.
Nous allons donc faire :

Code : JavaScript

```
javascript:alert(alert(document.body.style.backgroundColor="blue"));
```

Et là, on voit que c'est **`undefined`** qui est renvoyée.

La solution simple : `void`

On Google-ise un peu et on voit un opérateur a été prévu pour faire ça : **`void()`** .
Il évalue l'expression qui lui est donnée (donc il faut lui donner un paramètre) et renvoie **`undefined`** .

Il peut s'utiliser de plusieurs façons :

Code : JavaScript

```
javascript:document.body.style.backgroundColor='green';void(0)
//ou
javascript:document.body.style.backgroundColor='purple';void 0
//ou
javascript:void(document.body.style.backgroundColor='orange')
```

Les exemples ont été mis dans mon ordre de préférence.
Je préfère personnellement les deux premiers, car ils sont clairs et permettent, en enlevant `javascript:` et `void(0) /* ou */ void 0` , d'obtenir du code JavaScript pouvant être utilisé dans une page web.
Après, vous choisissez. Tous marchent (en théorie) sur tous les navigateurs.

La solution correcte mais pas compatible avec tous les navigateurs : `return undefined`

Cet opérateur est cependant déprécié en raison du fait que dans la plupart des langages,
`void` est un type de variable alors qu'en JavaScript, c'est une fonction qui renvoie `undefined` et qu'il peut donc porter à confusion.
Elle n'est donc pas acceptée par JSLint (une sorte de vérificateur de code JavaScript fait par Douglas Crockford, l'expert JavaScript actuel) qui permet de coder proprement et sans erreurs (enfin qui le signale si ce n'est pas le cas).
Il existe bien sûr une alternative qui passe à ce "validateur JavaScript" : `return undefined` .

Code : JavaScript - Exemple avec return undefined

```
javascript:document.body.style.backgroundColor='black';return
undefined
```

Cependant, certains navigateurs (et cette fois IE n'est pas le seul) n'acceptent pas les `return` dans la barre d'adresse (enfin dans les bookmarklets quoi).

Comme vous l'avez sans doute remarqué, je n'ai jusqu'à présent utilisé aucune closure dans cette partie.
Cela est dû au fait que nous avons changé la valeur d'attributs déjà existants et qu'aucune variable autre n'a été créée (dans le cas contraire, une closure aurait été utile pour éviter que la variable ne continue d'occuper de la mémoire après la fin de l'exécution du bookmarklet).

La meilleure solution : les closures

Pour comprendre cette partie du tutoriel, il faut savoir ce que sont les closures. Je vous conseil donc d'aller lire ce superbe tutoriel de Darkodam.

Comme vous le savez tous, une fonction ne renvoit une valeur qui si elle comporte une instruction `return` .

Code : JavaScript

```
function test() {
 alert('cette fonction ne sert à rien');
}
alert(test());//undefined
```

nous pouvons donc utiliser une closure afin d'intercepter le retour et d'empêcher ce retour de s'afficher à la place de la page HTML.

Code : JavaScript - Exemple

```
javascript:(function(){document.body.style.backgroundColor='aqua';})();
```

Cette façon de faire, bien que compliquée au premier abord permet, comme nous l'avons vu, en plus d'intercepter les retours, d'éviter de créer des variables globales qui pourraient nuire au fonctionnement du javascript déjà présent sur le site.
Elle est valide du point de vue de JSlint.

C'est la méthode à privilégier et celle que j'emploierais dans la suite de ce tutoriel.

Exemples de bookmarklets

Vous savez maintenant vous servir des bookmarklets (c'était dur hein). Nous allons donc nous pencher sur certains exemples concrets.

Si aucun exemple ne vous intéresse, sachez que si vous avez de bonnes connaissances en JavaScript (hors bookmarklets), ils ne vous apprendront rien (ou presque rien).

Désactiver un anti-clic droit

Il nous fait savoir comment les sites font pour désactiver le clic droit.
Il existe deux méthodes :

- Mettre une fonction contenant un `return false;` au oncontextmenu de l'élément `window` ou de `document.body`
- Mettre une fonction qui fait `return false;` au onclick (après avoir vérifié que c'était un clic droit) de l'élément `window` ou de `document.body`

Pour information, un `return false` , lorsqu'il est renvoyé par une fonction associée à un évènement, il annule l'évènement. Ainsi, `<a href="http://www.google.fr" onclick="return false;">Google</a>` n'enverra jamais celui qui clic dessus sur Google si il a activé le JavaScript.

Il nous faut donc empêcher ces retours (et donc enlever ces fonctions) associés à ces évènements.
On peut remplacer ces fonctions par :

- des fonctions vides (qu'on peut faire anonyme au passage) : `function(){}`
- le mot clef `undefined`

Nous pouvons donc faire :

Code : JavaScript

```
window.oncontextmenu = undefined; //ou function(){}
window.onclick = undefined; //ou function(){}
document.body.onclick = undefined; //ou function(){}
document.body.oncontextmenu = undefined; //ou function(){}
```

Or comme `a = b = 1;alert("a:"+a+" b:"+b);//a:1 b:1` , on peut faire :

Code : JavaScript

```
window.oncontextmenu = window.onclick = document.body.onclick =
document.body.oncontextmenu = undefined; //ou function(){}
```

On va stocker `document.body` dans `b` et `window` dans `w` , ce qui nous permettra de ne les écrire qu'une fois (et dans dans le cas de `document.body` , d'éviter de rechercher `body` dans `document` à chaque fois).
On a donc ce bookmarklet :

Code : JavaScript

```
javascript:(function(){var
w=window,b=document.body;w.onclick=w.oncontextmenu=b.onclick=b.oncontextmenu=func
```

Afficher le code HTML généré par le javascript d'une page

Pour afficher du code, le plus simple est (à l'heure où j'écris ce tutoriel) de le mettre dans un textarea.

C'est donc ce que nous allons faire.
On va faire en sorte que le `innerHTML` du corps de la page (`document.body`) soit entouré par **`<textarea>`** et **`</textarea>`**.

On fait donc :
```
javascript:document.body.innerHTML='<textarea>'+document.body.innerHTML+'</textarea>'
```
Mais là, le code est affiché dans un textarea minuscule.
On va donc lui mettre un attribut `style` et écrire dedans que la largeur et la hauteur sont égales à 100% de celles de l'élément parent :
```
javascript:document.body.innerHTML='<textarea
style="width:100%;height:100%;">'+document.body.innerHTML+'</textarea>'
```
Et voilà le tour est joué

Les exemples suivants s'appliquant au SdZ et le SdZ utilisant Jquery, les codes suivants utiliseront JQuery.
La connaissance de JQuery n'est pas forcément necessaire à la compréhension de ces codes dans la mesure où nous ne ferons que réutiliser des morceaux de code mais peut tout de même servir.
Je vous conseille cependant de lire le tutoriel de MisterDo et tit_toinou : http://www.siteduzero.com/tutoriel-3-1 [...] ire-plus.html.

Vider son pense-bête

Si vous tenez à ce que contient votre pense-bête, pensez à sauvegarder son contenu quelque part, car on ne va pas seulement vider le pense-bête sur la page actuelle, mais bien dans la BDD du SdZ !

Vous avez sans doute souvent plein de choses qui datent dans votre pense-bête sur le SdZ...
Et vous avez la flemme d'aller l'éditer...
Eh bien vous pouvez le vider avec un "simple" bookmarklet.

Tout d'abord, étudions le code qui permet au SdZ de proposer l'édition du pense-bête sans recharger la page :

Code : JavaScript

```
$("#menu_notepad h5 a").click(function(f){var h="<p class=&#92;'centre&#92;'><img
src=&#92;'Templates/images/designs/2/loading.gif&#92;' /></p>";f.preventDefault()
do=xml_edit_notepad",{to_unparse:$.trim($("#text_notepad").html())},function(j){f
id="text_notepad_modify" onblur="'+g+'">'+j+"</textarea>");$("#text_notepad_modif
e="$.trim($('#text_notepad_modify').val())";var g="event.preventDefault();$.post
function(data2){$('#text_notepad').html(data2);});$('#text_notepad').html('"+h+"'
```

Et là, vous vous dites : "Cool, c'est bien parti ".
Ne vous inquiétez pas, ce n'est pas long d'indenter et d'arriver à un code lisible. Mais avant cela, nous allons vérifier que c'est bien ce code :

Code : JavaScript

```
javascript:$("#menu_notepad h5 a").click()
```

Eh oui, c'est bien le bon code
On indente donc :

Code : JavaScript

```
$("#menu_notepad h5 a").click(//on définit le onclick
 function(f){//c'est une fonction anonyme
  var h="<p class=&#92;'centre&#92;'><img
src=&#92;'Templates/images/designs/2/loading.gif&#92;' /></p>";//on
définit le code HTML à mettre pendant le chargement
  f.preventDefault();//on évite que ça fasse des trucs implicites
```

```
donc on a pas besoin
  $.post(//on post qqch en ajax
   "ajax.php?do=xml_edit_notepad", //l'url
   {
    to_unparse:$.trim($("#text_notepad").html())//les valeurs
envoyées (à savoir le HTML de votre notepad qui va être déparsé
puis qui vous sera renvoyé comme zcode)
   },
   function(j){//la fonction de callback :
    f.preventDefault();//pareil, on évite les trucs par défaut qui
peuvent nous emm*****
    $("#text_notepad").html('<textarea id="text_notepad_modify"
onblur="'+g+'">'+j+"</textarea>");//on met un textarea dans le
notepad avec comme contenu le retour (le HTML déparssé et redevenu
du zcode) et on met en onblur une fonction qui aura été définie
[elle est plus bas dans le code])
    $("#text_notepad_modify").focus()//on met le focus sur le
notepad
   }
  );
  var e="$.trim($('#text_notepad_modify').val())";//on vire les
espaces en début et fin de ligne. (enfin on stocke ça dans une
chaîne pour la rajoute r dans du HTML plus tard
  //là, on a une giga chaîne... je la met en plusieurs ligne pour
pouvoir détailler.
  //mais en théorie, c'est une seule ligne. C'est ce qui est mis au
onblur au textarea d'édition
  var g="
event.preventDefault(); //tjrs pareil
$.post(//tjrs pareil
'ajax.php?do=xml_edit_notepad', //l'url
{new_notepad:"+e+"}, //les données envoyées
function(data2){ //la fonction callback
$('#text_notepad').html(data2); //qui met le retour dans le notepad
}
);
$('#text_notepad').html('"+h+"');//on met le code HTML du chargement
en attendant le callback.
"
 }
);
```

Bon... à ce stade-là, le code ne sert à rien...
On va donc changer quelques trucs.

Premièrement,

Code : JavaScript

```
$("#menu_notepad h5 a").click(
 function(f){
```

ne nous sert à rien car on n'a pas besoin de rajouter le onclick mais d'effectuer directement la fonction donc on l'enlève.

Ensuite,

Code : JavaScript

```
f.preventDefault();
$.post(
 "ajax.php?do=xml_edit_notepad",
 {
  to_unparse:$.trim($("#text_notepad").html())
 },
 function(j){
  f.preventDefault();
  $("#text_notepad").html('<textarea id="text_notepad_modify"
onblur="'+g+'">'+j+"</textarea>");
```

```
    $("#text_notepad_modify").focus()
  }
);
```

ne nous sert pas dans la mesure où on ne veut pas éditer mais simplement envoyer les données à modifier.

Enfin,

Code : JavaScript

```
var e="$.trim($('#text_notepad_modify').val())";
```

ne nous sert pas car on va envoyer une valeur vide...
Il est donc inutile d'enlever les espaces en début ou fin de ligne...
On va donc le supprimer et comme on ne veut pas insérer la valeur du **<textarea>** mais rien, on va remplacer la variable `e` par `''` (une chaîne vide).

Il nous reste :

Code : JavaScript

```
var h="<p class=&#92;'centre&#92;'><img
src=&#92;'Templates/images/designs/2/loading.gif&#92;' /></p>";
var g="
event.preventDefault();
$.post(
'ajax.php?do=xml_edit_notepad',
{new_notepad:"+''+"},
function(data2){
$('#text_notepad').html(data2);
}
);
$('#text_notepad').html('"+h+"');
"
```

Vu qu'on ne va pas y mettre directement dans du HTML, on va les sortir des variables et on va placer `h` (le HTML pour le chargement) à l'endroit où il était normalement ajouté par le biais de sa variable.
On n'oublie bien sûr pas de remplacer `{new_notepad:"+''+"},` par `{new_notepad:''},` .
On obtient :

Code : JavaScript

```
event.preventDefault();
$.post(
 'ajax.php?do=xml_edit_notepad',
 {new_notepad:''},
 function(data2){
  $('#text_notepad').html(data2);
 }
);
$('#text_notepad').html('<p class="centre"><img
src="Templates/images/designs/2/loading.gif" /></p>');
```

Le `event.preventDefault();` ne nous sert à rien donc on l'enlève :

Code : JavaScript

```
$.post(
 'ajax.php?do=xml_edit_notepad',
 {new_notepad:''},
 function(data2){
  $('#text_notepad').html(data2);
 }
);
```

```
$('#text_notepad').html('<p class="centre"><img
src="Templates/images/designs/2/loading.gif" /></p>');
```

On test... et là, magie, le pense-bête se vide !
On a cependant un problème : Si on laisse le pense-bête tel quel, il va disparaître à la prochaine réactualisation de la page.
On va donc remplacer la chaîne vide par une chaîne contenant un espace :

Code : JavaScript

```
$.post(
 'ajax.php?do=xml_edit_notepad',
 {new_notepad:' '},
 function(data2){
  $('#text_notepad').html(data2);
 }
);
$('#text_notepad').html('<p class="centre"><img
src="Templates/images/designs/2/loading.gif" /></p>');
```

Et voilà, vous pouvez vider votre pense-bête en un clic
(D'après mes calculs, vu le temps que vous avez mis à créer ce bookmarklet, si vous venez une fois par jour sur le SdZ et en vidant votre pense-bête en arrivant, ça deviendra rentable niveau temps si vous vivez encore plus de 100 ans)

Q.C.M.

Le premier QCM de ce cours vous est offert en libre accès.
Pour accéder aux suivants

Connectez-vous Inscrivez-vous

Lequel de ces mots ne désigne pas forcément un bookmarklet ?

- ◯ marque-page scripté
- ◯ signet
- ◯ favelet

Laquelle de ces méthodes ne marche pas sous tous les navigateurs ?

- ◯ /*code*/;return undefined;
- ◯ /*code*/;void 0
- ◯ (function(){/*code*/})()
- ◯ /*code*/;void(0)
- ◯ void(/*code*/)

Devant quoi peut-on enlever les ; pour raccourcir le bookmarklet ?

- ◯ ,
- ◯ ;
- ◯ (
- ◯ +
- ◯ -
- ◯ *
- ◯ {
- ◯ }

Quel avantage, autre que d'empêcher de voir une valeur de retour s'afficher à l'écran, ont les closures ?

- ◯ Elles sont plus esthétiques.
- ◯ Elles permettent une exécution plus rapide du bookmarklet.
- ◯ Elles évitent que les variables soient globales.
- ◯ Elles font plus "pro".

- ◯ Elles évitent les bugs dans le bookmarklet.

Correction !

Statistiques de réponses au QCM

Je tiens à remercier les validateurs pour avoir validé ce tutoriel (et plus particulièrement Thunderseb qui l'a fait le premier) ainsi que tit_toinou et Golmote pour m'avoir donné des avis externes, conseillé et vous avoir évité de voir des abominations syntaxiques, orthographiques et autres et enfin Tiller pour m'avoir fait remarquer que le `return undefined;` ne marchait pas partout.

Voilà, vous savez tout
Ceci est mon premier tutoriel, donc soyez indulgents dans les commentaires

Partager