


La gestion des cookies en JavaScript

par Peter-Paul Koch ([Auteur](#)) Didier Mouronval ([Traducteur](#))

Date de publication : 15 janvier 2010


Dernière mise à jour :

Cet article est la traduction de :  **Cookies**.

Dans cet article, je vais vous présenter trois fonctions permettant d'enregistrer, lire et supprimer des  **cookies**. Ces fonctions vous permettront d'utiliser facilement les *cookies* sur votre site.


Tout d'abord, je vous présenterai rapidement ce qu'est un cookie ainsi que l'objet JavaScript `document.cookie`, suivi d'un exemple. Ensuite, je vous proposerai ces trois fonctions en expliquant leur fonctionnement.

Le script, écrit par **Scott Andrew**, a été copié et modifié avec son autorisation.

Retrouvez plus d'informations sur les cookies dans la partie 6G  **de mon livre**.

Les cookies.....	3
Fonctionnement des cookies.....	3
Paire nom / valeur.....	3
Date d'expiration.....	3
Domaine et arborescence.....	4
document.cookie.....	4
Exemple.....	5
Les scripts.....	5
Explication du code.....	6
createCookie.....	6
readCookie.....	7
eraseCookie.....	8
Remerciements.....	8

Les cookies

Les cookies ont été inventés par Netscape afin de donner une "mémoire" aux serveurs et navigateurs Web. Le protocole  **HTTP**, qui gère le transfert des pages Web vers le navigateur ainsi que les demandes de pages du navigateur vers le serveur, est dit *state-less* (sans état) : cela signifie qu'une fois la page envoyée vers le navigateur, il n'a aucun moyen d'en garder une trace. Vous pourrez donc venir deux, trois, cent fois sur la page, le serveur considérera toujours qu'il s'agit de votre première visite.

Cela peut être gênant à plusieurs titres : le serveur ne peut pas se souvenir si vous êtes authentifié à une page protégée, n'est pas capable de conserver vos préférences utilisateur, etc. En résumé, il ne peut se souvenir de rien ! De plus, lorsque la personnalisation a été créée, cela est vite devenu un problème majeur.

Les cookies ont été inventés pour remédier à ces problèmes. Il existe d'autres solutions pour les contourner, mais les cookies sont très simples à maintenir et très souples d'emploi.

Fonctionnement des cookies

Un cookie n'est rien d'autre qu'un petit fichier texte stocké par le navigateur. Il contient certaines données :

- 1 Une paire nom / valeur contenant les informations.
- 2 Une date d'expiration au-delà de laquelle il n'est plus valide.
- 3 Un domaine et une arborescence qui indiquent quel répertoire de quel serveur y aura accès.

Dès que vous demandez une page Web à laquelle le cookie peut être envoyé, celui-ci est ajouté dans l'en-tête HTTP. Les programmes côté serveur peuvent alors le lire et décider, par exemple, si vous avez le droit de voir la page ou si vous voulez que les liens soient jaunes avec un fond vert.

Ainsi, chaque fois que vous visitez la page d'où vient le cookie, les informations vous concernant sont disponibles. C'est parfois bien pratique, mais cela peut aussi nuire à votre vie privée. Heureusement, la plupart des navigateurs vous permettent de gérer les cookies (vous pouvez donc supprimer ceux provenant des sites publicitaires, par exemple).

Les cookies peuvent aussi être lus par JavaScript. Ils sont principalement destinés à conserver vos préférences.

Paire nom / valeur

Chaque cookie possède un couple nom / valeur qui contient les données actuelles. Le nom du cookie est destiné à vous aider, il vous suffira de chercher ce nom lorsque vous lirez les cookies de la page.

Si vous souhaitez lire le contenu d'un cookie, commencez par chercher son nom, puis la valeur qui lui est associée. Bien sûr, c'est à vous de décider quelle(s) valeur(s) peut avoir un cookie et d'écrire des scripts pour traiter cette (ces) valeur(s).

Date d'expiration

Chaque cookie doit avoir une date d'expiration au-delà de laquelle il ne sera plus valide et supprimé. Si vous ne renseignez pas cette date, le cookie sera supprimé à la fermeture du navigateur. Cette date est exprimée au format UTC (Greenwich).

Domaine et arborescence

Chaque cookie possède également un domaine et une arborescence. Le domaine indique au navigateur à quel domaine (quel site) le cookie doit être envoyé. Si vous ne l'indiquez pas, le domaine correspondra à celui qui a créé le cookie : dans cette page, par exemple, *ppk.developpez.com*.

Notez que l'utilité du domaine est d'autoriser l'envoi du cookie aux sous-domaines associés. Mon cookie ne pourra pas être lu depuis *www.developpez.com* puisque par défaut, son domaine est *ppk.developpez.com*. Si je précise le domaine *developpez.com*, alors *www.developpez.com* pourra lui aussi lire le cookie.

Je ne peux pas affecter un domaine sur lequel je ne me trouve pas : *quirksmode.org* par exemple. Dans notre cas, seul *developpez.com* est autorisé.

L'arborescence vous permet d'indiquer un répertoire sur lequel le cookie sera actif. Par exemple, si vous voulez que le cookie ne soit envoyé que vers le répertoire *tutoriels*, spécifiez */tutoriels* comme arborescence. Habituellement, l'arborescence est */*, ce qui signifie que le cookie est valide sur l'intégralité du domaine.

C'est comme cela que fonctionnent les scripts proposés. Les cookies créés sur cette page seront envoyés sur toutes les pages de *www.developpez.com* (bien que seule cette page contienne un script permettant de les récupérer et de les utiliser).

document.cookie

Les cookies peuvent être créés, lus et supprimés par JavaScript. Ils sont accessibles à travers la propriété `document.cookie`. Vous pouvez traiter cette propriété comme une chaîne (bien que cela n'en soit pas réellement une). Vous n'avez accès qu'au couple nom / valeur.

Pour créer un cookie sur ce domaine avec une paire nom / valeur valant 'ppkcookie1=testcookie' et expirant le 28 février 2010, il faut faire :

```
document.cookie = 'ppkcookie1=testcookie; expires=Sun, 28 Feb 2010 00:00:00 UTC; path=/'
```

- 1 D'abord, la paire nom / valeur ('ppkcookie1=testcookie').
- 2 Puis un point-virgule et un espace.
- 3 La date d'expiration dans un format correct (expires=Sun, 28 Feb 2010 00:00:00 UTC).
- 4 De nouveau un point-virgule et un espace.
- 5 Le domaine (path=/).

La syntaxe est très stricte, ne la changez pas (bien sûr, les scripts vous permettront de gérer ces valeurs) !

En revanche, même si la syntaxe donne l'impression d'écrire la chaîne dans `document.cookie`, dès que vous essaierez de la lire, seul le couple nom / valeur sera accessible :

```
ppkcookie1=testcookie
```

Il est possible de créer un autre cookie avec :

```
document.cookie = 'ppkcookie2=un autre test; expires=Mon, 1 Mar 2010 00:00:00 UTC; path=/'
```

Le premier cookie n'est pas écrasé, ce qui serait le cas si `document.cookie` était une véritable chaîne. En réalité, le second cookie est ajouté à `document.cookie`. Si on le lit une nouvelle fois, on obtient donc :

```
ppkcookie1=testcookie; ppkcookie2=un autre test
```

Cependant, si l'on crée à nouveau un cookie portant le même nom que le précédent :

```
document.cookie = 'ppkcookie2=encore un autre test; expires=Mon, 1 Mar 2010 00:00:00 UTC; path=/'
```

cette fois-ci, le précédent cookie est bien écrasé, document.cookie renvoie désormais :

```
ppkcookie1=testcookie; ppkcookie2=encore un autre test
```

Pour lire un cookie, vous devrez récupérer la valeur de document.cookie et la traiter comme s'il s'agissait d'une chaîne en recherchant des caractères, le point-virgule puis le nom du cookie par exemple. Je vous montrerai comment procéder dans la suite de cet article.

Enfin, pour supprimer un cookie, il suffit de fixer sa date d'expiration à une date passée. Comme cela, le navigateur détecte qu'il n'est plus valide et le supprime :

```
document.cookie = 'ppkcookie2=encore un autre test; expires=Fri, 01 Jan 2010 00:0:00 UTC; path=/'
```

Exemple

Si ces explications vous semblent encore un peu confuses, allez à la page d'exemple. Vous pourrez y gérer deux cookies, ppkcookie1 et ppkcookie2. Saisissez leur valeur dans la zone de texte.

Accéder à la page d'exemple.

Les cookies de l'exemple sont fixés pour être valables une semaine. Si vous revisitez cette page avant ce délai, une alerte vous avertissant que le(s) cookie(s) est (sont) toujours actif(s). Essayez de retourner sur la page d'exemple.

Les scripts

Voici le code des scripts dont vous aurez besoin :

```
function createCookie(name,value,days) {
  if (days) {
    var date = new Date();
    date.setTime(date.getTime()+(days*24*60*60*1000));
    var expires = "; expires="+date.toGMTString();
  }
  else var expires = "";
  document.cookie = name+"="+value+expires+"; path=/";
}

function readCookie(name) {
  var nameEQ = name + "=";
  var ca = document.cookie.split(';');
  for(var i=0;i < ca.length;i++) {
    var c = ca[i];
    while (c.charAt(0)==' ') c = c.substring(1,c.length);
    if (c.indexOf(nameEQ) == 0) return c.substring(nameEQ.length,c.length);
  }
  return null;
}

function eraseCookie(name) {
  createCookie(name,"",-1);
}
```

Explication du code

Les fonctions ne sont pas très compliquées. La principale difficulté réside dans l'écriture de la bonne syntaxe de création du cookie.

createCookie

Lorsque vous appelez `createCookie()`, vous devez lui passer trois paramètres : le nom et la valeur du cookie ainsi que le nombre de jours pendant lequel il restera actif. Voici un exemple pour lequel `ppkcookie=testcookie` sera actif 7 jours :

```
createCookie('ppkcookie','testcookie',7)
```

Si le nombre de jours vaut 0, le cookie sera effacé à la fermeture du navigateur. Si vous mettez un nombre négatif de jours, le cookie sera effacé immédiatement.

La fonction reçoit donc ses paramètres et commence son exécution :

```
function createCookie(name,value,days) {
```

Tout d'abord, elle vérifie que le paramètre `days` est présent. Si ce n'est pas le cas, nous n'avons pas besoin de faire de calculs sur la date :

```
if(days) {
```

Si le paramètre existe, nous créons un nouvel objet `Date()` contenant la date du jour :

```
var date = new Date();
```

Puis nous récupérons le temps actuel (en millisecondes) auquel on ajoute le nombre de jours (millisecondes également). Nous affectons alors à la propriété `time` de la variable `date` la valeur calculée. `date` a maintenant pour valeur la date en millisecondes à laquelle le cookie doit expirer :

```
date.setTime(date.getTime()+(days*24*60*60*1000));
```

Ensuite, nous affectons cette date au format UTC/GMT à la variable `expires` :

```
var expires = ""; expires="+date.toGMTString();
```

Si 0 est passé comme paramètre, `expires` n'est pas définie, ce qui provoque l'effacement du cookie à la fermeture du navigateur :

```
else var expires = "";
```

Enfin, nous écrivons le nouveau cookie dans `document.cookie` avec la syntaxe adéquate :

```
document.cookie = name+"="+value+expires+"; path=/";
```

Le cookie est désormais créé.

readCookie

Pour lire un cookie, nous appelons `readCookie()` à laquelle nous passons en paramètre le nom du cookie. Affectez le résultat de la fonction à une variable et testez si la variable a une valeur (si le cookie n'existe pas, la fonction renvoie null, ce qui peut perturber votre propre fonction), puis effectuez vos traitements :

```
var x = readCookie('ppkcookie1')
if (x) {
  // Traitement de la valeur du cookie
}
```

La fonction reçoit donc son paramètre et commence son exécution :

```
function readCookie(name) {
```

Nous allons commencer par chercher le nom du cookie suivi du signe "=". Stockons cette chaîne dans la variable `nameEQ` :

```
var nameEQ = name + "=";
```

Ensuite, effectuons un `split(';')` sur `document.cookie` : ainsi la variable `ca` contient un tableau de tous les cookies de ce domaine et de cette arborescence :

```
var ca = document.cookie.split(';');
```

Nous parcourons alors ce tableau (nous bouclons sur tous les cookies trouvés) :

```
for(var i=0;i < ca.length;i++) {
```

Appelons `c` le cookie en cours :

```
var c = ca[i];
```

Si le premier caractère est un espace, nous le retirons à l'aide de la méthode `substring()`. Nous réitérons cette étape tant que le premier caractère est un espace :

```
while (c.charAt(0)==' ') c = c.substring(1,c.length);
```

Maintenant, `c` commence par le nom du cookie courant. Si c'est le nom que nous cherchons :

```
if (c.indexOf(nameEQ) == 0)
```

alors nous avons trouvé notre cookie. Nous voulons retourner la valeur du cookie, qui correspond à la partie de `c` qui vient juste après `nameEQ`. Nous retournons donc cette valeur et sortons de la fonction : sa mission est terminée.

```
if (c.indexOf(nameEQ) == 0) return c.substring(nameEQ.length,c.length);
```

Si, après avoir passé tous les cookies en revue, nous n'avons pas trouvé le nom recherché, cela signifie que le cookie n'existe pas. Nous retournons alors null :

```
return null;
```

eraseCookie

Effacer un cookie est très simple.

```
eraseCookie('ppkcookie');
```

Il suffit de passer à la fonction le nom du cookie à effacer :

```
function eraseCookie(name) {
```

puis appelons la fonction createCookie() pour affecter au cookie une date d'expiration passée correspondant à la veille :

```
createCookie(name, "", -1);
```

Le navigateur se rend compte que le cookie n'est plus valide et l'efface immédiatement.

Remerciements

Je tiens à remercier [eusebe19](#) et [Wachter](#) dont les relectures attentives ont permis d'améliorer significativement la qualité de cet article.