1

Les bases de JavaScript

Ce chapitre traite de certaines bases concernant JavaScript. Il ne s'intéresse pas explicitement à la syntaxe du langage, un aspect abordé dans suffisamment de didacticiels et d'ouvrages et qui n'entre pas dans l'objectif de ce livre. Toutefois, nous expliquerons en détail des aspects essentiels comme l'insertion de code JavaScript dans une page. Nous ferons également un peu d'histoire et relaterons les faits de guerre des navigateurs pour vous préparer au chapitre suivant.

Compréhension de JavaScript (et de son histoire)

JavaScript est un langage de script côté client, ce qui signifie qu'il s'exécute côté client, dans un navigateur Web. JavaScript peut aussi être employé côté serveur et en dehors d'un navigateur, mais ce n'est pas l'objet de cet ouvrage. Si le navigateur est compatible, JavaScript donne accès à la page en cours et permet au script de déterminer les propriétés du client, de rediriger l'utilisateur vers une autre page, d'accéder aux cookies, etc.

JavaScript naît en septembre 1995, parallèlement à la sortie de la version 2.0 du navigateur Netscape, la première à être dotée du langage de script. À cette époque, le langage s'appelle Mocha puis, à sa sortie, LiveScript ; Netscape conclut ensuite un accord marketing avec Sun (le créateur de Java) et décide de renommer le langage en décembre de cette année, il devient JavaScript.

Le concept connaît immédiatement du succès : Microsoft en intègre une prise en charge dans Internet Explorer versions 3 et suivantes (au milieu de l'année 1996). Pour des raisons légales, une mouture de Microsoft du langage est alors appelée JScript. JScript était plus ou moins compatible avec JavaScript mais a commencé à inclure des fonctions supplémentaires, spécifiques à Internet Explorer (ce qui, à quelques exceptions près, ne s'est jamais véritablement installé).

En 1997, est publiée la norme ECMAScript (ECMA-262) ; JavaScript en est donc la première implémentation. La norme ne précise que le langage et non les fonctions des hôtes environnants (par exemple, comment accéder à la fenêtre courante du navigateur ou en ouvrir une nouvelle). ECMAScript devient norme ISO en 1998.

Aux alentours de 1997 ou 1998, la guerre des navigateurs entre Netscape et Microsoft atteint son apogée, les deux éditeurs ajoutant une nouvelle fonctionnalité incompatible avec la version 5 de leurs navigateurs. Le reste n'est qu'histoire : Netscape abandonne l'idée de publier une version 5 du navigateur et décide de tout recommencer avec Netscape 6 ; Internet Explorer peut ainsi augmenter ses parts de marché, passant à plus de 90 %. Il a d'ailleurs fallu plusieurs années au projet Mozilla, alors en cours de création, pour revenir à la vie. À noter que le navigateur Firefox est fondé sur Mozilla et commence alors à gagner des parts de marché sur Microsoft.

La situation s'est considérablement accélérée par la suite, avec l'apparition de nouveaux navigateurs Web modernes, en particulier Google Chrome, et surtout la mise en place progressive du "Web 2.0". Derrière ce concept se rangent des milliers d'applications Web d'un genre nouveau, imitant l'interface et le comportement d'applications bureautiques standard. JavaScript n'est pas étranger à cette rapide montée en puissance du Web, en soutenant en particulier un véritable "dialogue" avec le serveur : l'ensemble de technologies AJAX et les requêtes asynchrones qu'il introduit renforcent le dynamisme des sites Web, en mettant à jour ponctuellement des éléments de contenu sans rafraîchir des pages entières.

Méprisé à ses débuts par les développeurs "professionnels", à cause de son champ d'action balbutiant et limité, JavaScript s'est ainsi définitivement érigé au rang des langages de référence du Web. De très nombreuses bibliothèques libres d'utilisation ont ainsi vu le jour, facilitant les tâches les plus récurrentes, parmi lesquelles jQuery constitue la figure de proue. Plus récemment encore, l'apparition de HTML5 plébiscite l'utilisation de JavaScript à travers de nouvelles API favorisant le stockage local de données, la lecture de contenus multimédias ou encore le dessin dans le cadre du navigateur.

En marge, l'essor des smartphones et des tablettes Internet a fait naître une grande variété de navigateurs Web mobiles ; là encore, vos possibilités en tant que développeur se démultiplient et vous devez prendre en compte les spécificités de ces appareils d'un type nouveau à travers l'ensemble de vos projets. En 2013, d'après Yahoo! Developer Network, moins de 1 % des navigateurs Web du marché, qu'ils soient bureautiques ou mobiles, ne prennent pas en charge JavaScript.

Établissement d'un système de test

Nous venons de le mentionner, l'immense majorité des navigateurs accepte JavaScript. Mais si leurs éditeurs sont généralement parvenus à un consensus, en respectant les mêmes standards, ils n'interprètent pas toujours l'ensemble des fonctionnalités JavaScript de la même manière. C'est notamment le cas des navigateurs mobiles, encore jeunes et disparates, pour lesquels vous devez méticuleusement vérifier s'ils prennent en charge vos développements.

Quelle est donc la meilleure stratégie pour tester un site Web sur un maximum de systèmes, avec le moins d'efforts possible ?

En fait, tout dépend du public auquel s'adresse votre site Web. Si vous ciblez principalement des utilisateurs de Mac, vous devrez procéder à un test important sur le navigateur Safari, puisqu'il est livré par défaut avec les versions récentes de Mac OS X.

Quel que soit le type de site Web utilisé, Internet Explorer bénéficie toujours d'une très forte part de marché, même sur des sites Web plutôt centrés sur l'open source. Pour le test, il vous faut donc Internet Explorer et un système Windows (ou au moins un élément du style Virtual PC ou VMware, avec une machine virtuelle Windows). Tous les navigateurs Mozilla partagent la même base de code pour le rendu et pour JavaScript, peu importe donc la plate-forme que vous utilisez (même s'il existe des différences minimes). Vous pourriez, par exemple, utiliser Firefox sur la machine Windows sur laquelle réside également votre installation Internet Explorer. Il en va de même pour Google Chrome, désormais incontournable, et qui se décline pour l'ensemble des plates-formes du marché, y compris les appareils mobiles.

Vous avez donc intérêt à installer l'ensemble des navigateurs du marché sur une machine de test, afin de vérifier

par la pratique le rendu de vos projets. Pour les plates-formes plus exotiques, vous pouvez vous reporter à des services en ligne comme BrowserStack (**http://www.browserstack.com/**). Au terme de la période d'évaluation, il vous en coûtera 19 dollars par mois pour disposer de l'ensemble des navigateurs Web du marché, y compris leurs déclinaisons mobiles, sous forme de machines virtuelles.

Enfin, n'oubliez pas de tester votre site Web lorsque JavaScript est activé et désactivé dans les navigateurs. Comme nous l'avons vu, cette situation est devenue anecdotique mais vous devez envisager une solution de repli face à ce genre de cas de figure.

Configuration de navigateurs Web

Par défaut, la plupart des navigateurs Web activés pour JavaScript prennent en charge ce langage. D'ailleurs, la toute première version de Netscape à accepter JavaScript ne disposait même pas d'une fonction pour le désactiver !

Il est toutefois possible de le désactiver, vous devez donc trouver comment simuler cette situation (et comment demander aux utilisateurs de l'activer). Cela dépend non seulement des navigateurs utilisés mais parfois aussi de la version du navigateur. Dans le navigateur Firefox, JavaScript peut être activé dans le menu Options, Contenu, Activer JavaScript. Sous Internet Explorer, vous devez creuser un peu plus. Cliquez sur Outils, Options Internet, Sécurité, zone Internet, Personnaliser le niveau, Script, Active Scripting, Activer.

Astuce

Internet Explorer possède une fonction de sécurité qui empêche l'exécution de JavaScript sur les pages locales (voir Figure 1.1). Cela est en fait assez utile mais peut se révéler ennuyeux lorsque vous testez une application. Il est possible de contourner ce problème de deux manières : utilisez un serveur Web local pour tester votre application ou désactivez simplement le message d'erreur en choisissant Outils, Options Internet, Avancé, Sécurité, Autoriser l'exécution du contenu actif dans les fichiers de mon ordinateur.

Figure 1.1 : message d'erreur assez ennuyeux avec JavaScript sur les pages locales.

Inclusion du code JavaScript

```
<script type="text/JavaScript">
  window.alert("Bienvenue dans JavaScript !");
</script>
```

Le code JavaScript peut se présenter de deux manières : intégré dans une page HTML ou intégré dans un fichier externe. La manière la plus fréquente de l'inclure consiste à utiliser les éléments `<script>`.Vous pouvez le placer n'importe où, il est

alors exécuté après que cette partie de la page HTML a été chargée et analysée. Le code qui précède (fichier `script.html` dans l'archive à télécharger) ouvre une fenêtre modale et affiche un texte plutôt simple.

L'attribut `type` prévoit le type MIME pour JavaScript. Précédemment, on utilisait `language="JavaScript"` ; toutefois, puisque ce n'était pas standardisé, il vaut mieux utiliser `type` et le type MIME à la place. Dans cet ouvrage, nous suivons la méthode adoptée par de nombreux sites Web de nos jours : on utilise à la fois `type` et `language`.

De même, par le passé, il était possible de cibler un script sur un numéro de version spécifique de JavaScript, comme ceci :

```
<script language="JavaScript1.6">
   window.alert("seulement avec JavaScript 1.6 !");
</script>
```

Ce n'est quasiment plus usité. L'implémentation de cette fonction génère un certain nombre de bogues dans les navigateurs, et il existe de meilleures manières de tester les capacités JavaScript d'un navigateur.

Info

Dans certains anciens didacticiels, vous trouverez des conseils pour utiliser les commentaires HTML présentés de la façon suivante :

```
<script language="JavaScript"><!--
  // ...
//--></script>
```

Cela servait auparavant à gérer les navigateurs qui ne savaient rien de JavaScript. Or, même ceux qui n'acceptent pas JavaScript connaissent l'élément `<script>` et savent l'ignorer (lui et son contenu). Ces commentaires HTML ne sont donc plus nécessaires.

2

Expressions communes

Certaines tâches JavaScript récurrentes doivent être réalisées quasiment chaque jour. Elles sont à la base de nombreuses applications JavaScript mais n'entrent dans aucune catégorie particulière. Nous présentons donc au cours de ce chapitre une suite de problèmes fréquents, avec leurs solutions.

Détection du type de navigateur

```
window.alert(navigator.appName);
```

Même si les implémentations de navigateurs par JavaScript sont, de nos jours, assez compatibles les unes avec les autres (en particulier si on se souvient de l'époque de la guerre des navigateurs, à la fin des années 1990), la détection du type de navigateur constitue une partie essentielle de la boîte à outils du développeur JavaScript.

L'objet JavaScript `navigator` propose des informations sur le navigateur. Sa propriété `userAgent`, qui contient la chaîne d'identification complète du navigateur, est très utile mais parfois difficile à analyser. Elle est aussi envoyée dans l'en-tête `User-Agent` HTTP de chaque requête.

Pour déterminer simplement le type d'un navigateur, il suffit d'employer la propriété `appName`, comme le montre le code précédent. Le Tableau 2.1 contient les valeurs `appName` des navigateurs les plus importants.

Tableau 2.1 : valeurs `appName` pour divers navigateurs

Navigateur	`appName`
Internet Explorer	`Microsoft Internet Explorer`
Navigateurs Mozilla	`Netscape`
Konqueror (KDE)	`Konqueror`
Apple Safari	`Netscape`
Navigateur Opera	`Opera`
Google Chrome	`Netscape`

Vous le voyez, les navigateurs Safari et Chrome renvoient un nom incorrect. Pour contrebalancer cet effet, vous pouvez rechercher `navigator.userAgent` pour certains types de navigateurs. Et puisque le navigateur Opera risque d'être mal identifié (bien qu'il stocke `"Opera"` dans `navigator.userAgent`), il convient de le vérifier en premier.

```
<script language="JavaScript"
  type="text/JavaScript">
var uA = navigator.userAgent;
var browserType = "unknown";
if (uA.indexOf("Opera") > -1) {
  browserType = "Opera";
} else if (uA.indexOf("Safari") > -1) {
  browserType = "Safari";
} else if (uA.indexOf("Konqueror") > -1) {
  browserType = "Konqueror";
} else if (uA.indexOf("Gecko") > -1) {
  browserType = "Mozilla";
} else if (uA.indexOf("MSIE") > -1) {
  browserType = "Internet Explorer";
}
window.alert(browserType);
</script>
```

Détermination du type de navigateur (navigateur.html)

Avec quelques efforts de plus, ce script peut être prolongé pour distinguer les dérivés de Mozilla (Firefox, Epiphany, Galeon, Camino, SeaMonkey, etc.).

Détection du numéro de version du navigateur

Pour déterminer le numéro de version du navigateur, il existe plusieurs manières. La plupart du temps, vous devez rechercher `navigator.userAgent`, ce qui peut ressembler à ceci :

```
Mozilla/5.0 (Windows; U; Windows NT 5.1; en;
     rv:1.8.0.3) Gecko/20060426 Firefox 1.5.0.3
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US;
     rv:1.4) Gecko/20030619 Netscape/7.1 (ax)
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;
     SV1; .NET CLR 1.0.3705; .NET CLR 1.1.4322;
     .NET CLR 2.0.50727)
Mozilla/5.0 (compatible; Konqueror/3.4; FreeBSD)
     KHTML/3.4.2 (like Gecko)
Mozilla/5.0 (Macintosh; U; Intel Mac OS X; en)
     AppleWebKit/418 (KHTML, like Gecko)
     Safari/417.9.3
Mozilla/5.0 (Macintosh; U; PPC Mac OS X; en)
     AppleWebKit/312.8 (KHTML, like Gecko)
     Safari/312.6
Opera/9.00 (Windows NT 5.1; U; en)
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;
     en) Opera 9.00
```

Vous le voyez, selon le type du navigateur, le numéro de version est enfoui ailleurs dans la valeur de `navigator.userAgent`. Vous aurez alors la tâche fastidieuse de traiter tous les navigateurs et de vous tenir au courant des nouvelles méthodes. Il existe toutefois quelques ressources Web pour implémenter une détection des navigateurs et qui sont assez bien faites. Vous trouverez de la documentation et des codes sur ces sites Web :

- http://www.webreference.com/tools/browser/JavaScript-020214.html ;
- http://www.gemal.dk/browserspy/basic.html.

Vérification des capacités du navigateur

```
if (document.getElementById) {
  // ...
```

Vous le voyez dans l'exemple précédent, se fier aux numéros de version des navigateurs n'est pas seulement difficile, c'est également gager les possibilités d'évolution. Il vaut mieux vérifier spécifiquement la prise en charge des objets spéciaux.

Par exemple, pour utiliser DOM (voir Chapitre 5, "DOM"), vous pourrez tenter d'utiliser le code précédent. Si la méthode `getElementById()` est implémentée, `document.getElementById` (sans parenthèses) renvoie une référence à la fonction. Si elle est utilisée dans une condition, elle renvoie `true`. Le code associé est alors exécuté.

Autre exemple : Internet Explorer accepte les objets ActiveX pour certaines applications, par exemple la prise en charge du XML. Toutefois, seules les versions Windows de IE connaissent ActiveX. La vérification systématique pour Internet Explorer génère donc des problèmes pour les utilisateurs de Mac. Dès lors, si vous vérifiez spécifiquement la prise en charge d'ActiveX, vous éviterez ces problèmes :

```
if (window.ActiveXObject) {
  // ...
}
```

Empêcher la mise en cache

```
document.write("<img src=\"image.png?" +
  Math.random() + "\" />");
```

Grâce aux en-têtes côté serveur, il est possible d'éviter la mise en cache d'images de type "contenu dynamique" ainsi que de pages HTML. Mais, la méthode n'est pas infaillible,

puisque certains navigateurs ou serveurs proxy peuvent ignorer ces paramètres. Pour y remédier, ajoutez un paramètre de chaîne de requête insignifiant à l'URL, comme dans l'extrait suivant : `Math.random()` renvoie un nombre aléatoire compris entre 0 et 1, par exemple `0,1296601696732852`. L'annexer à une image ne modifie généralement pas les données envoyées du serveur, mais la requête est totalement nouvelle pour le navigateur. L'image (ou les autres données) n'est donc pas mise en cache.

Redirection du navigateur

```
location.href = "nouvellePage.html";
```

La propriété `location.href` permet un accès en lecture et en écriture à l'URL de la page en cours. Conséquence, régler `location.href` sur une autre valeur redirige le navigateur, qui charge alors la nouvelle page, comme le montre le code précédent.

Astuce

Cela peut également être réalisé avec du HTML :

```
<meta http-equiv="Refresh" content="X; URL=Y" />
```

L'emplacement X désigne le nombre de secondes à patienter avant que la nouvelle page ne soit chargée ; Y indique la nouvelle URL.

La page précédente arrive ensuite dans l'historique du navigateur. Si vous souhaitez remplacer l'ancienne page dans l'historique (pour que le bouton Précédent ne fonctionne pas comme l'on s'y attendrait ici), utilisez la méthode `location.replace()` :

```
location.replace("nouvellePage.html");
```

Demande de confirmation à l'utilisateur

```
<a href="unePage.html"
  onclick="return window.confirm('En êtes-vous sûr ?');">
  Cliquez ici</a>
```

JavaScript propose une prise en charge limitée des fenêtres modales. La méthode `window.alert()` est assez commune mais il existe d'autres options. Avec `window.confirm()`, l'utilisateur se voit présenter une fenêtre de type OK/Annuler. S'il clique sur OK, `window.confirm()` renvoie `true`, et `false` dans le cas contraire. Le code qui précède (fichier `confirmation.html`) l'utilise comme valeur de retour pour un lien. Ainsi, si l'utilisateur clique sur Annuler, le navigateur ne suit pas le lien.

Attention

Sachez que cette boîte de dialogue est traduite par les navigateurs, vous devez donc éviter d'inscrire un texte du style "Cliquez sur Annuler pour..." car les personnes disposant d'un système étranger risquent de ne pas voir s'afficher de bouton Annuler.

Demande de données utilisateur

```
var nom =
  window.prompt("Saisissez votre nom", "<Votre nom>");
```

La méthode `window.prompt()` permet aux utilisateurs de saisir du texte dans un champ de texte d'une seule ligne (voir Figure 2.2). Ces informations correspondent à la valeur de retour de l'appel de méthode et peuvent ensuite être utilisées dans le script.

```
<script language="JavaScript" type="text/JavaScript">
var nom = window.prompt("Saisissez votre nom", "<Votre
nom>");
if (nom != null) {
  window.alert("Bonjour, " + nom + " !");
}
</script>
```

Demande de données utilisateur (prompt.html)

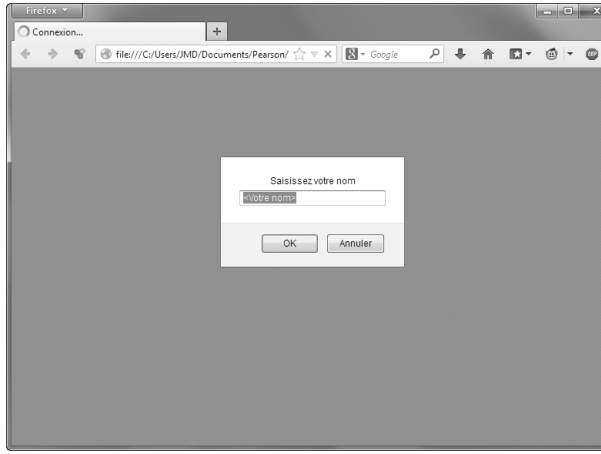

Figure 2.2 : la boîte de saisie générée par window.prompt().

Info

Sachez que si l'utilisateur clique sur le bouton Annuler ou appuie sur la touche d'échappement, `window.prompt()` renvoie `null`. Le code qui précède vérifie que, si l'utilisateur clique sur le bouton OK, les données saisies sont affichées.