

Les objets du navigateur

Formation

Développeur Web et Web Mobile

Module : 03

Développer la partie front-end d'une application web

Séquence : 03

Développer une interface utilisateur web dynamique

Séance : 03

Développer des scripts clients dans une page web

Libellé réduit :	Objets du navigateur
Type de document :	Support de cours
Version :	1
Auteur :	Didier Bonneau
Centre afpa :	Créteil
Date de création :	01/01/2014
Date de mise à jour :	14/10/2022



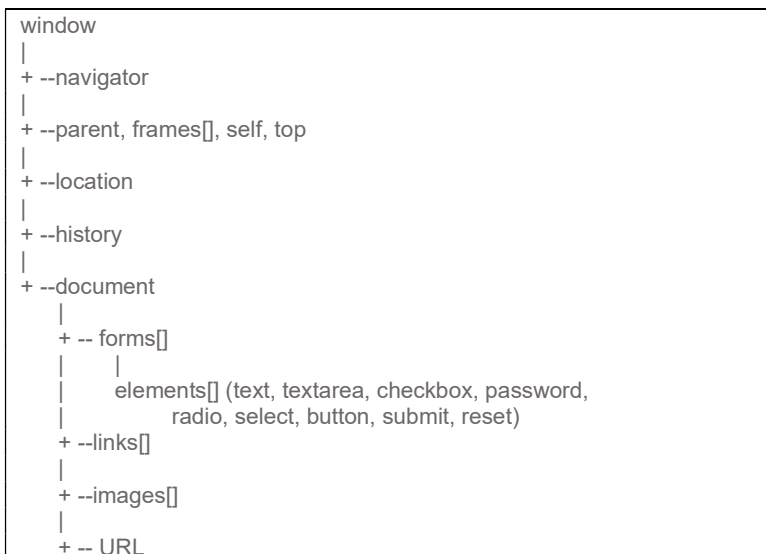
Sommaire

Contenu

I.	La hiérarchie des classes.....	2
II.	Les classes de base du navigateur.....	2
III.	L'objet « window »	3
A.	Présentation.....	3
B.	Principales propriétés de window.....	3
C.	Principales méthodes de window.....	3
D.	Gestion de fenêtres.....	4
IV.	l'objet « navigator »	5
V.	l'objet « history »	6
VI.	l'objet « location »	6
VII.	L'objet « document » (DOM 0)	6
VIII.	Evénements usuels (DOM 0)	7
A.	Syntaxe (en principe obsolète ..).....	7
B.	Tableau récapitulatif.....	8

I. LA HIERARCHIE DES CLASSES

Le navigateur lors de son lancement et du chargement d'une page Html va créer un certain nombre d'objets dont la hiérarchie est la suivante :



II. LES CLASSES DE BASE DU NAVIGATEUR

Les objets de ces classes sont automatiquement instanciés à chaque étape du fonctionnement du navigateur, par exemple lors de l'ouverture d'une fenêtre (ou de frames), le document contenu dans chaque fenêtre (ou frame) et les divers éléments (formulaires, images, liens ...) contenus dans ce dernier sont créés.

- Les applications JavaScript peuvent alors dialoguer avec ces objets visuels et les manipuler. Le programmeur peut ainsi agir sur l'état du navigateur, de ses fenêtres et de ses documents.
- Remarque : attention, cette hiérarchie d'objets n'a rien à voir avec la notion d'héritage : les objets "descendants" ne sont considérés que comme des propriétés particulières de l'objet "ancêtre". Ainsi, un objet *document* n'est pas un objet *window* particulier, mais une propriété de *window*, qui est elle-même est un objet doté de propriétés et de méthodes.
- Objets fondamentaux :
 - *window* : l'objet de plus haut niveau créé par le navigateur, c'est sa fenêtre.
 - *window.navigator* : propriété objet -> c'est le logiciel client
 - *window.location* : propriété objet -> URL de la fenêtre courante.
 - *window.history* : propriété objet -> URL précédemment visitées.
 - *window.document* : propriété objet -> document courant, dont les propriétés sont le titre, les couleurs (fond, texte, lien ...), les formulaires, les images etc..

En javascript, le mot « *window* » peut être omis pour atteindre ces propriétés :

Exemple : *navigator.appName*

III. L'OBJET « WINDOW »

A. PRESENTATION

Il s'agit de la classe située au sommet de la hiérarchie.

De façon générale, comme tout se déroule dans une fenêtre du navigateur, le nom de la fenêtre est implicite, le préfixe « window » peut être omis pour désigner un objet ou une méthode de la fenêtre courante (sauf dans un gestionnaire d'événement pour lequel l'objet courant étant un document, il faut préciser la fenêtre de ce document).

Bien entendu si la propriété ou la méthode s'adresse à une fenêtre définie par le programmeur à l'aide de la méthode open(), il faudra la préfixer par son nom.

Voici ses principales propriétés et méthodes

B. PRINCIPALES PROPRIETES DE WINDOW

- defaultStatus : représente le message de défaut qui sera affiché dans la barre de statut
`<body onLoad="defaultStatus='Bonjour à tous'">`
- status : est un message affiché dans la barre de statut de la fenêtre.
`window.status="N'oubliez pas de fermer vos fenêtres !"`
- length : représente le nombre de cadres dans la fenêtre parente (0 sinon).
- name : représente le nom de la fenêtre
- opener : spécifie le nom de la fenêtre parent, qui l'a créée dynamiquement, avec open().
- parent : est le nom de la fenêtre parente (où se trouve éventuellement la fenêtre courante)
- self : est un synonyme pour le nom de la fenêtre et fait référence à la fenêtre courante
- top : fait référence à la fenêtre principale du navigateur.
- closed : booléen qui indique si la fenêtre a été fermée.
utilisation : if (! fen.closed) fen.close();


C. PRINCIPALES METHODES DE WINDOW

- L'objet « window » est implicite, pour appeler ces méthodes on peut omettre le « window. » devant ces dernières.
- Les méthodes alert(), prompt() et confirm(), qui permettent un dialogue interactif, ont déjà été utilisées.

Par exemple :

```
nb = prompt('Donner un nombre : ', 7)
if (isNaN(nb)) alert( nb+ " n'est pas un entier !");
```

- méthodes focus(), blur() qui active ou désactive la fenêtre
- moveBy(dx, dy) déplace la fenêtre par translation et moveTo(x,y) déplace le coin gauche, haut au point (x, y).
- resizeBy(dl, dh) et resizeTo(l, h) jouent des rôles analogues, quant à la taille de la fenêtre.
- les méthodes de temporisation setTimeout() et clearTimeout()
- print() pour imprimer la page courante :
`Imprimer cette page`

	Auteur	Nom région	Formation	Date Mise à jour	Page 3/ 9
	Didier Bonneau	GRN164	DWWM	14/10/2022	Support_De_Cours- Les_objets_du_navigateur.docx

- `open()` et `close()` qui ouvre et ferme des fenêtres enfants (voir ci-dessous).

D. GESTION DE FENETRES

Avec `open()` et `close()`, le programmeur dispose de moyens très souples avec les gestionnaires d'événements pour ouvrir ou fermer des fenêtres auxiliaires.

`fen = open("URL","nom_fenetre","options", replace);` ouvre une nouvelle fenêtre

- "URL" est l'adresse du document à charger dans la nouvelle fenêtre. Si on met une chaîne vide alors une nouvelle fenêtre « about :blank » est créée.
- "nom_fenetre" est le nom de la fenêtre, à donner à l'attribut TARGET des balises `<form>` ou `<a>` (ne précise pas le « title »)
- "options" est une liste d'éléments optionnels qui précisent l'aspect de la fenêtre.
- `replace` : booléen indiquant si cette nouvelle url devient le document courant dans l'historique (`true` remplace)

On utilise ensuite la variable `fen` pour faire référence à un objet ou une propriété de la nouvelle fenêtre.

Par exemple, pour écrire dans le composant message du formulaire `formu` situé dans un document de la fenêtre `fen`, on écrit :

`fen.document.formu.message.value= "...";`

La méthode `close()` adressée à une variable fenêtre permet de la fermer. Il est préférable avant de tester si elle a été ouverte (elle peut être masquée), avec une condition du genre `if (fen != null)`.

Exemple :

```
<script type="text/javascript">
  var fen = null;
  var options="width=500,height=200,toolbar=yes,"+
    "directories=no,menubar=no,scrollbars=yes,status=yes";
  function ouvrirFenetre() {
    fen = open("", "fenetre", options, false);
  }
</script>
<body>
  <form>
    <h3>Pour créer une fenêtre "enfant", cliquer sur ce bouton :</h3>
    <input type="button" value="Nouvelle fenêtre"
onclick="ouvrirFenetre()"/>
    <input type="button" value="Fermer" onclick="fen.close()"/>
  </form>
</body>
```

IV. L'OBJET « NAVIGATOR »

L'objet appelé « navigator », est créé au démarrage du logiciel.

Propriétés de l'objet Navigator

- Plate-forme utilisée : navigator.platform
- Son nom : navigator.appName
- Son nom de code : navigator.appCodeName
- Les infos d'entête envoyées au serveur requête http : navigator.userAgent
- Les plugins installés : navigator.plugins (tableau d'objets)
- Savoir si les cookies sont autorisés : navigator.cookieEnabled (true, false)

Ex : voici les valeurs des propriétés essentielles pour le navigateur actuel.



Il est toutefois nécessaire de tester le navigateur, pour pouvoir adapter le code aux 2 navigateurs ennemis Netscape et IE. Par exemple, voici un script qui renverrait à la page précédente s'il ne détecte pas la famille Netscape (Firefox, Opera, ...).

```
if (navigator.appName != 'Netscape')
    window.history.back();
else document.write('Vous avez fait le bon choix !')
```

Tout aussi important est la nécessité de savoir quels sont les plugins, module externe, installés pour pouvoir interpréter certains types de fichiers. Le navigateur stocke ces informations dans l'objet « navigator.plugins »

```
<script type="text/javascript">
if (navigator.plugins) {
    document.writeln("<table border=\"1\">");
    for(var i=0; i < navigator.plugins.length; i++) {
        document.writeln("<tr>");
        document.writeln("<td>" + navigator.plugins[i].name + "</td>");
        document.writeln("<td>" + navigator.plugins[i].description +
"</td>");
        document.writeln("<td>" + navigator.plugins[i].filename + "</td>");
        document.writeln("</tr>");
    }
    document.writeln("</table>");
}
```

```
else document.write('Pas de plugin !');
</script>
```

La liste des plugins est aussi accessible en entrant « `about:plugins` » dans la barre d'adresse de votre navigateur.

De façon tout-à-fait semblable, on peut connaître les types-mimes connus par le navigateur en parcourant les propriétés l'objet `navigator.mimeTypes`, c'est-à-dire les types de fichiers connus et les logiciels associés pour leur affichage.

V. L'OBJET « HISTORY »

Il contient les urls des documents chargés dans la fenêtre (et ainsi ses méthodes simulent les actions des boutons suivant et précédent)

- `history.back()` : charge le document précédent dans la fenêtre spécifiée
- `history.forward()` : charge le document suivant
- `history.go(n)` : recharge le document situé à `n` étapes du présent document (`n` est de signe qcq)

VI. L'OBJET « LOCATION »

On a accès à l'adresse complète de la page web affichée actuellement, et on peut la questionner, en obtenir des parties et modifier cette adresse. Lors d'une modification, le navigateur exécute un saut à la nouvelle adresse, exactement comme s'il s'agissait d'un lien activé.


Voici les valeurs des propriétés et méthodes essentielles pour le navigateur actuel.

- `location.href`, chaîne contenant l'url = `http://www.afpa.fr/Dom0/objets-navigateur.html`
- `location.pathname`, le chemin de l'url = `/web2/Dom0/objetsnavigateur.html`
- `location.host`, nom du domaine = `www.afpa.fr`
- `location.protocol` = `http:`
- `location.search` =
- `location.reload()`

VII. L'OBJET « DOCUMENT » (DOM 0)

La hiérarchie des objets a été vue précédemment. Elle reflète la construction de la page et l'insertion des différents éléments. Un document HTML peut contenir diverses balises d'insertion d'images, de formulaires, d'hyperliens, etc... qui sont autant d'objets du point de vue du navigateur.

- Quand il a complètement chargé la page HTML, le navigateur commence en interne à créer des tableaux, ou plutôt des collections indicées, dans lesquels sont rangées des références dans l'ordre de leur apparition dans le document. Le programmeur pourra avoir accès à ces éléments individuellement ce qui permettra des traitements par JS.

	Auteur	Nom région	Formation	Date Mise à jour	Page 6/ 9
	Didier Bonneau	GRN164	DWWM	14/10/2022	Support_De_Cours- Les_objets_du_navigateur.docx

- Ainsi, il crée les collections `document.images[]`, pour y placer toutes les images de la page, `document.forms[]`, pour y adresser tous les formulaires de la page, `document.links[]` pour tous les liens de la page. Chacun de ces objets peut posséder des propriétés et des méthodes.
- Par exemple, soit le 1er formulaire déclarée par `<form name="formu">` : on pourra accéder à cet objet en le nommant aussi bien par « `document.formu` » que par « `document.forms[0]` ».
- De même si la balise `` décrit la 3ème image insérée dans la page, alors on pourra accéder à l'objet Image en JS par `document.images[2]`
- Autre exemple : Soit un champ de texte nommé « nom », inclus dans un formulaire nommé « formu » par la déclaration `<input type="text" name="nom" id="nom">`, pour changer le texte on écrira simplement en JS : `document.formu.nom.value="Toto"`;
- Le programmeur JS peut avoir accès en lecture à ces collections, et en écriture à certaines des propriétés de ses sous-objets. Pour cela il devra les nommer à l'aide des noms de collections prédéfinis OU des noms donnés individuellement à ces éléments (à l'aide de l'attribut `name` ou `id`). Leur taille est obtenue avec la propriété « `length` » : ainsi `forms.length` est le nombre de formulaires inclus dans le document.
- Voici la liste des collections, et les balises correspondantes (les plus utilisés sont `forms[]` et `images[]`)
 - `forms[]` collection de tous les formulaires `<form>`
 - `elements[]` collection de tous les composants d'un formulaire (`<input ...>`, `<select>`, dans l'ordre d'apparition)
 - `options[]`, collection des items d'une liste déroulante quelconque.
 - `images[]` collection des images ``
 - `links[]` collection des liens hypertextes ``
 - `anchors[]` collection des liens internes ``
 - `applets[]` collection des applets `<applet code=...>`


VIII. EVENEMENTS USUELS (DOM 0)

Sur une interface graphique, l'utilisateur déclenche des "événements" (déplacement souris, clic sur un bouton, choix d'une option de liste déroulante etc ...) relativement à un objet (lien, composant de formulaire ..). L'événement est décelé (capté) par l'objet cible si celui-ci possède une "sensibilité" à l'événement. Il faut donc connaître la correspondance objet-événement.

S'il prévoit un intérêt à "répondre" à cet événement décelé, le programmeur doit à l'avance associer du code JS ou une fonction JS à un tel couple objet-événement, concrétisé par la présence d'un gestionnaire d'événement sur cet objet. L'appel et l'exécution de ce code ou de cette fonction seront automatiquement déclenchés par l'événement, et constituent ainsi la "réponse" à celui-ci.

A. SYNTAXE (EN PRINCIPE OBSOLETE ..)

`<balise name="..." onxxx="fonctJS()"></balise>`

	Auteur	Nom région	Formation	Date Mise à jour	Page 7/ 9
	Didier Bonneau	GRN164	DWWM	14/10/2022	Support_De_Cours- Les_objets_du_navigateur.docx

- onxxx est le nom du gestionnaire d'événements associé à l'événement, par exemple onclick
- balise est un nom de balise qui sait gérer un tel événement.
- "fonctJS()" est généralement une fonction déclarée auparavant, mais ce peut être aussi une suite d'instructions JS, séparées par des « ; ». Si la fonction doit dialoguer avec des composants de formulaire, il est pratique de passer « this.form » comme paramètre (ou le nom du formulaire comme dans l'exemple au-dessus). Avec cette référence au formulaire dans son ensemble, on pourra dans la fonction s'adresser à chaque composant et à chacune de ses propriétés.

B. TABLEAU RECAPITULATIF

Gest. Événement	provoqué par l'utilisateur qui ...
onblur	enlève le focus du composant
onchange	change la valeur d'un texte ou d'un composant à options
onclick	clique sur un composant ou un hyperlien
onfocus	donne le focus au composant
onload	après le chargement complet de la page dans le navigateur
onmouseout	fait quitter la souris d'un composant
onmouseover	fait passer la souris sur un composant
onreset	réinitialise un formulaire
onselect	sélectionne une zone d'édition d'un formulaire
onsubmit	soumet l'envoi du formulaire
onunload	après la fermeture de la page
ondblclick	fait un double-clic sur un composant
onmousedown	enfonce le bouton de la souris
onmouseup	relâche le bouton de la souris
onmousemove	déplace la souris dans un objet du document
onkeydown	enfonce une touche du clavier
onkeyup	relâche la touche
onkeypress	enfonce une touche du clavier