DOMAssistant

Cette version est la traduction française de la documentation originale.

DOMAssistant est constitué de 6 modules individuels dont seul le module *Core* est obligatoire. Chacun des autres modules peut être utilisé sans aucune relation de dépendance avec les autres. Vous trouverez ci-dessous chacune des méthodes documentées et accompagnées d'exemples de code.

Table of Contents

DOMAssistant Core Module - Module de base

Le module de base DOMAssistant est obligatoire : il fournit un socle commun à l'ensemble des fonctionnalités de DOMAssistant. Il est constitué des fonctions et méthodes principales permettant de le mettre en oeuvre.

$(cssSelector/elementReference)

La méthode *$* est utilisée pour donner une référence à un ou plusieurs éléments. Elle supporte un sélecteur CSS en tant que chaîne de caractères *(string)* ou une référence déjà existante à un élément. Elle renverra l'élément (ou les éléments) correspondant(s) après lui (leur) avoir appliqué toute méthode DOMAssistant additionnelle. Un appel à l'une de ces méthodes échouera sans provoquer d'erreur

si la méthode *$* a renvoyé un tableau *(array)* vide.

Paramètres
Un sélecteur CSS ou une référence objet.

Valeur de retour
Renvoie toujours un tableau *(array)* constitué des éléments correspondants si un sélecteur CSS est passé en argument. Dans le cas où un objet unique est passé en argument, la méthode *$* renvoie cette même référence et, dans le cas où plusieurs références objets lui auraient été passées, elle renverra un tableau constitué de celles-ci.

Exemples de code

```
$("#container input[type=text]");

$("#navigation a");

$("#news-list");

$("#container", "#navigation .important-item", "#content");

$(document.body);
```

$$(elementId)

La méthode *$$* est utilisée pour obtenir une référence rapide à un élément, de la même maniére que document.getElementById. Elle renverra une référence directe à l'élément DOM trouvé, aprés lui avoir appliqué toute méthode DOMAssistant additionnelle.

Contrairement à la méthode *$*, si la méthode *$$* n'a pas trouvé de résultat, elle provoquera une erreur si on tente de lui appliquer une méthode.

Paramètres
Id de l'élément recherché

Valeur de retour
Renvoie une référence DOM.

Exemples de code

```
$$("container");

$$("navigation");
```

elmsByClass(className, tag)

Permet d'obtenir des éléments en se basant sur leur nom de classe. Cette méthode impose un paramètre obligatoire qui est le nom de classe recherché et accepte un paramètre optionnel si vous désirez restreindre la recherche sur un type de balise particulier.

Paramètres

className
Nom de la classe CSS à rechercher. Obligatoire.

tag
Rechercher seulement les éléments correspondant à ce type de balise. Optionnel.

Valeur de retour

Tout appel renvoie un tableau constitué des références aux éléments trouvés.

Exemples de code

```
$("#container").elmsByClass("mandatory");

$$("container").elmsByClass("external-link", "a");
```

elmsByAttribute(attr, attrVal, tag)

Permet d'obtenir les éléments en se basant sur le fait qu'ils posséde un attribut particulier. Il est également possible de spécifier la valeur attendue de cet attribut et s'il faut limiter la recherche à un type de balise particulier. Seul le premier paramètre est obligatoire.

Paramètres

attr
Nom de l'attribut à rechercher. Obligatoire.

attrVal

Valeur que l'attribut recherché doit posséder. Optionnel. Utilisez le caractére Joker ("*") si vous ne cherchez pas de valeur particulière mais que vous souhaitez tout de même spécifier un type de balise.

tag
Limiter la recherche aux éléments correspondants ayant ce type de balise.

Valeur de retour

Tout appel renvoie un tableau constitué des références aux éléments trouvés.

Exemples de code

```
$("#container").elmsByAttribute("href");
```

```
$$("container").elmsByAttribute("name", "subscription");
```

```
$$("container").elmsByAttribute("type", "text", "input");
```

elmsByTag(tag)

Permet d'obtenir les éléments en se basant sur leur nom de balise (c'est à dire l'élément dont il s'agit). Cette méthode impose un paramètre obligatoire qui est le type de la balise recherchée.

Paramètres

tag
Nom du type de balise recherché. Obligatoire.

Valeur de retour

Tout appel renvoie un tableau constitué des références aux éléments trouvés.

Exemples de code

```
$$("container").elmsByTag("input");
```

each(functionRef)

Permet d'exécuter une fonction sur chacun des éléments d'une collection de références (renvoyées par les méthode *$* ou *$$*, par exemple).

Valeur de retour

Tout appel renvoie soit une référence à un élément unique, soit un tableau de références.

Paramètres

functionRef
Fonction qui devra être appelée pour chaque élément du tableau sur lequel la méthode est appelée. Cette fonction peut être anonyme ou une référence à une fonction.

Exemples de code

```
$("#navigation a").each(function () {
     // Do some JavaScript magic}
);
```

DOMAssistant AJAX - Module Ajax

Le module Ajax de DOMAssistant offre les interactions AJAX de base pour atteindre des données à partir d'un URL et traiter le contenu renvoyé avec une fonction quelconque ou charger ce contenu dans un élément de la page.

get(url, callBack)

Exécute une requête GET sur l'URL spécifié et appelle la fonction de callBack définie. Le premier paramètre de cette fonction de callBack sera le texte de réponse *(responseText)* de la requête AJAX. Si cette méthode est appelée sur un élément, le contexte de la fonction de callBack sera cet élément, c'est à dire que le mot-clé *this* fera référence à l'élément qui aura appelée la méthode *get()* en premier.

Paramètres

url
URL sur lequel faire la requête AJAX. Obligatoire.

callBack
Nom de fonction ou fonction anonyme à appeler lorsque la requête est complète. Optionnel.

Valeur de retour

Aucune.

Exemples de code

```
$("news").get("news.php", insertNews);
```

```
DOMAssistant.AJAX.get("my-url.aspx", callbackFunctionName);
```

post(url, callBack)

Exécute une requête POST sur l'URL spécifié et appelle la fonction de callBack définie. Le premier paramètre de la fonction de callBack sera de texte de la réponse *(responseText)* de l'appel AJAX. Si cette méthode est appelée sur un élément, le contexte de la fonction de callBack sera cet élément, c'est à dire que le mot-clé this fera référence à l'élément qui aura appelée la méthode *post()* en premier.

Paramètres

url
URL sur lequel faire la requête AJAX. Obligatoire.

callBack
Nom de fonction ou function anonyme à appeler lorsque la requête est complète. Optionnel.

Valeur de retour

Aucune.

Exemples de code

```
$("news").post("news.php?value=true", insertNews);
```

```
DOMAssistant.AJAX.post("my-url.aspx?number=10",
callbackFunctionName);
```

load(url, add)

Exécute une requête sur l'URL spécifié et charge le contenu renvoyé dans l'élément sur lequel la méthode est appelée.

Paramètres

url
URL sur lequel faire la requête AJAX. Obligatoire.

add
Booléen, si le contenu renvoyé doit être ajouté au contenu existant au lieu de le remplacer. Optionnel.

Valeur de retour

Aucune.

Exemples de code

```
$("directions").load("maps.php");
```

```
$("contacts”).load("staff.aspx", true);
```

getReadyState()

Méthode-outil pour tester l'état *(readyState)* courant de la requête *XMLHttpRequest*.

Paramètres

Aucun.

Valeur de retour

Nombre (integer)

0 = Non initialisé

1 = en chargement
2 = chargé
3 = interactif
4 = complet

Exemples de code

```
DOMAssistant.AJAX.getReadyState();
```

getStatus()

Méthode-outil pour tester le statut *(status)* courant de la requête *XMLHttpRequest*.

Paramètres

Aucun

Valeur de retour

Nombre (integer). Exemples : *200* pour Ok ou *404* pour non-trouvé.

Exemples de code

```
DOMAssistant.AJAX.getStatus();
```

getStatusText()

Méthode outil pour tester le *readyState* courant de la requête *XMLHttpRequest*.

Paramètres

Aucun.

Valeur de retour

Chaîne (string). Le texte qui accompagne le statut.

Exemples de code

```
DOMAssistant.AJAX.getStatusText();
```

DOMAssistant Content - Module "Contenu"

Le module "contenu" de DOMAssistant propose diverses méthodes pour ajouter et enlever du contenu et des éléments à une page.

prev()

Trouve la référence de l'élément HTML immédiatement précédent en sautant automatiquement tout noeud-texte qui pourrait se trouver entre deux.

Paramètres

Aucun.

Valeur de retour

Référence de l'élément précédant celui sur lequel est appelée la méthode.

Exemples de code

```
$("container").prev();
```

next()

Trouve la référence de l'élément HTML suivant en sautant automatiquement tout noeud-texte qui pourrait se trouver entre deux.

Paramètres

Aucun.

Valeur de retour

Référence de l'élément qui suit celui sur lequel est appelée la méthode.

Exemples de code

```
$("container").next();
```

create(name, attr, append, content)

Crée un élément et, optionnellement, lui applique des attributs, l'ajoute à l'élément sur lequel est appelée la méthode et lui ajoute un contenu.

Paramètres

name
Type de la balise du nouvel élément. Obligatoire.

attr
Un objet contenant des attributs et leurs valeurs. Optionnel.

append
Booléen. Suivant que le nouvel élément doit ou non être ajouté immédiatement. Optionnel.

content
Une chaîne *(string)* ou un objet HTML qui deviendra le contenu de l'élément à sa création. Otionnel.

Valeur de retour

L'élément créé par la méthode.

Exemples de code

```
$("container").create("div");
```

```
$("container").create("div", {id : "my-div",className : "my-class" });
```

```
$("container").create("div", null, false, "Hello!");
```

```
$("container").create("div", {id : "my-div",className : "my-class" }, true, "Hi there!");
```

setAttributes(attr)

Définit des attributs à l'élément sur lequel est appelée la méthode.

Paramètres

attr
Un objet contenant les attributs et leurs valeurs. Obligatoire.

Valeur de retour

L'élément qui a appelé la méthode.

Exemples de code

```
$("container").setAttributes({id : "my-div",className : "my-class"
});
```

addContent(content)

Ajoute du contenu à l'élément sur lequel est appliquée la méthode.

Paramètres

content
Peut être soit une chaîne *(string)*, laquelle sera appliquée en utilisant *innerHTML*, soit un objet HTML qui sera appliqué en utilisant *appendChild*.

Valeur de retour

L'élément qui a appelée la méthode.

Exemples de code

```
$("container").addContent("<p>A new paragraph</p>");

$("container").addContent(document.createElement("p"));
```

replaceContent(newContent)

Remplace le contenu de l'élément sur lequel est appliquée la méthode par autre chose.

Paramètres

newContent
Peut être soit une chaîne *(string)*, laquelle sera appliquée en utilisant *innerHTML*, soit un objet HTML qui sera appliqué en utilisant *appendChild*.

Valeur de retour

L'élément qui a appelé la méthode.

Exemples de code

```
$("container").replaceContent("<p>A new paragraph</p>");

$("container").replaceContent(document.createElement("p"));
```

remove()

Supprime l'élément sur lequel est appelée la méthode.

Paramètres

Aucun.

Valeur de retour

Null.

Exemples de code

```
$("container").remove();
```

DOMAssistant CSS - Module CSS

Le module CSS de DOMAssistant propose diverses méthodes pour ajouter ou supprimer des classes CSS, tester si une classe donnée est appliquée à un élément et récupérer la valeur de style pour une propriété spécifique d'un élément.

addClass(className)

Ajoute un nom de classe à l'élément sur lequel porte la méthode, sauf si ce nom

de classe existe déjà.

Paramètres

className
Nom de la classe qui doit être ajoutée. Obligatoire.

Valeur de retour

L'élément qui a appelé la méthode.

Exemples de code

```
$("container").addClass("selected");
```

removeClass(className)

Supprime une classe de l'élément sur lequel est appelée la méthode.

Paramètres

className
Nom de la classe qui doit être supprimée. Obligatoire.

Valeur de retour

L'élément qui a appelé la méthode.

Exemples de code

```
$("container").removeClass("selected");
```

hasClass(className)

Teste si l'élément sur lequel est appelée la méthode possède ce nom de classe.

Paramètres

className
Nom de la classe dont on souhaite tester la présence.

Valeur de retour

Booléen. Suivant que l'élément possède cette classe ou non.

Exemples de code

```
$("container").hasClass("selected");
```

getStyle(cssRule)

Récupère la valeur de style pour une propriété CSS de l'élément sur lequel est appelée la méthode sans tenir compte du fait qu'il ait été appliqué dynamiquement ou depuis une feuille de styles externe. Note : Assurez-vous de chercher une propriété précise plutôt qu'une propriété globale, c'est à dire *background-color* plutôt que *background*.

Paramètres

cssRule
Propriété CSS dont on souhaite récupérer la valeur. Utilisez la syntaxe CSS orthodoxe pour la propriété, c'est à dire *background-color* et non *background*. Obligatoire.

Valeur de retour

Chaîne de caractères *(string)*. Valeur du style recherché.

Exemples de code

```
$("container").getStyle("background-color");
```

DOMAssistant Events - Module "événements"

Le module "événements" de DOMAssistant propose différentes méthodes pour ajouter et supprimer des gestionnaires *(event handlers)* pour un ou plusieurs événements sur un élément. Il contient également des fonctionnalités pour empêcher le comportement par défaut et la propagation *(bubbling)* des événements.

addEvent(evt, func)

Ajoute un gestionnaire d'événements pour l'élement sur lequel est appelée la méthode. De multiples gestionnaires d'événements sont supportés et la fonction appelée recevra une référence à ce gestionnaire ainsi qu'une référence *this* à l'élément sur lequel l'événement a été déclenché, sans avoir à se soucier du type de navigateur. *Pour des raison d'accessibilité, assurez-vous d'appliquer les gestionnaires de clicks uniquement sur des éléments qui les autorisent même si Javascript est désactivé.*

Paramètres

evt
Type d'événement à détecter, spécifié en tant que chaîne de caractères *(string)* sans le préfixe *"on"*.

func
Fonction pour traiter l'événement, spécifiée comme référence à une fonction (sans les parenthèses) ou une fonction anonyme.

Valeur de retour

L'élément qui a appelé la méthode.

Exemples de code

```
$("container").addEvent("click", getListing);
```

```
$("container").addEvent("click", function (){alert("Hello darling!");});
```

removeEvent(evt, func)

Supprime un gestionnaire d'événements lié à l'élément sur lequel est appelée la méthode. Ne fonctionne qu'avec des références de fonctions et pas avec des fonctions anonymes.

Paramètres

evt
Gestionnaire d'événements à supprimer, spécifié sous forme d'une chaîne de

caractères *(string)* sans le préfixe *"on"*.

func
Fonction à appeler à l'arrêt, spécifiée sous forme d'une référence de focntion (sans les parenthèses).

Valeur de retour

L'élément qui a appelée la méthode.

Exemples de code

```
$("container").removeEvent("click", getListing);
```

preventDefault(evt)

Empêche le déclenchement de l'action par défaut pour l'élément sur lequel est appelée la méthode. Peut être appelé depuis n'importe quelle fonction, n'est une méthode d'aucun élément.

Paramètres

evt
Evénément dont on veut empêcher l'action par défaut.

Valeur de retour

Aucun.

Exemples de code

```
DOMAssistant.preventDefault(eventReference);
```

cancelBubble(evt)

Empêche la propagation de l'événement. Peut être appelé par n'importe quelle fonction, n'est une méthode pour aucun objet.

Paramètres

evt

Evénement dont on veut empêcher la propagation.

Valeur de retour

Aucun.

Exemples de code

```
DOMAssistant.cancelBubble(eventReference);
```

DOMAssistantLoad - Module chargement

Le module Chargement de DOMAssistant offre la possibilité d'effectuer une certain nombre d'appels de fonctions dès que le DOM est chargé, par opposition avec le fait d'attendre qu'images et autres fichiers externes soient intégralement chargés. Cette fonctionnalité a été inspirée et influencée par Dean Edwards, Mathias Miller et John Resig dans window.onload (again).

DOMReady()

Depuis n'importe quel fichier, appelez simplement la méthode DOMReady avec les fonctions désirées et elles seront exécutées aussitôt que le DOM aura fini d'être chargé.

Paramètres

Toute référence de fonction, toute fonction anonyme ou toute chaîne de caractères *(string)* constituée du nom de fonction et des parenthèses.

Valeur de retour

Aucun.

Exemples de code

```
DOMAssistant.DOMReady(myFunc);

DOMAssistant.DOMReady("myFunc('Some text')");

DOMAssistant.DOMReady(myFunc, "anotherFunction()");
```

```
DOMAssistant.DOMReady(myFunc, function(){// Perform some magic});
```

Sélecteurs CSS supportés (CSS 1, CSS 2, CSS 3)

Le support d'utilisation des sélecteurs CSS est implémenté dans le module de base de DOMAssistant et sa méthode *$* pour obtenir la référence à un ou à de multiples éléments. La plupart des sélecteurs CSS pour CSS 1, CSS 2 et CSS 3 sont supportés avec exactement la même syntaxe que celle que vous utiliseriez dans un fichier CSS.

Pour une explication approfondie des sélecteurs CSS ainsi que des exemples sur la manière de les employer, veuillez lire CSS 2.1 selectors et CSS 3 slectors explained.

Sélecteurs CSS implémentés

#container

Désigne un élément ayant l'id *"container"*.

.item

Désigne tous les éléments auxquels la classe *"item"* est attribuée.

#container.item

Désigne un élément ayant comme id *"container"* et auquel la classe *"item"* soit aussi appliquée.

p.info.error

Désigne tout élément P possédant à la fois la classe *"info"* et la classe *"error"*.

div p

Désigne tous les éléments P qui sont descendants d'un élément DIV.

div > p

Désigne tous les éléments P qui sont descendants directs d'un élément DIV.

div + p

Désigne tous les éléments P dont l'élément de même niveau immédiatement précédent est un élément DIV.

div ~ p

Désigne tous les éléments P qui sont parmi les éléments qui suivent un élement DIV de même niveau *(pas forcément immédiatement suivants)*.

div[id]

Désigne n'importe quel élément DIV possédant un attribut ID.

div[id=container]

Désigne tout élément DIV possédant un attribut ID de valeur *"container"*.

input[type=text][value=Yes]

Désigne tout élément INPUT possédant un attribut TYPE possédant la valeur *"text"* et un attribut VALUE égal à *"yes"*.

div[id^=empt]

Désigne tout élément DIV dont la valeur de l'attribut ID commence par *"empt"*.

div[id$=parent]

Désigne tout élément DIV dont la valeur de l'attribut ID s'achève par *"parents"*.

div[id*=mpt]

Désigne tout élément DIV dont la valeur de l'attribut contient la chaîne *"mpt"*.

div:first-child

Désigne tout élément DIV qui soit le premier élément enfant de son élément parent.

div:last-child

Désigne tout élément DIV qui soit le dernier élément enfant de son élément parent.

div:only-child

Désigne tout élément DIV qui soit le seul enfant de son élément parent.

div#container p:first-of-type

Désigne l'élément P qui est le premier élément P enfant dan'un élément DIV dont l'attribut ID à la valeur *"container"*.

p:last-of-type

Désigne l'élément P qui est le dernier élément P de son élément parent.

p:only-of-type

Désigne l'élément P qui est le seul élément P de son élément parent.

p:nth-of-type(7)

Désigne l'élément P qui est le 7eme élément P de son élément parent.

div#container p:nth-last-of-type(7)

Désigne l'élément P qui est le 7eme avant-dernier élément P de son élément parent. *(i.e. le 7eme en partant de la fin)*

div:empty

Désigne tout élément DIV qui soit totalement vide *(y-compris de noeud-texte)*

div:not([id=container])

Désigne tout élément DIV dont l'attribut ID n'est pas *"container"*.

div:nth-child(3)

Désigne tout élément DIV qui soit le troisième élément enfant de son élément parent.

div:nth-child(odd)

Désigne tout élément impair de son élément parent.

div:nth-child(even)

Désigne tout élément pair de son élément parent.

div:nth-child(n5+3)

Désigne chaque 5ème élément DIV enfant de son élément parent en commençant par le troisième. *C'est à dire les 8ème, 13ème, 18ème etc.*

input:enabled

Désigne tout élément INPUT qui soit activé.

input:disabled

Désigne tout élément INPUT qui soit désactivé.

input:checked

Désigne tout élément INPUT qui soit coché.

p, a

Désigne tous les éléments P et tous les éléments A.