DOMAssistant

Version 2.8

Cette version est la traduction française de la documentation originale. Responsable de la traduction – et des erreurs éventuelles : Samuel Desnoës.

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

Table des matières

DOMAssistant Core Module - Module de base 3

- $(cssSelector/elementReference) 3
- $$(elementId) 4
- elmsByAttribute(attr, attrVal, tag) 6
- hasChild(elementRef) 7
- prev() 7
- next() 8
- indexOf(elementRef) 9
- map(functionRef) 9
- filter(functionRef) 10
- every(functionRef) 10
- some(functionRef) 11
- first() 11
- end() 12
- store(key, value) 12
- retrieve(key) 13
- unstore(key) 13
- harmonize() 14

DOMAssistant AJAX - Module Ajax 14

get(url, callBack) 15
post(url, callBack) 15
load(url, add) 16
ajax(options) 17
getReadyState() 18
getStatus() 18
getStatusText() 19
DOMAssistant Content - Module "Contenu" 19
create(name, attr, append, content) 19
setAttributes(attr) 20
addContent(content) 20
replaceContent(newContent) 21
replace(content, returnNew) 21
remove() 22
DOMAssistant CSS - Module CSS 22
addClass(className) 23
removeClass(className) 23
hasClass(className) 23
setStyle(style, value) 24
getStyle(cssRule) 25
DOMAssistant Events - Module "événements" 25
addEvent(evt, func) 25
removeEvent(evt, func) 26
relayEvent(evt, selector, func) 27
unrelayEvent(evt) 28
triggerEvent(evt, target) 28

preventDefault(evt)....29
cancelBubble(evt)....29
DOMAssistantLoad - Module chargement....30
DOMReady()....30
setErrorHandling(func)....31
Sélecteurs CSS supportés (CSS 1, CSS 2, CSS 3)....31
Sélecteurs CSS implémentés....31

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 œuvre.

`$(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 renvoie 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 manière que document.getElementById. Elle renverra une référence directe à l'élément DOM trouvé, après 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");
```

cssSelect(cssSelector)

Utilisé sur une référence objet existante, permet d'utiliser un sélecteur CSS pour trouver un/des élément(s) enfant(s) de l'objet courant.

Paramètres

cssSelector
Chaîne de caractères (string) permettant de sélectionner le/les éléments désirés. *Obligatoire*.

Valeur de retour

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

Exemples de code

```
$(document).cssSelect(".mandatory");
```

```
$$(DOMElementReference).cssSelect(".important[type=test]");
```

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 possèdent 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 caractère 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 correspondant aux critères précédents et 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");
```

hasChild(elementRef)

Teste si *elementRef* est un descendant de l'élément courant.

Valeur de retour

TRUE si elementRef est un élément enfant (direct ou non), FALSE sinon.

Paramètres

elementRef
Une référence à l'élément à tester. Obligatoire.

Exemples de code

```
if (outerElm.hasChild(innerElm))

alert("innerElm is a child of outerElm!");
```

prev()

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

Paramètres

Aucun.

Valeur de retour

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

Exemple de code

$("container").prev();

next()

Trouve la référence de l'élément HTML suivant en sautant automatiquement tout nœud-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.

Exemple de code

$("container").next();

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.

Exemple de code

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

`indexOf(elementRef)`

Retourne l'index de la première occurrence d'*elementRef* dans une collection.

Valeur de retour

Index, basé sur zéro, de l'élément en question ; -1 s'il n'est pas trouvé.

Paramètres

elementRef
La référence de l'élément à rechercher.

Exemples de code

```
var elm = $$("content");
var idx = $("div").indexOf(elm);
```

`map(functionRef)`

Exécute une fonction sur chaque item et retourne le résultat dans un tableau.

Valeur de retour

Un tableau des résultats mappés.

Paramètres

functionRef
Fonction de mappage à appliquer à chaque item. Cette fonction reçoit en argument un index de référence à l'item courant.

Exemple de code

```
// Crée un tableau contenant des ID d'éléments DIV
var arrayID = $("div").map( function(idx) {
     return this.id;
});
```

`filter(functionRef)`

Exécute une fonction destinée à filtrer la collection d'éléments et retourne tous les items pour lesquels la fonction a renvoyé TRUE.

Valeur de retour

La collection des éléments filtrés.

Paramètres

functionRef
Fonction de filtrage à appliquer à chaque item. Cette fonction reçoit en argument un index de référence à l'item courant.

Exemple de code

```
var oddItems = $("div").filter( function(idx) {
     return (idx % 2 === 1);
});
```

`every(functionRef)`

Exécute une fonction sur chaque item d'une collection tant que cette fonction renvoie TRUE.

Valeur de retour

Valeur booléenne. TRUE si tous les items renvoient TRUE dans la fonction, FALSE sinon.

Paramètres

functionRef
Fonction à appliquer à chaque item. Cette fonction renvoie soit TRUE soit FALSE et reçoit en argument un index de référence à l'item courant.

Exemple de code

```
var tousOntDesEnfants = $("div").every( function(idx) {
     return (this.childNodes.length > 0);
});
```

some(functionRef)

Exécute une fonction sur chaque item dans une collection d'éléments tant que cette fonction renvoie FALSE.

Valeur de retour

Valeur booléenne. TRUE si certains items renvoient TRUE dans la fonction, FALSE sinon.

Paramètres

functionRef
Fonction à appliquer à chaque item. Cette fonction renvoie soit TRUE soit FALSE et reçoit en argument un index de référence à l'item courant.

Exemple de code

```
var certainsOntDesEnfants = $("div").some( function(idx) {
     return (this.childNodes.length > 0);
});
```

first()

Permet d'obtenir une référence au premier élément d'une collection.

Valeur de retour

Renvoie la référence au premier élément d'une collection, après lui avoir appliqué toute méthode DOMAssistant additionnelle.

Paramètres

Aucun

Exemple de code

```
$("#navigation a").first();
```

end()

Permet de remonter d'un niveau dans le contexte de chaîne courant.

Valeur de retour

Renvoie une référence à l'élément précédemment défini dans la chaîne.

Paramètres

Aucun

Exemple de code

```
// Renvoie la référence à l'élément A
$("#navigation a").create("span", null, true).end();
```

store(key, value)

Enregistre des données arbitraires dans l'élément courant.

Valeur de retour

L'élément qui a appelé la méthode.

Paramètres

key
Nom de la valeur qui doit être enregistrée. Ce nom peut être une chaîne de caractères, dans ce cas il est obligatoirement utilisé en conjonction avec le paramètre *value*. *Key* peut aussi être un objet constitué de plusieurs paires clé-valeur. Obligatoire.

value
La valeur qui doit être enregistrée, si le paramètre *key* est une chaîne de caractères. Optionnel suivant le type utilisé pour *key*.

Exemples de code

```
$$("elem").store("foo", "bar");
```

```
$$("child").store( { "name": "Tom", "age": 6, "sex": "M" } );
```

`retrieve(key)`

Retrouve les données préalablement enregistrées dans l'élément courant.

Valeur de retour

Valeur de la clé si elle est spécifiée, sinon, l'ID unique de l'élément.

Paramètres

key

Nom des données qui doivent être retrouvées. Optionnel. Si *key* n'est pas spécifié, l'ID unique de l'élément courant est renvoyé.

Exemples de code

```
var foo = $$("elem").retrieve("foo");
var age = $$("child").retrieve("age");
alert("L'ID unique est " + $$("child").retrieve());

// bar
// 6
```

`unstore(key)`

Supprime des données préalablement enregistrées dans l'élément courant.

Valeur de retour

L'élément qui a appelé la méthode.

Paramètres

key

Le nom des données à supprimer. Optionnel. Si Key n'est pas spécifié, toutes les données associées à l'élément courant sont supprimées.

Exemples de code

```
$$("elem").unstore("foo");
```

```
//Supprime toutes les données nommées "foo" dans $$("elem")


$$("child").unstore();
//Supprime toutes les données dans $$("child")
```

harmonize()

Permet à DOMAssistant de coexister en harmonie avec d'autres librairies javascript en empêchant la collision de $ et $$ dans l'espace de nommage global. Les méthodes dollar demeurent alors accessibles via DOMAssistant.$ et DOMAssistant.$$.

Valeur de retour

Aucune.

Paramètres

Aucune.

Exemple de code

```
<script src="OtherLib.js" type="text/javascript"> </script>
<script src="DOMAssistantCompressed.js" type="text/javascript">
</script>
```

```
<script type="text/javascript">
     DOMAssistant.DOMReady( function() {
          // Votre code DOMAssistant ici
     } );
     DOMAssistant.harmonize();
     // Le code d'autres librairies ici
</script>
```

DOMAssistant AJAX - Module Ajax

Le module Ajax de DOMAssistant offre les fonctionnalités AJAX de base permettant d'accéder à des données à partir d'un URL et d'en traiter le contenu renvoyé avec une fonction quelconque ou de 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 appelé 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 fonction 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);
```

ajax(options)

Méthode générique permettant d'exécuter des requêtes AJAX avancées dont les paramètres sont définis manuellement.

Paramètres

Options

Objet JavaScript dont différents paramètres sont définis. Les paramètres disponibles sont les suivants :

- url
- method ("GET" ou "POST")
- params
- callback (référence de fonction)
- headers
- responseType ("text" ou "xml")

L'objet est obligatoire mais, en l'absence de paramètres, seul l'url est invoqué. S'il n'est pas renseigné, il se résume à une requête GET avec un responseType `"text"`.

Valeur de retour

L'élément qui a appelé la fonction.

Exemple de code

```
$("#container").ajax({
     url: "ajax.php",
     method: "POST",
     params : "name=DOMAssistant",
     callback: functionReference,
     headers : {
          "Content-type" : "application/x-www-form-urlencoded"
     },
     responseType : "text",
     timeout : 5000,
     exception : function(e) {
          alert("Ajax error: " + (e.message || e.description));
     }
});
```

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.

`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. Optionnel.

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.

Exemple 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 appelé 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>Un nouveau paragraphe</p>");

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

replace(content, returnNew)

Remplace l'élément courant par un autre.

Paramètres

content
Nouveau contenu. Peut être spécifié soit par une référence à un élément, soit par une chaîne de caractères soit par un nombre.

returnNew
Si sa valeur est fixée à TRUE, le nouvel élément est renvoyé, sinon c'est l'élément remplacé qui est renvoyé. Optionnel.

Valeur de retour

Soit la référence à l'élément remplacé, soit la référence au nouvel élément.

Exemples de code

```
$("container").replace("<div><em>Way</em> cooler content!</div>");

var newElem = $$("placeholder").replace(elementRef, true);
```

`remove()`

Supprime l'élément sur lequel est appelée la méthode.

Paramètres

Aucun.

Valeur de retour

Null.

Exemple 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.

Exemple de code

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

removeClass(className)

Supprime la classe CSS appelée *className* 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.

Exemple 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.

Exemple de code

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

setStyle(style, value)

Définit un ou plusieurs styles pour un élément. *Note :* cette méthode utilise la syntaxe CSS et non celle de JavaScript pour les propriétés ; c'est à dire `background-color` à la place de `backgroundColor` etc.

Paramètres

style
Propriété CSS à définir. Il peut s'agir d'une chaîne de caractères (*string*), qui est alors utilisée conjointement avec le paramètre *`value`*, ou d'un objet avec plusieurs valeurs de styles. *Obligatoire*.

value
La valeur définie pour le style, si le paramètre `style` est une chaîne de caractères (*string*). *Optionnel*.

Valeur de retour

L'élément qui a appelé la méthode.

Exemples de code

```
$("#container").setStyle("border", "10px solid red");

$("#container").setStyle({
     background : "#ffffa2",
     color : "#f00",
     opacity : 0.5
});
```

getStyle(cssRule)

Récupère la valeur du 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.

Exemple 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'événement pour l'élément 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 sous la forme d'une 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 soit un gestionnaire d'événement spécifique, soit tous les gestionnaires d'événements liés à l'élément sur lequel est appelée la méthode.

Paramètres

evt
Gestionnaire d'événement à supprimer, spécifié sous forme d'une chaîne de caractères *(string)* sans le préfixe "*on*". Optionnellement, si aucun événement n'est passé en argument, tous les gestionnaires liés à l'élément (y-compris les gestionnaires d'événements *inline*) seront supprimés.

func
Fonction à appeler à l'arrêt, spécifiée sous forme d'une référence de fonction

(sans les parenthèses). Optionnellement, si aucune référence de fonction n'est spécifiée tous les gestionnaires liés à l'événement *evt* seront supprimés.

Valeur de retour

L'élément qui a appelé la méthode.

Exemple de code

```
$("container").removeEvent("click", getListing);
// Pour supprimer tous les événements liés aux lignes impaires,
vous pouvez écrire ceci :
$("tr:nth-child(odd)")
     .removeEvent("mouseover")
     .removeEvent("mouseout");
// ou ceci :
$("tr:nth-child(odd)").removeEvent();
```

relayEvent(evt, selector, func)

Ajoute un gestionnaire d'événements centralisé à l'élément courant. Tout événement déclenché sur l'élément correspondant au sélecteur est propagé aux éléments parents tout en restant géré dans l'élément courant. Les événements autres que ceux traditionnellement sujets à propagation sont aussi supportés. (*focus*, *blur*, *submit*, *reset*, *change* ou *select,* par exemple)

Paramètres

evt
Événement à gérer, spécifié sous la forme d'une chaîne de caractères sans le préfixe “on”. Des événements personnalisés sont acceptables.

selector
Cible actuelle sur laquelle l'événement pourrait être déclenché, spécifiée sous la forme d'un sélecteur CSS.

func
Fonction permettant de traiter l'événement, spécifiée sous la forme d'une référence de fonction (sans les parenthèses) ou sous la forme d'une fonction anonyme.

Valeur de retour

L'élément qui a appelé la méthode.

Exemples de code

```
$("ul").relayEvent("click", "li", jumpToPage);
$("form").relayEvent("focus", "input[type=text]", function() {
     this.addClass("yellow");
});
```

unrelayEvent(evt)

Supprime tous les événement “relayés” d'un type spécifique de l'élément courant.

Paramètres

evt
Événement à supprimer, spécifié sous la forme d'une chaîne de caractères sans le préfixe “on”.

Valeur de retour

L'élément qui a appelé la méthode.

Exemple de code

```
$("ul").unrelayEvent("click");
```

triggerEvent(evt, target)

Déclenche un événement sur l'élément courant et, optionnellement, définit la cible à laquelle l'événement est envoyé. Notez que l'événement n'est pas réellement déclenché (l'action correspondante ne survient pas) – cette méthode déclenche simplement l'événement au niveau du gestionnaire.

Paramètres

evt
Événement à déclencher, spécifié sous la forme d'une chaîne de caractères sans le préfixe “on”. Des événements personnalisés sont acceptables.

target
L'élément cible auquel l'événement est envoyé. Optionnel.

Valeur de retour

L'élément qui a appelé la méthode.

Exemples de code

```
$("news").triggerEvent("click");
$("home-link").triggerEvent("customevent", elementRef);
```

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
Événement dont on veut empêcher l'action par défaut.

Valeur de retour

Aucun.

Exemple 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
Événement dont on veut empêcher la propagation.

Valeur de retour

Aucun.

Exemple 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
```

```
});
```

setErrorHandling(func)

Donne la possibilité d'intercepter d'éventuelles erreurs survenues pendant l'appel à la méthode DOMReady et d'agir en conséquence.

Paramètres

Une référence à une fonction

Valeur de retour

Aucune.

Exemple de code

```
DOMAssistant.DOMLoad.setErrorHandling(function (e) {
     // e est l'objet erreur passé
});
```

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 selectors 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[foo~=bar]

Désigne tout élément DIV dont l'attribut foo est une liste de valeurs séparées par des espaces et dont l'une d'elles est exactement égale à “bar”.

div[lang|=en]

Désigne tout élément DIV dont l'attribut lang est une liste de valeurs séparées par des traits d'union qui commence (à partir de la gauche) par “en”.

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(5n+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.*

tr:nth-last-of-type(-n+3)

Désigne les trois dernières lignes de n'importe quel tableau.

td:nth-last-child(n+1)

Désigne toutes les colonnes sauf la dernière de n'mporte quel tableau.

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.

div:lang(zh)

Désigne tout élément DIV dont la langue est le Chinois.

p:target

Désigne l'élément P qui est la cible de l'URL de référence.

p, a

Désigne tous les éléments P et tous les éléments A.