eBook Gratuit

APPRENEZ jQuery

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#jquery

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: jquery

It is an unofficial and free jQuery ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official jQuery.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Démarrer avec jQuery

Remarques

jQuery est une bibliothèque JavaScript qui simplifie les opérations DOM, la gestion des événements, AJAX et les animations. Il prend également en charge de nombreux problèmes de compatibilité des navigateurs dans les moteurs DOM et javascript sous-jacents.

Chaque version de jQuery peut être téléchargée depuis https://code.jquery.com/jquery/ dans des formats compressés (minifiés) et non compressés.

Versions

Version	Remarques	Date de sortie
1.0	Première version stable	2006-08-26
1.1		2007-01-14
1.2		2007-09-10
1.3	Sizzle introduit dans le noyau	2009-01-14
1.4		2010-01-14
1,5	Gestion différée des rappels, réécriture du module ajax	2011-01-31
1.6	Des gains de performance significatifs dans les méthodes `attr()` et `val()`	2011-05-03
1,7	Nouvelles API d'événement: `on()` et `off()` .	2011-11-03
1.8	Sizzle réécrit, animations améliorées et flexibilité `$(html, props)` .	2012-08-09
1,9	Suppression des interfaces obsolètes et nettoyage du code	2013-01-15
1.10	Corrections de bugs incorporées et différences signalées pour les cycles bêta 1.9 et 2.0	2013-05-24
1.11		2014-01-24
1.12		2016-01-08
2.0	Suppression du support IE 6–8 pour améliorer les performances et réduire la taille	2013-04-18
2.1		2014-01-24

Version	Remarques	Date de sortie
2.2		2016-01-08
3.0	Des accélérations massives pour certains sélecteurs personnalisés jQuery	2016-06-09
3.1	Plus d'erreurs silencieuses	2016-07-07

Examples

Espace de noms jQuery ("jQuery" et "$")

`jQuery` est le point de départ pour écrire un code jQuery. Il peut être utilisé comme une fonction `jQuery(...)` ou une variable `jQuery.foo` .

`$` est un alias pour `jQuery` et les deux peuvent généralement être interchangés (sauf si `jQuery.noConflict();` a été utilisé - voir Eviter les collisions entre espaces de noms).

En supposant que nous avons cet extrait de HTML -

```
<div id="demo_div" class="demo"></div>
```

Nous pourrions vouloir utiliser jQuery pour ajouter du contenu textuel à cette div. Pour ce faire, nous pourrions utiliser la fonction `text()` jQuery. Cela pourrait être écrit en utilisant soit `jQuery` ou `$` . c'est à dire -

```
jQuery("#demo_div").text("Demo Text!");
```

Ou -

```
$("#demo_div").text("Demo Text!");
```

Les deux résulteront dans le même HTML final -

```
<div id="demo_div" class="demo">Demo Text!</div>
```

Comme `$` est plus concis que `jQuery` c'est généralement la méthode préférée pour écrire du code jQuery.

jQuery utilise des sélecteurs CSS et dans l'exemple ci-dessus, un sélecteur d'ID a été utilisé. Pour plus d'informations sur les sélecteurs dans jQuery, voir les types de sélecteurs .

Commencer

Créez un fichier `hello.html` avec le contenu suivant:

```
<!DOCTYPE html>
<html>
<head>
    <title>Hello, World!</title>
</head>
<body>
    <div>
        <p id="hello">Some random text</p>
    </div>
    <script src="https://code.jquery.com/jquery-2.2.4.min.js"></script>
    <script>
        $(document).ready(function() {
            $('#hello').text('Hello, World!');
        });
    </script>
</body>
</html>
```

Démo en direct sur JSBin

Ouvrez ce fichier dans un navigateur Web. En conséquence, vous verrez une page avec le texte: `Hello, World!`

Explication du code

1. Charge la bibliothèque jQuery à partir du CDN jQuery:

   ```
   <script src="https://code.jquery.com/jquery-2.2.4.min.js"></script>
   ```

   Cela introduit la variable globale `$` , un alias pour la fonction `jQuery` et l'espace de noms.

   > *Sachez que l'une des erreurs les plus courantes commises lors de l'inclusion de jQuery est de ne pas charger la bibliothèque AVANT tout autre script ou toute autre bibliothèque pouvant en dépendre ou l'utiliser.*

2. Diffère une fonction à exécuter lorsque le DOM (Document Object Model) est détecté comme étant "prêt" par jQuery:

   ```
   // When the `document` is `ready`, execute this function `...`
   $(document).ready(function() { ... });

   // A commonly used shorthand version (behaves the same as the above)
   $(function() { ... });
   ```

3. Une fois que le DOM est prêt, jQuery exécute la fonction de rappel indiquée ci-dessus. A l'intérieur de notre fonction, il n'y a qu'un seul appel qui fait 2 choses principales:

   1. Obtient l'élément avec l'attribut `id` égal à `hello` (notre sélecteur `#hello`). L'utilisation d'un sélecteur en tant qu'argument passé constitue le cœur des fonctionnalités et des noms de jQuery; la bibliothèque entière a essentiellement évolué à partir de l'extension document.querySelectorAll MDN .

2. Définissez le `text()` dans l'élément sélectionné sur `Hello, World!` .

```
#    ↓ - Pass a `selector` to `$` jQuery, returns our element
$('#hello').text('Hello, World!');
#             ↑ - Set the Text on the element
```

Pour plus d'informations, reportez-vous à la page jQuery - Documentation .

Inclure une balise de script dans la tête de la page HTML

Pour charger **jQuery** depuis le CDN officiel, rendez-vous sur le site Web de jQuery. Vous verrez une liste des différentes versions et formats disponibles.

Maintenant, copiez la source de la version de jQuery à charger. Supposons que vous vouliez charger **jQuery 2.X** , cliquez sur une balise **non compressée** ou **minifiée** qui vous montrera quelque chose comme ceci:

Copiez le code complet (ou cliquez sur l'icône de copie) et collez-le dans `<head>` ou `<body>` de votre code HTML.

La meilleure pratique consiste à charger toutes les bibliothèques JavaScript externes dans la balise head avec l'attribut `async` . Voici une démonstration:

```
<!DOCTYPE html>
    <html>
      <head>
         <title>Loading jquery-2.2.4</title>
         <script src="https://code.jquery.com/jquery-2.2.4.min.js" async></script>
      </head>
      <body>
          <p>This page is loaded with jquery.</p>
      </body>
   </html>
```

Lorsque vous utilisez un attribut `async` soyez conscient lorsque les bibliothèques javascript sont chargées et exécutées de manière asynchrone dès qu'elles sont disponibles. Si deux bibliothèques sont incluses et que la deuxième bibliothèque dépend de la première bibliothèque, est-ce que si la deuxième bibliothèque est chargée et exécutée avant la première bibliothèque, une erreur peut se produire et l'application peut se rompre.

Éviter les collisions d'espaces de noms

Les bibliothèques autres que jQuery peuvent également utiliser `$` comme un alias. Cela peut provoquer des interférences entre ces bibliothèques et jQuery.

Pour libérer `$` pour une utilisation avec d'autres bibliothèques:

```
jQuery.noConflict();
```

Après avoir appelé cette fonction, `$` n'est plus un alias pour `jQuery` . Cependant, vous pouvez toujours utiliser la variable `jQuery` elle-même pour accéder aux fonctions jQuery:

```
jQuery('#hello').text('Hello, World!');
```

Vous pouvez éventuellement affecter une variable différente en tant qu'alias pour jQuery:

```
var jqy = jQuery.noConflict();
jqy('#hello').text('Hello, World!');
```

À l'inverse, pour empêcher d'autres bibliothèques d'interférer avec jQuery, vous pouvez envelopper votre code jQuery dans une expression de fonction immédiatement invoquée (IIFE) et passer en `jQuery` comme argument:

```
(function($) {
    $(document).ready(function() {
        $('#hello').text('Hello, World!');
    });
})(jQuery);
```

Dans cet IIFE, `$` est un alias pour jQuery uniquement.

Un autre moyen simple de **sécuriser l'alias `$` de jQuery et de s'assurer que DOM est prêt** :

```
jQuery(function( $ ) { // DOM is ready
   // You're now free to use $ alias
   $('#hello').text('Hello, World!');
});
```

Résumer,

- `jQuery.noConflict()` : `$` ne fait plus référence à jQuery, `jQuery.noConflict()` à la variable `jQuery` .
- `var jQuery2 = jQuery.noConflict()` - `$` ne fait plus référence à jQuery, alors que la variable `jQuery` fait, de même que la variable `jQuery2` .

Maintenant, il existe un troisième scénario - Que faire si nous voulons que jQuery ne soit disponible **que dans `jQuery2` ** ? Utilisation,

```
var jQuery2 = jQuery.noConflict(true)
```

Cela se traduit par ni `$` ni `jQuery` faisant référence à jQuery.

Ceci est utile lorsque plusieurs versions de jQuery doivent être chargées sur la même page.

```
<script src='https://code.jquery.com/jquery-1.12.4.min.js'></script>
```

```
<script>
    var jQuery1 = jQuery.noConflict(true);
</script>
<script src='https://code.jquery.com/jquery-3.1.0.min.js'></script>
<script>
    // Here, jQuery1 refers to jQuery 1.12.4 while, $ and jQuery refers to jQuery 3.1.0.
</script>
```

https://learn.jquery.com/using-jquery-core/avoid-conflicts-other-libraries/

Chargement de jQuery via la console sur une page qui ne l'a pas.

Il faut parfois travailler avec des pages qui n'utilisent pas `jQuery` alors que la plupart des développeurs ont l'habitude d'avoir `jQuery` portée de main.

Dans de telles situations, vous pouvez utiliser la console des `Chrome Developer Tools` (`F12`) pour ajouter manuellement `jQuery` sur une page chargée en exécutant les opérations suivantes:

```
var j = document.createElement('script');
j.onload = function(){ jQuery.noConflict(); };
j.src = "https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js";
document.getElementsByTagName('head')[0].appendChild(j);
```

La version que vous souhaitez peut différer de celle ci-dessus (`1.12.4`), vous pouvez obtenir le lien pour celui dont vous avez besoin ici .

L'objet jQuery

Chaque fois que jQuery est appelé, en utilisant `$()` ou `jQuery()` , en interne, il crée une `new` instance de `jQuery` . Ceci est le code source qui montre la nouvelle instance:

```
// Define a local copy of jQuery
jQuery = function( selector, context ) {

    // The jQuery object is actually just the init constructor 'enhanced'
    // Need init if jQuery is called (just allow error to be thrown if not included)
    return new jQuery.fn.init( selector, context );
}
```

En interne, jQuery se réfère à son prototype sous la forme `.fn` , et le style utilisé ici pour instancier en interne un objet jQuery permet à ce prototype d'être exposé sans l'utilisation explicite de `new` par l'appelant.

En plus de la configuration d'une instance (qui est la manière dont l'API jQuery, telle que `.each` , `children` , `filter` , etc., est exposée), jQuery créera en interne une structure de type tableau pour correspondre au résultat du sélecteur (à condition que quelque chose d'autre que rien, `undefined` , `null` ou similaire a été passé en tant qu'argument). Dans le cas d'un seul élément, cette structure de type tableau ne contiendra que cet élément.

Une démonstration simple consisterait à trouver un élément avec un identifiant, puis à accéder à l'objet jQuery pour renvoyer l'élément DOM sous-jacent (cela fonctionnera également lorsque

plusieurs éléments sont mis en correspondance ou présents).

```
var $div = $("#myDiv");//populate the jQuery object with the result of the id selector
var div = $div[0];//access array-like structure of jQuery object to get the DOM Element
```

Chargement des plugins jQuery

En général, lorsque vous chargez des plugins, veillez à toujours inclure le plugin *après* jQuery.

```
<script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>
<script src="some-plugin.min.js"></script>
```

Si vous *devez* utiliser plusieurs versions de jQuery, assurez-vous de charger le ou les plug-ins *après* la version requise de jQuery suivie du code pour définir `jQuery.noConflict(true)` ; puis chargez la prochaine version de jQuery et ses plug-ins associés:

```
<script src="https://code.jquery.com/jquery-1.7.0.min.js"></script>
<script src="plugin-needs-1.7.min.js"></script>
<script>
// save reference to jQuery v1.7.0
var $oldjq = jQuery.noConflict(true);
</script>
<script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>
<script src="newer-plugin.min.js"></script>
```

Maintenant, lors de l'initialisation des plugins, vous devrez utiliser la version associée de jQuery

```
<script>
// newer jQuery document ready
jQuery(function($){
  // "$" refers to the newer version of jQuery
  // inside of this function

  // initialize newer plugin
  $('#new').newerPlugin();
});

// older jQuery document ready
$oldjq(function($){
  // "$" refers to the older version of jQuery
  // inside of this function

  // initialize plugin needing older jQuery
  $('#old').olderPlugin();
});
</script>
```

Il est possible d'utiliser une seule fonction de document pour initialiser les deux plugins, mais pour éviter toute confusion et tout problème avec du code jQuery supplémentaire dans la fonction de document prêt, il serait préférable de garder les références séparées.

Lire Démarrer avec jQuery en ligne: https://riptutorial.com/fr/jquery/topic/211/demarrer-avec-jquery

Chapitre 2: Ajax

Syntaxe

- var jqXHR = $ .ajax (URL [, paramètres])
- var jqXHR = $ .ajax ([paramètres])
- jqXHR.done (function (data, textStatus, jqXHR) {});
- jqXHR.fail (fonction (jqXHR, textStatus, errorThrown) {});
- jqXHR.always (function (jqXHR) {});

Paramètres

Paramètre	Détails
URL	Spécifie l'URL à laquelle la demande sera envoyée
paramètres	un objet contenant de nombreuses valeurs qui affectent le comportement de la requête
type	La méthode HTTP à utiliser pour la requête
Les données	Données à envoyer par la demande
Succès	Une fonction de rappel à appeler si la requête réussit
Erreur	Un rappel pour gérer les erreurs
statusCode	Un objet de codes HTTP numériques et de fonctions à appeler lorsque la réponse a le code correspondant
Type de données	Le type de données que vous attendez du serveur
contentType	Type de contenu des données à envoyer au serveur. La valeur par défaut est "application / x-www-form-urlencoded; charset = UTF-8"
le contexte	Spécifie le contexte à utiliser dans les rappels, généralement `this` qui fait référence à la cible actuelle.

Remarques

AJAX signifie **A** synchrone **J** avascript **un** e **X** ML. AJAX permet à une page Web d'effectuer une requête HTTP asynchrone (AJAX) sur le serveur et de recevoir une réponse, sans avoir à recharger la page entière.

Examples

Gestion des codes de réponse HTTP avec $ .ajax ()

Outre les `.done` , `.fail` et `.always` , qui sont déclenchés selon que la requête a réussi ou non, il est possible de déclencher une fonction lorsqu'un code d'état HTTP spécifique est renvoyé par le serveur. Cela peut être fait en utilisant le paramètre `statusCode` .

```
$.ajax({
    type: {POST or GET or PUT etc.},
    url:  {server.url},
    data: {someData: true},
    statusCode: {
        404: function(responseObject, textStatus, jqXHR) {
            // No content found (404)
            // This code will be executed if the server returns a 404 response
        },
        503: function(responseObject, textStatus, errorThrown) {
            // Service Unavailable (503)
            // This code will be executed if the server returns a 503 response
        }
    }
})
.done(function(data){
    alert(data);
})
.fail(function(jqXHR, textStatus){
    alert('Something went wrong: ' + textStatus);
})
.always(function(jqXHR, textStatus) {
   alert('Ajax request was finished')
});
```

Comme indiqué dans la documentation officielle de jQuery:

> Si la requête réussit, les fonctions de code d'état prennent les mêmes paramètres que le rappel de succès; si cela entraîne une erreur (y compris la redirection 3xx), ils prennent les mêmes paramètres que le rappel d' `error` .

Utiliser Ajax pour soumettre un formulaire

Parfois, vous pouvez avoir un formulaire et le soumettre en utilisant ajax.

Supposons que vous ayez cette forme simple -

```
<form id="ajax_form" action="form_action.php">
  <label for="name">Name :</label>
  <input name="name" id="name" type="text" />
  <label for="name">Email :</label>
  <input name="email" id="email" type="text" />
  <input type="submit" value="Submit" />
</form>
```

Le code jQuery suivant peut être utilisé (dans un appel à `$(document).ready`) -

```
$('#ajax_form').submit(function(event){
  event.preventDefault();
  var $form = $(this);

  $.ajax({
      type: 'POST',
      url: $form.attr('action'),
      data: $form.serialize(),
      success: function(data) {
        // Do something with the response
      },
      error: function(error) {
        // Do something with the error
      }
  });
});
```

Explication

- `var $form = $(this)` - le formulaire, mis en cache pour être réutilisé
- `$('#ajax_form').submit(function(event){` - Lorsque le formulaire avec l'ID "ajax_form" est soumis, exécutez cette fonction et passez l'événement en tant que paramètre.
- `event.preventDefault();` - Empêcher le formulaire de se soumettre normalement (nous pouvons aussi utiliser `return false` après la commande `ajax({});` qui aura le même effet)
- `url: $form.attr('action'),` - Récupère la valeur de l'attribut "action" du formulaire et l'utilise pour la propriété "url".
- `data: $form.serialize(),` - Convertit les entrées du formulaire en une chaîne adaptée à l'envoi au serveur. Dans ce cas, il renverra quelque chose comme "name=Bob&email=bob@bobsemailaddress.com"

Envoi de données JSON

jQuery facilite la gestion des *réponses* JSON, mais un peu plus de travail est nécessaire lorsqu'une requête donnée souhaite que vous *envoyiez des* données au format JSON:

```
 $.ajax("/json-consuming-route", {
      data: JSON.stringify({author: {name: "Bullwinkle J. Moose",
                                     email: "bullwinkle@example.com"} }),
      method: "POST",
      contentType: "application/json"
   });
```

Notez que nous spécifions le `contentType` correct pour les données que nous envoyons. C'est une bonne pratique en général et peut être requise par l'API sur laquelle vous publiez - mais cela a *aussi* pour effet secondaire de demander à jQuery de ne pas effectuer la conversion par défaut de `%20` à `+` , ce qui serait le cas si `contentType` était laissé à la valeur par défaut de l' `application/x-www-form-urlencoded` . Si pour une raison quelconque vous devez laisser contentType défini sur la valeur par défaut, veillez à définir `processData` sur false pour éviter cela.

L'appel à `JSON.stringify` pourrait être évité ici, mais son utilisation nous permet de fournir les

données sous la forme d'un objet JavaScript (évitant ainsi les erreurs de syntaxe JSON embarrassantes telles que l'impossibilité de citer les noms de propriété).

Tous dans un exemple

Ajax Obtenir:

Solution 1:

```
$.get('url.html', function(data){
    $('#update-box').html(data);
});
```

Solution 2:

```
 $.ajax({
     type: 'GET',
     url: 'url.php',
 }).done(function(data){
     $('#update-box').html(data);
 }).fail(function(jqXHR, textStatus){
     alert('Error occured: ' + textStatus);
 });
```

Ajax Load: Une autre méthode ajax get créée pour simplifier

```
$('#update-box').load('url.html');
```

.load peut également être appelé avec des données supplémentaires. La partie de données peut être fournie sous forme de chaîne ou d'objet.

```
$('#update-box').load('url.php', {data: "something"});
$('#update-box').load('url.php', "data=something");
```

Si .load est appelé avec une méthode de rappel, la demande au serveur sera une publication

```
$('#update-box').load('url.php', {data: "something"}, function(resolve){
    //do something
});
```

Ajax Post:

Solution 1:

```
$.post('url.php',
    {date1Name: data1Value, date2Name: data2Value},  //data to be posted
    function(data){
        $('#update-box').html(data);
    }
);
```

Solution 2:

```
$.ajax({
    type: 'Post',
    url: 'url.php',
    data: {date1Name: data1Value, date2Name: data2Value}  //data to be posted
}).done(function(data){
    $('#update-box').html(data);
}).fail(function(jqXHR, textStatus){
    alert('Error occured: ' + textStatus);
});
```

Ajax Post JSON:

```
var postData = {
    Name: name,
    Address: address,
    Phone: phone
};

$.ajax({
     type: "POST",
     url: "url.php",
     dataType: "json",
     data: JSON.stringfy(postData),
     success: function (data) {
         //here variable data is in JSON format
     }
 });
```

Ajax Get JSON:

Solution 1:

```
$.getJSON('url.php', function(data){
    //here variable data is in JSON format
});
```

Solution 2:

```
 $.ajax({
      type: "Get",
      url: "url.php",
      dataType: "json",
      data: JSON.stringfy(postData),
      success: function (data) {
          //here variable data is in JSON format
      },
      error: function(jqXHR, textStatus){
          alert('Error occured: ' + textStatus);
      }
  });
```

Ajax Annuler un appel ou une demande

```
var xhr = $.ajax({
    type: "POST",
    url: "some.php",
    data: "name=John&location=Boston",
    success: function(msg){
       alert( "Data Saved: " + msg );
    }
});
```

// tue la requête

```
xhr.abort()
```

Ajax File Uploads

1. Un exemple simple et complet

Nous pourrions utiliser cet exemple de code pour télécharger les fichiers sélectionnés par l'utilisateur chaque fois qu'une nouvelle sélection de fichier est effectuée.

```
<input type="file" id="file-input" multiple>
```

```
var files;
var fdata = new FormData();
$("#file-input").on("change", function (e) {
    files = this.files;

    $.each(files, function (i, file) {
        fdata.append("file" + i, file);
    });

    fdata.append("FullName", "John Doe");
    fdata.append("Gender", "Male");
    fdata.append("Age", "24");

    $.ajax({
        url: "/Test/Url",
        type: "post",
        data: fdata, //add the FormData object to the data parameter
        processData: false, //tell jquery not to process data
        contentType: false, //tell jquery not to set content-type
        success: function (response, status, jqxhr) {
            //handle success
        },
        error: function (jqxhr, status, errorMessage) {
            //handle error
        }
    });
});
```

Maintenant, décomposons cela et inspectons-le une par une.

2. Travailler avec des entrées de fichier

Ce document MDN (Utilisation de fichiers provenant d'applications Web) est une bonne lecture sur les différentes méthodes de gestion des entrées de fichiers. Certaines de ces méthodes seront également utilisées dans cet exemple.

Avant de télécharger des fichiers, nous devons d'abord donner à l'utilisateur un moyen de sélectionner les fichiers qu'ils souhaitent télécharger. Pour cela, nous utiliserons une `file input` . La propriété `multiple` permet de sélectionner plusieurs fichiers, vous pouvez la supprimer si vous souhaitez que l'utilisateur sélectionne un fichier à la fois.

```
<input type="file" id="file-input" multiple>
```

Nous utiliserons l' `change event` l'entrée pour capturer les fichiers.

```
var files;
$("#file-input").on("change", function(e){
    files = this.files;
});
```

À l'intérieur de la fonction de gestionnaire, nous accédons aux fichiers via la propriété files de notre entrée. Cela nous donne une FileList , qui est un objet de type tableau.

3. Créer et remplir les FormData

Afin de télécharger des fichiers avec Ajax, nous allons utiliser FormData .

```
var fdata = new FormData();
```

FileList que nous avons obtenu à l'étape précédente est un objet de type tableau et peut être itéré à l'aide de diverses méthodes, notamment pour loop , for ... of loop et jQuery.each . Nous allons rester avec le jQuery dans cet exemple.

```
$.each(files, function(i, file) {
  //...
});
```

Nous allons utiliser la méthode append de FormData pour ajouter les fichiers dans notre objet formdata.

```
$.each(files, function(i, file) {
  fdata.append("file" + i, file);
});
```

Nous pouvons également ajouter d'autres données que nous voulons envoyer de la même manière. Disons que nous voulons envoyer des informations personnelles que nous avons reçues

de l'utilisateur avec les fichiers. Nous pourrions ajouter cette information dans notre objet formdata.

```
fdata.append("FullName", "John Doe");
fdata.append("Gender", "Male");
fdata.append("Age", "24");
//...
```

4. Envoi des fichiers avec Ajax

```
$.ajax({
    url: "/Test/Url",
    type: "post",
    data: fdata, //add the FormData object to the data parameter
    processData: false, //tell jquery not to process data
    contentType: false, //tell jquery not to set content-type
    success: function (response, status, jqxhr) {
        //handle success
    },
    error: function (jqxhr, status, errorMessage) {
        //handle error
    }
});
```

Nous définissons les propriétés `processData` et `contentType` sur `false` . Ceci est fait pour que les fichiers puissent être envoyés au serveur et traités par le serveur correctement.

Lire Ajax en ligne: https://riptutorial.com/fr/jquery/topic/316/ajax

Chapitre 3: Ajouter

Syntaxe

1. $ (sélecteur) .append (contenu)
2. $ (contenu) .appendTo (sélecteur)

Paramètres

Paramètres	Détails
contenu	Types possibles: Elément, chaîne HTML, texte, tableau, objet ou même une fonction renvoyant une chaîne.

Remarques

- .append () & .after () peut potentiellement exécuter du code. Cela peut se produire par l'injection de balises de script ou l'utilisation d'attributs HTML qui exécutent du code (par exemple). N'utilisez pas ces méthodes pour insérer des chaînes obtenues à partir de sources non fiables, telles que des paramètres de requête d'URL, des cookies ou des entrées de formulaire. Cela peut introduire des vulnérabilités XSS (cross-site-scripting). Supprimez ou échappez à toute entrée utilisateur avant d'ajouter du contenu au document.
- jQuery ne supporte pas officiellement SVG. Utiliser les méthodes jQuery sur SVG les documents, sauf s'ils sont explicitement documentés pour cette méthode, peuvent provoquer des comportements inattendus. Exemples de méthodes prenant en charge SVG à partir de
  jQuery 3.0 est addClass et removeClass.

Examples

Ajout d'un élément à un conteneur

Solution 1:

```
$('#parent').append($('#child'));
```

Solution 2:

```
$('#child').appendTo($('#parent'));
```

Les deux solutions `#child` l'élément `#child` (en ajoutant à la fin) à l'élément `#parent` .

Avant:

```
<div id="parent">
  <span>other content</span>
</div>
<div id="child">

</div>
```

Après:

```
<div id="parent">
  <span>other content</span>
  <div id="child">

  </div>
</div>
```

Remarque: Lorsque vous ajoutez du contenu qui existe déjà dans le document, ce contenu sera supprimé de son conteneur parent d'origine et ajouté au nouveau conteneur parent. Donc, vous ne pouvez pas utiliser `.append()` ou `.appendTo()` pour cloner un élément. Si vous avez besoin d'un clone, utilisez `.clone()` -> [http://api.jquery.com/clone/][1]

Utilisation efficace consécutive de .append ()

Commençant:

HTML

```
<table id='my-table' width='960' height='500'></table>
```

JS

```
var data = [
    { type: "Name", content: "John Doe" },
    { type: "Birthdate", content: "01/01/1970" },
    { type: "Salary", content: "$40,000,000" },
    // ...300 more rows...
    { type: "Favorite Flavour", content: "Sour" }
];
```

Ajout dans une boucle

Vous venez de recevoir un grand nombre de données. Maintenant, il est temps de faire un tour et de le rendre sur la page.

Votre première pensée pourrait être de faire quelque chose comme ceci:

```
var i;                          // <- the current item number
var count = data.length;        // <- the total
var row;                        // <- for holding a reference to our row object

// Loop over the array
for ( i = 0; i < count; ++i ) {
    row = data[ i ];

    // Put the whole row into your table
    $('#my-table').append(
        $('<tr></tr>').append(
            $('<td></td>').html(row.type),
            $('<td></td>').html(row.content)
        )
    );
}
```

Ceci est *parfaitement valide* et rendra exactement ce que vous attendez, mais ...

Ne faites pas cela.

Rappelez-vous ces **300** lignes de données?

Chacun forcera le navigateur à recalculer les valeurs de largeur, de hauteur et de positionnement de chaque élément, ainsi que tous les autres styles - à moins qu'ils ne soient séparés par une limite de mise en page , malheureusement pour cet exemple (ils sont les descendants d'un élément `<table>`), ils ne peuvent pas.

À petites quantités et peu de colonnes, cette pénalité de performance sera certainement négligeable. Mais nous voulons compter chaque milliseconde.

De meilleures options

1. Ajouter à un tableau séparé, ajouter après la fin de la boucle

```
/**
 * Repeated DOM traversal (following the tree of elements down until you reach
 * what you're looking for - like our <table>) should also be avoided wherever possible.
 */

// Keep the table cached in a variable then use it until you think it's been removed
var $myTable = $('#my-table');

// To hold our new <tr> jQuery objects
var rowElements = [];

var count = data.length;
var i;
var row;
```

```
// Loop over the array
for ( i = 0; i < count; ++i ) {
    rowElements.push(
        $('<tr></tr>').append(
            $('<td></td>').html(row.type),
            $('<td></td>').html(row.content)
        )
    );
}

// Finally, insert ALL rows at once
$myTable.append(rowElements);
```

Parmi ces options, celle-ci repose le plus sur jQuery.

---

2. Utilisation des méthodes modernes de tableau. *

```
var $myTable = $('#my-table');

// Looping with the .map() method
//  - This will give us a brand new array based on the result of our callback function
var rowElements = data.map(function ( row ) {

    // Create a row
    var $row = $('<tr></tr>');

    // Create the columns
    var $type = $('<td></td>').html(row.type);
    var $content = $('<td></td>').html(row.content);

    // Add the columns to the row
    $row.append($type, $content);

    // Add to the newly-generated array
    return $row;
});

// Finally, put ALL of the rows into your table
$myTable.append(rowElements);
```

Fonctionnellement équivalent à celui précédent, plus facile à lire.

---

3. Utiliser des chaînes de HTML (au lieu des méthodes intégrées jQuery)

```
// ...
var rowElements = data.map(function ( row ) {
    var rowHTML = '<tr><td>';
    rowHTML += row.type;
    rowHTML += '</td><td>';
    rowHTML += row.content;
    rowHTML += '</td></tr>';
```

```
    return rowHTML;
});

// Using .join('') here combines all the separate strings into one
$myTable.append(rowElements.join(''));
```

Parfaitement valide mais encore une fois, **pas recommandé** . Cela oblige jQuery à analyser une très grande quantité de texte à la fois et n'est pas nécessaire. jQuery est très performant lorsqu'il est utilisé correctement.

---

4. Créer manuellement des éléments, ajouter au fragment de document

```
var $myTable = $(document.getElementById('my-table'));

/**
 * Create a document fragment to hold our columns
 * - after appending this to each row, it empties itself
 *   so we can re-use it in the next iteration.
 */
var colFragment = document.createDocumentFragment();

/**
 * Loop over the array using .reduce() this time.
 * We get a nice, tidy output without any side-effects.
 *  - In this example, the result will be a
 *    document fragment holding all the <tr> elements.
 */
var rowFragment = data.reduce(function ( fragment, row ) {

    // Create a row
    var rowEl = document.createElement('tr');

    // Create the columns and the inner text nodes
    var typeEl = document.createElement('td');
    var typeText = document.createTextNode(row.type);
    typeEl.appendChild(typeText);

    var contentEl = document.createElement('td');
    var contentText = document.createTextNode(row.content);
    contentEl.appendChild(contentText);

    // Add the columns to the column fragment
    // - this would be useful if columns were iterated over separately
    //   but in this example it's just for show and tell.
    colFragment.appendChild(typeEl);
    colFragment.appendChild(contentEl);

    rowEl.appendChild(colFragment);

    // Add rowEl to fragment - this acts as a temporary buffer to
    // accumulate multiple DOM nodes before bulk insertion
    fragment.appendChild(rowEl);

    return fragment;
}, document.createDocumentFragment());
```

```
// Now dump the whole fragment into your table
$myTable.append(rowFragment);
```

Mon préféré Cela illustre une idée générale de ce que fait jQuery à un niveau inférieur.

Plonger plus profond

- Visualiseur de source jQuery
- Array.prototype.join ()
- Array.prototype.map ()
- Array.prototype.reduce ()
- document.createDocumentFragment ()
- document.createTextNode ()
- Google Web Fundamentals - Performances

jQuery ajoute

HTML

```
<p>This is a nice </p>
<p>I like </p>

<ul>
  <li>List item 1</li>
  <li>List item 2</li>
  <li>List item 3</li>
</ul>

<button id="btn-1">Append text</button>
<button id="btn-2">Append list item</button>
```

Scénario

```
$("#btn-1").click(function(){
    $("p").append(" <b>Book</b>.");
});
$("#btn-2").click(function(){
    $("ul").append("<li>Appended list item</li>");
 });
});
```

Lire Ajouter en ligne: https://riptutorial.com/fr/jquery/topic/1910/ajouter

Chapitre 4: Case à cocher Tout sélectionner avec cocher / décocher automatiquement les autres cases à cocher

Introduction

J'ai utilisé différents exemples et réponses de Stackoverflow pour arriver à cet exemple très simple sur la façon de gérer la case à cocher "Sélectionner tout" associée à une vérification / décocher automatiquement si l'un des états de la case à cocher du groupe change. Contrainte: L'identifiant "select all" doit correspondre aux noms d'entrée pour créer le groupe select all. Dans l'exemple, l'entrée select all ID est cbGroup1. Les noms d'entrée sont également cbGroup1

Le code est très court, pas plein de déclaration (temps et ressources).

Examples

2 cochez toutes les cases avec les cases à cocher correspondantes du groupe

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>

<p>
<input id="cbGroup1" type="checkbox">Select all
<input name="cbGroup1" type="checkbox" value="value1_1">Group1 value 1
<input name="cbGroup1" type="checkbox" value="value1_2">Group1 value 2
<input name="cbGroup1" type="checkbox" value="value1_3">Group1 value 3
</p>
<p>
<input id="cbGroup2" type="checkbox">Select all
<input name="cbGroup2" type="checkbox" value="value2_1">Group2 value 1
<input name="cbGroup2" type="checkbox" value="value2_2">Group2 value 2
<input name="cbGroup2" type="checkbox" value="value2_3">Group2 value 3
</p>

<script type="text/javascript" language="javascript">
    $("input").change(function() {
      $('input[name=\''+this.id+'\']').not(this).prop('checked', this.checked);
      $('#'+this.name).prop('checked', $('input[name=\''+this.name+'\']').length ===
$('input[name=\''+this.name+'\']').filter(':checked').length);
    });
</script>
```

Lire Case à cocher Tout sélectionner avec cocher / décocher automatiquement les autres cases à cocher en ligne: https://riptutorial.com/fr/jquery/topic/10076/case-a-cocher-tout-selectionner-avec-cocher---decocher-automatiquement-les-autres-cases-a-cocher

Chapitre 5: Chaque fonction

Examples

Utilisation de base

```
// array
var arr = [
   'one',
   'two',
   'three',
   'four'
];
$.each(arr, function (index, value) {
  console.log(value);

  // Will stop running after "three"
  return (value !== 'three');
});
// Outputs: one two three
```

jQuery chaque fonction

HTML:

```
<ul>
  <li>Mango</li>
  <li>Book</li>
</ul>
```

Scénario:

```
$( "li" ).each(function( index ) {
  console.log( index + ": " + $( this ).text() );
});
```

Un message est ainsi enregistré pour chaque élément de la liste:

0: mangue

1: livre

Lire Chaque fonction en ligne: https://riptutorial.com/fr/jquery/topic/10853/chaque-fonction

Chapitre 6: événement prêt pour le document

Examples

Qu'est-ce qu'un document prêt et comment l'utiliser?

Le code jQuery est souvent enveloppé dans `jQuery(function($) { ... });` de sorte qu'il ne s'exécute qu'après le chargement du DOM.

```
<script type="text/javascript">
  jQuery(function($) {
    // this will set the div's text to "Hello".
    $("#myDiv").text("Hello");
  });
</script>

<div id="myDiv">Text</div>
```

Ceci est important car jQuery (et JavaScript en général) ne peut pas sélectionner un élément DOM qui n'a pas été rendu à la page.

```
<script type="text/javascript">
  // no element with id="myDiv" exists at this point, so $("#myDiv") is an
  // empty selection, and this will have no effect
  $("#myDiv").text("Hello");
</script>

<div id="myDiv">Text</div>
```

Notez que vous pouvez alias l'espace de noms jQuery en transmettant un gestionnaire personnalisé à la méthode `.ready()` . Cela est utile dans les cas où une autre bibliothèque JS utilise le même alias `$` abrégé que *jQuery* , ce qui crée un conflit. Pour éviter ce conflit, vous devez appeler `$.noConflict();` - Cela vous oblige à utiliser uniquement l'espace de noms *jQuery* par défaut (au lieu du court alias `$`).
En transmettant un gestionnaire personnalisé au gestionnaire `.ready()` , vous pourrez choisir le nom d'alias à utiliser avec *jQuery* .

```
$.noConflict();

jQuery( document ).ready(function( $ ) {
   // Here we can use '$' as jQuery alias without it conflicting with other
   // libraries that use the same namespace
   $('body').append('<div>Hello</div>')
});

jQuery( document ).ready(function( jq ) {
   // Here we use a custom jQuery alias 'jq'
   jq('body').append('<div>Hello</div>')
});
```

Plutôt que de simplement placer votre code jQuery au bas de la page, l'utilisation de la fonction `$(document).ready` garantit que tous les éléments HTML ont été rendus et que l'ensemble du modèle d'objet de document (DOM) est prêt à l'exécution du code JavaScript.

jQuery 2.2.3 et versions antérieures

Celles-ci sont toutes équivalentes, le code à l'intérieur des blocs sera exécuté lorsque le document sera prêt:

```
$(function() {
  // code
});

$().ready(function() {
  // code
});

$(document).ready(function() {
  // code
});
```

Comme ils sont équivalents, le premier est le formulaire recommandé, ce qui suit est une version du mot clé `jQuery` au lieu du `$` qui produit les mêmes résultats:

```
jQuery(function() {
  // code
});
```

jQuery 3.0

Notation

A partir de jQuery 3.0, seul ce formulaire est recommandé:

```
jQuery(function($) {
  // Run when document is ready
  // $ (first argument) will be internal reference to jQuery
  // Never rely on $ being a reference to jQuery in the global namespace
});
```

Tous les autres gestionnaires de documents sont déconseillés dans jQuery 3.0.

Asynchrone

A partir de jQuery 3.0, le gestionnaire prêt sera toujours appelé de manière asynchrone . Cela signifie que dans le code ci-dessous, le journal «gestionnaire externe» sera toujours affiché en premier, que le document soit prêt au moment de l'exécution.

```
$(function() {
```

```
  console.log("inside handler");
});
console.log("outside handler");
```

> gestionnaire externe
> gestionnaire interne

Différence entre $ (document) .ready () et $ (window) .load ()

`$(window).load()` est `$(window).load()` **obsolète dans jQuery version 1.8 (et complètement supprimé de jQuery 3.0)** et ne devrait donc plus être utilisé. Les raisons de la dépréciation sont notées sur la page jQuery à propos de cet événement

> Mises en garde de l'événement de chargement lorsqu'il est utilisé avec des images
>
> Un défi commun que les développeurs tentent de résoudre en utilisant le raccourci `.load()` consiste à exécuter une fonction lorsqu'une image (ou une collection d'images) est complètement chargée. Il existe plusieurs mises en garde connues à ce sujet. Ceux-ci sont:
>
> - Il ne fonctionne pas de manière cohérente ni fiable
> - Il ne se déclenche pas correctement dans WebKit si l'image `src` est définie sur le même `src` qu'auparavant
> - Il ne remplit pas correctement l'arborescence DOM
> - Peut cesser de tirer pour des images qui vivent déjà dans le cache du navigateur

Si vous souhaitez toujours utiliser `load()` il est décrit ci-dessous:

`$(document).ready()` attend que le DOM complet soit disponible - tous les éléments du HTML ont été analysés et se trouvent dans le document. Toutefois, les ressources telles que les images peuvent ne pas être complètement chargées à ce stade. S'il est important d'attendre que toutes les ressources soient chargées, `$(window).load()` **et que vous êtes conscient des limitations significatives de cet événement, vous** pouvez utiliser les éléments ci-dessous:

```
$(document).ready(function() {
  console.log($("#my_large_image").height()); // may be 0 because the image isn't available
});

$(window).load(function() {
  console.log($("#my_large_image").height()); // will be correct
});
```

Joindre des événements et manipuler le DOM à l'intérieur prêt ()

Exemple d'utilisations de `$(document).ready()` :

1. **Attacher des gestionnaires d'événements**
   Joindre des gestionnaires d'événement jQuery

```
$(document).ready(function() {
  $("button").click(function() {
    // Code for the click function
  });
});
```

2. **Exécutez le code jQuery après la création de la structure de page**

```
jQuery(function($) {
// set the value of an element.
   $("#myElement").val("Hello");
});
```

3. **Manipuler la structure DOM chargée**
   Par exemple: masquez un `div` lorsque la page se charge pour la première fois et affichez-le sur l'événement de clic d'un bouton

```
$(document).ready(function() {
  $("#toggleDiv").hide();
  $("button").click(function() {
    $("#toggleDiv").show();
  });
});
```

Différence entre jQuery (fn) et l'exécution de votre code avant

L'utilisation de l'événement prêt pour le document peut avoir de petits inconvénients en termes de performances , avec une exécution retardée allant jusqu'à ~ 300 ms. Parfois, le même comportement peut être obtenu en exécutant du code juste avant la `</body>` fermeture `</body>` :

```
<body>
  <span id="greeting"></span> world!
  <script>
    $("#greeting").text("Hello");
  </script>
</body>
```

produira un comportement similaire mais fonctionnera plus tôt que s'il n'attendait pas le déclencheur d'événement prêt pour le document comme il le fait dans:

```
<head>
  <script>
    jQuery(function($) {
      $("#greeting").text("Hello");
    });
  </script>
</head>
<body>
  <span id="greeting"></span> world!
</body>
```

L'accent est mis sur le fait que le premier exemple repose sur votre connaissance de votre page

et sur le placement du script juste avant la `</body>` fermeture `</body>` et spécifiquement après la balise `span` .

Lire événement prêt pour le document en ligne:
https://riptutorial.com/fr/jquery/topic/500/evenement-pret-pour-le-document

Chapitre 7: Événements

Remarques

jQuery gère en interne les événements via la fonction addEventListener . Cela signifie qu'il est parfaitement légal d'avoir plusieurs fonctions liées au même événement pour le même élément DOM.

Examples

Attacher et détacher des gestionnaires d'événements

Joindre un gestionnaire d'événements

Depuis la version 1.7, jQuery possède l'événement API `.on()` . Ainsi, tout événement JavaScript ou événement standard standard peut être lié à l'élément jQuery actuellement sélectionné. Il existe des raccourcis tels que `.click()` , mais `.on()` vous donne plus d'options.

HTML

```
<button id="foo">bar</button>
```

jQuery

```
$( "#foo" ).on( "click", function() {
  console.log( $( this ).text() ); //bar
});
```

Détacher un gestionnaire d'événements

Naturellement, vous avez également la possibilité de détacher des événements de vos objets jQuery. Vous le faites en utilisant `.off( events [, selector ] [, handler ] )` .

HTML

```
<button id="hello">hello</button>
```

jQuery

```
$('#hello').on('click', function(){
    console.log('hello world!');
    $(this).off();
});
```

Lorsque vous cliquez sur le bouton `$(this)` , vous vous référez à l'objet jQuery en cours et supprimez tous les gestionnaires d'événements attachés. Vous pouvez également spécifier quel gestionnaire d'événement doit être supprimé.

jQuery

```
$('#hello').on('click', function(){
    console.log('hello world!');
    $(this).off('click');
});

$('#hello').on('mouseenter', function(){
    console.log('you are about to click');
});
```

Dans ce cas, l'événement `mouseenter` fonctionnera toujours après un clic.

Événements délégués

Commençons par l'exemple. Voici un exemple très simple de HTML.

Exemple HTML

```
<html>
    <head>
    </head>
    <body>
        <ul>
            <li>
                <a href="some_url/">Link 1</a>
            </li>
            <li>
                <a href="some_url/">Link 2</a>
            </li>
            <li>
                <a href="some_url/">Link 3</a>
            </li>
        </ul>
    </body>
</html>
```

Le problème

Maintenant, dans cet exemple, nous voulons ajouter un écouteur d'événement à tous les éléments `<a>` . Le problème est que la liste dans cet exemple est dynamique. `<li>` éléments `<li>` sont ajoutés et supprimés au fil du temps. Cependant, la page ne s'actualise pas entre les modifications, ce qui nous permettrait d'utiliser des écouteurs d'événements de clic simples pour les objets de lien (par exemple, `$('a').click()`).

Le problème est de savoir comment ajouter des événements aux éléments `<a>` qui vont et viennent.

Informations générales - Propagation des événements

Les événements délégués ne sont possibles que grâce à la propagation des événements (souvent appelée événementing). Chaque fois qu'un événement est déclenché, il se déploie complètement (à la racine du document). Ils *délèguent* la gestion d'un événement à un élément ancêtre non modifié, d'où le nom d'événements "délégués".

Ainsi, dans l'exemple ci-dessus, cliquer sur le lien `<a>` élément déclenche un événement "clic" dans ces éléments dans cet ordre:

- une
- li
- ul
- corps
- html
- racine du document

Solution

Connaissant ce que font les événements, nous pouvons attraper l’un des événements recherchés qui se propagent dans notre HTML.

L’élément `<ul>` est un bon endroit pour le capturer dans cet exemple, car cet élément n’est pas dynamique:

```
$('ul').on('click', 'a', function () {
  console.log(this.href); // jQuery binds the event function to the targeted DOM element
                          // this way `this` refers to the anchor and not to the list
  // Whatever you want to do when link is clicked
});
```

En haut:

- Nous avons 'ul' qui est le destinataire de cet écouteur d'événement
- Le premier paramètre ('click') définit les événements que nous essayons de détecter.
- Le second paramètre ('a') est utilisé pour déclarer d'où doit *provenir* l'événement (de tous les éléments enfants sous le destinataire de cet écouteur d'événement, ul).
- Enfin, le troisième paramètre est le code exécuté si les exigences des premier et second paramètres sont remplies.

En détail comment fonctionne la solution

1. L'utilisateur clique sur l'élément `<a>`
2. Cela déclenche un événement click sur l'élément `<a>` .
3. L'événement commence à bouillonner vers la racine du document.
4. L'événement entre en premier dans l'élément `<li>` , puis dans l'élément `<ul>` .
5. L'écouteur d'événement est exécuté en tant `<ul>` événement associé à l'élément `<ul>` .
6. L'écouteur d'événement détecte d'abord l'événement déclencheur. L'événement bouillonnant est "clic" et l'auditeur a un "clic", c'est une passe.
7. L'auditeur vérifie que le second paramètre («a») correspond à chaque élément de la chaîne de bulles. Comme le dernier élément de la chaîne est un "a", cela correspond au filtre et c'est aussi une passe.
8. Le code du troisième paramètre est exécuté en utilisant l'élément correspondant tel qu'il se `this` . Si la fonction n'inclut pas un appel à `stopPropagation()` , l'événement continuera à se propager vers la racine (`document`).

Remarque: Si un ancêtre approprié ne changeant pas n'est pas disponible / pratique, vous devez utiliser `document` . Comme habitude, n'utilisez pas le `'body'` pour les raisons suivantes:

- `body` a un bogue, en rapport avec le style, qui peut signifier que les événements de la souris ne bouillonnent pas. Cela dépend du navigateur et peut se produire lorsque la hauteur du corps calculée est 0 (par exemple, lorsque tous les éléments enfants ont des positions absolues). Les événements de souris bouillonnent toujours pour `document` .
- `document` existe *toujours* pour votre script, vous pouvez donc attacher des gestionnaires délégués à `document` dehors d'un *gestionnaire prêt pour DOM et assurez-vous* qu'ils fonctionneront toujours.

Evénement de chargement de document .load ()

Si vous souhaitez que votre script attende le chargement d'une certaine ressource, telle qu'une image ou un PDF, vous pouvez utiliser `.load()` , raccourci pour le raccourci pour `.on( "load", handler)` .

HTML

```
<img src="image.jpeg" alt="image" id="image">
```

jQuery

```
$( "#image" ).load(function() {
  // run script
});
```

Evénements pour répéter des éléments sans utiliser d'identifiant

Problème

Il y a une série d'éléments répétés dans la page que vous devez savoir sur lesquels un événement s'est produit pour faire quelque chose avec cette instance spécifique.

Solution

- Donne à tous les éléments communs une classe commune
- Appliquer un écouteur d'événement à une classe. `this` gestionnaire d'événements interne est l'élément de sélecteur correspondant à l'événement sur lequel il s'est produit
- Traverse de récipient le plus externe de répétition pour cette instance en commençant par `this`
- Utilisez `find()` dans ce conteneur pour isoler d'autres éléments spécifiques à cette instance.

HTML

```
<div class="item-wrapper" data-item_id="346">
   <div class="item"><span class="person">Fred</span></div>
   <div class="item-toolbar">
      <button class="delete">Delete</button>
   </div>
</div>
<div class="item-wrapper" data-item_id="393">
   <div clss="item"><span class="person">Wilma</span></div>
   <div class="item-toolbar">
      <button class="delete">Delete</button>
   </div>
</div>
```

jQuery

```
$(function() {
  $('.delete').on('click', function() {
    // "this" is element event occured on
    var $btn = $(this);
    // traverse to wrapper container
    var $itemWrap = $btn.closest('.item-wrapper');
    // look within wrapper to get person for this button instance
    var person = $itemWrap.find('.person').text();
    // send delete to server and remove from page on success of ajax
    $.post('url/string', { id: $itemWrap.data('item_id')}).done(function(response) {
      $itemWrap.remove()
    }).fail(function() {
      alert('Ooops, not deleted at server');
    });
  });
```

```
});
```

originalEvent

Parfois, il y aura des propriétés qui ne sont pas disponibles dans l'événement jQuery. Pour accéder aux propriétés sous-jacentes, utilisez `Event.originalEvent`

Obtenir la direction du défilement

```
$(document).on("wheel",function(e){
    console.log(e.originalEvent.deltaY)
    // Returns a value between -100 and 100 depending on the direction you are scrolling
})
```

Activer et désactiver des événements spécifiques via jQuery. (Auditeurs nommés)

Parfois, vous souhaitez désactiver tous les écouteurs précédemment enregistrés.

```
//Adding a normal click handler
$(document).on("click",function(){
    console.log("Document Clicked 1")
});
//Adding another click handler
$(document).on("click",function(){
    console.log("Document Clicked 2")
});
//Removing all registered handlers.
$(document).off("click")
```

Un problème avec cette méthode est que TOUS les écouteurs liés au `document` par d'autres plugins, etc. seraient également supprimés.

Le plus souvent, nous voulons détacher tous les auditeurs attachés que par nous.

Pour ce faire, nous pouvons lier des auditeurs nommés comme,

```
//Add named event listener.
$(document).on("click.mymodule",function(){
    console.log("Document Clicked 1")
});
$(document).on("click.mymodule",function(){
    console.log("Document Clicked 2")
});

//Remove named event listener.
$(document).off("click.mymodule");
```

Cela garantit que tout autre écouteur de clic n'est pas modifié par inadvertance.

Lire Événements en ligne: https://riptutorial.com/fr/jquery/topic/1321/evenements

Chapitre 8: jQuery Objets différés et promesses

Introduction

Les promesses de jQuery sont un moyen astucieux de chaîner des opérations asynchrones dans un bloc de construction. Cela remplace la nidification à l'ancienne des rappels, qui ne sont pas facilement réorganisés.

Examples

Création de promesse de base

Voici un exemple très simple d'une fonction qui " *promet* de procéder quand un temps donné s'écoule". Il le fait en créant un nouvel objet `Deferred` , qui est résolu plus tard et renvoyant la promesse du différé:

```
function waitPromise(milliseconds){

   // Create a new Deferred object using the jQuery static method
   var def = $.Deferred();

   // Do some asynchronous work - in this case a simple timer
   setTimeout(function(){

       // Work completed... resolve the deferred, so it's promise will proceed
       def.resolve();
   }, milliseconds);

   // Immediately return a "promise to proceed when the wait time ends"
   return def.promise();
}
```

Et utiliser comme ça:

```
waitPromise(2000).then(function(){
     console.log("I have waited long enough");
});
```

Promesses asynchrones

Si vous devez exécuter plusieurs tâches asynchrones l'une après l'autre, vous devrez regrouper les objets promis. Voici un exemple simple:

```
function First() {
    console.log("Calling Function First");
    return $.get("/ajax/GetFunction/First");
}
```

```
function Second() {
    console.log("Calling Function Second");
    return $.get("/ajax/GetFunction/Second");
}

function Third() {
    console.log("Calling Function Third");
    return $.get("/ajax/GetFunction/Third");
}

function log(results){
    console.log("Result from previous AJAX call: " + results.data);
}

First().done(log)
       .then(Second).done(log)
       .then(Third).done(log);
```

jQuery ajax () succès, erreur VS .done (), .fail ()

succès et erreur: Un rappel de **succès** qui est appelé lors de la réussite d'une requête Ajax.

Un rappel d' **échec** qui est appelé en cas d'erreur lors de la requête.

Exemple:

```
 $.ajax({
        url: 'URL',
        type: 'POST',
        data: yourData,
        datatype: 'json',
        success: function (data) { successFunction(data); },
        error: function (jqXHR, textStatus, errorThrown) { errorFunction(); }
    });
```

.done () et .fail ():

.ajax (). done (function (data, textStatus, jqXHR) {}); Remplace la méthode .success () qui était déconseillée dans jQuery 1.8.Ceci est une construction alternative pour la fonction de rappel réussie ci-dessus.

.ajax (). fail (fonction (jqXHR, textStatus, errorThrown) {}); Remplace la méthode .error () qui était déconseillée dans jQuery 1.8.Ceci est une construction alternative pour la fonction de rappel complète ci-dessus.

Exemple:

```
$.ajax({
    url: 'URL',
    type: 'POST',
    data: yourData,
    datatype: 'json'
})
.done(function (data) { successFunction(data); })
```

```
.fail(function (jqXHR, textStatus, errorThrown) { serrorFunction(); });
```

Obtenez l'état actuel d'une promesse

Par défaut, l'état d'une promesse est en attente lors de sa création. L'état d'une promesse est modifié lorsque l'objet différé qui a créé la promesse soit résolu ou rejeté.

```
var deferred = new $.Deferred();
var d1= deferred.promise({
    prop: "value"
});
var d2= $("div").promise();
var d3= $("div").hide(1000).promise();

console.log(d1.state()); // "pending"
console.log(d2.state()); // "resolved"
console.log(d3.state()); // "pending"
```

Lire jQuery Objets différés et promesses en ligne:
https://riptutorial.com/fr/jquery/topic/8308/jquery-objets-differes-et-promesses

Chapitre 9: Les attributs

Remarques

La fonction jQuery `.attr()` , obtient la valeur d'un attribut pour le **premier** élément de l'ensemble des éléments correspondants ou définit un ou plusieurs attributs pour **chaque** élément correspondant.

Il convient de noter que lorsqu'on obtient la valeur d'un attribut, il ne l'obtient que du premier élément qui correspond au sélecteur (c.- `$("input").attr("type");` d. `$("input").attr("type");` , s'il y en a plus d'un

Cependant, lors de la définition d'un attribut, il sera appliqué à tous les éléments correspondants.

Examples

Récupère la valeur d'attribut d'un élément HTML

Lorsqu'un seul paramètre est passé à la fonction `.attr()` , il renvoie la valeur de l'attribut transmis sur l'élément sélectionné.

Syntaxe:

```
$([selector]).attr([attribute name]);
```

Exemple:

HTML:

```
<a href="/home">Home</a>
```

jQuery:

```
$('a').attr('href');
```

Récupération `data` attributs de `data` :

jQuery propose la fonction `.data()` pour gérer les attributs de données. `.data` fonction `.data` renvoie la valeur de l'attribut data sur l'élément sélectionné.

Syntaxe:

```
$([selector]).data([attribute name]);
```

Exemple:

Html:

```
<article data-column="3"></article>
```

jQuery:

```
$("article").data("column")
```

Remarque:

> La méthode data () de jQuery vous donnera accès aux attributs data- *, MAIS, elle interrompt la casse du nom de l'attribut. Référence

Valeur de réglage de l'attribut HTML

Si vous souhaitez ajouter un attribut à un élément, vous pouvez utiliser la fonction `attr(attributeName, attributeValue)` . Par exemple:

```
$('a').attr('title', 'Click me');
```

Cet exemple ajoute le texte `"Click me"` au passage de la souris à tous les liens de la page.

La même fonction est utilisée pour modifier les valeurs des attributs.

Supprimer l'attribut

Pour supprimer un attribut d'un élément, vous pouvez utiliser la fonction `.removeAttr(attributeName)` . Par exemple:

```
$('#home').removeAttr('title');
```

Cela supprimera l'attribut `title` de l'élément avec ID `home` .

Differece entre attr () et prop ()

`attr()` obtient / définit l'attribut HTML à l'aide des fonctions DOM `getAttribute()` et `setAttribute()` . `prop()` fonctionne en définissant la propriété DOM sans modifier l'attribut. Dans de nombreux cas, les deux sont interchangeables, mais il en faut parfois un sur l'autre.

Pour définir une case à cocher comme cochée:

```
$('#tosAccept').prop('checked', true); // using attr() won't work properly here
```

Pour supprimer une propriété, vous pouvez utiliser la méthode `removeProp()` . De même, `removeAttr()` supprime les attributs.

Lire Les attributs en ligne: https://riptutorial.com/fr/jquery/topic/4429/les-attributs

Chapitre 10: Manipulation CSS

Syntaxe

- .css (cssProperty) // Récupère la valeur de la propriété CSS rendue
- .css ([cssProperty, ...]) // Récupère les valeurs du tableau de cssProperties
- .css (cssProperty, valeur) // Définir la valeur
- .css ({cssProperty: value, ...}) // Définit les propriétés et les valeurs
- .css (cssProperty, function) // Exposer le cssProperty à une fonction de rappel

Remarques

Valeurs rendues

Si une unité réactive est utilisée (comme `"auto"` , `"%"` , `"vw"` etc.), `.css()` renverra la valeur réelle rendue en `px`

```
.myElement{ width: 20%; }
```

```
var width = $(".myElement").css("width"); // "123px"
```

Mise en forme des propriétés et des valeurs

Les propriétés peuvent être définies en utilisant **le formatage CSS standard en tant que chaîne** ou en utilisant **camelCase**

```
"margin-bottom"
marginBottom
```

Les valeurs doivent être exprimées en chaîne. Les valeurs numériques sont traitées comme des unités `px` interne par jQuery

```
.css(fontSize: "1em")
.css(fontSize: "16px")
.css(fontSize: 16)       // px will be used
```

A partir de jQuery 3, évitez d'utiliser `.show()` et `.hide()`

Selon cet article de blog jQuery , en raison de problèmes de performances et de surcharge, vous ne devriez plus utiliser `.show()` ou `.hide()` .

> Si vous avez des éléments dans une feuille de style qui sont définis pour `display: none` , la méthode `.show()` ne remplacera plus cela. La règle la plus importante pour passer à jQuery 3.0 est la suivante: n'utilisez pas de feuille de style pour définir l' `display: none`

par défaut `display: none` et essayez ensuite d'utiliser `.show()` - ou toute méthode `.slideDown()` éléments, tels que `.slideDown()` et `.fadeIn()` - pour le rendre visible. Si vous avez besoin de masquer un élément par défaut, le meilleur moyen est d'ajouter un nom de classe comme «caché» à l'élément et de définir cette classe comme étant `display: none` dans une feuille de style. Vous pouvez ensuite ajouter ou supprimer cette classe à l'aide des `.addClass()` et `.removeClass()` jQuery pour contrôler la visibilité. Alternativement, vous pouvez avoir un appel au gestionnaire `.ready()` `.hide()` sur les éléments avant qu'ils ne soient affichés sur la page. Ou, si vous devez conserver la valeur par défaut de la feuille de style, vous pouvez utiliser `.css("display", "block")` (ou la valeur d'affichage appropriée) pour remplacer la feuille de style.

Examples

Définir la propriété CSS

Définition d'un seul style:

```
$('#target-element').css('color', '#000000');
```

Définition de plusieurs styles en même temps:

```
$('#target-element').css({
    'color': '#000000',
    'font-size': '12pt',
    'float': 'left',
});
```

Obtenir la propriété CSS

Pour obtenir la propriété CSS d'un élément, vous pouvez utiliser la méthode `.css(propertyName)` :

```
var color    = $('#element').css('color');
var fontSize = $('#element').css('font-size');
```

Incrémenter / Décrémenter les propriétés numériques

Les propriétés CSS numériques peuvent être incrémentées et décrémentées avec la syntaxe `+=` et `-=` , respectivement, en utilisant la méthode `.css()` :

```
  // Increment using the += syntax
  $("#target-element").css("font-size", "+=10");

  // You can also specify the unit to increment by
  $("#target-element").css("width", "+=100pt");
  $("#target-element").css("top", "+=30px");
  $("#target-element").css("left", "+=3em");

  // Decrementing is done by using the -= syntax
  $("#target-element").css("height", "-=50pt");
```

CSS Getter

La fonction **getter** `.css()` peut être appliquée à chaque élément DOM de la page comme suit:

```
// Rendered width in px as a string. ex: `150px`
// Notice the `as a string` designation - if you require a true integer,
// refer to `$.width()` method
$("body").css("width");
```

Cette ligne renverra la **largeur calculée** de l'élément spécifié, chaque propriété CSS que vous fournissez entre parenthèses donnera la valeur de la propriété pour cet élément DOM `$("selector")` , si vous demandez un attribut CSS qui n'existe pas sera `undefined` comme une réponse.

Vous pouvez également appeler le **getter CSS** avec un tableau d'attributs:

```
$("body").css(["animation","width"]);
```

Cela retournera un objet de tous les attributs avec leurs valeurs:

```
Object {animation: "none 0s ease 0s 1 normal none running", width: "529px"}
```

Setter CSS

La méthode **setter** `.css()` peut également être appliquée à chaque élément DOM de la page.

```
$("selector").css("width", 500);
```

Cette instruction définit la `width` du `$("selector")` sur `500px` et renvoie l'objet jQuery pour que vous puissiez chaîner plus de méthodes au sélecteur spécifié.

Le **setter** `.css()` peut également être utilisé en passant un objet de propriétés CSS et des valeurs telles que:

```
$("body").css({"height": "100px", width:100, "padding-top":40, paddingBottom:"2em"});
```

Toutes les modifications apportées par le dispositif de réglage sont ajoutées à la propriété de `style` élément DOM `style` affectant ainsi les styles des éléments (à moins que la valeur de la propriété de style ne soit déjà définie comme `!important` ailleurs dans les styles).

Lire Manipulation CSS en ligne: https://riptutorial.com/fr/jquery/topic/2732/manipulation-css

Chapitre 11: Manipulation DOM

Examples

Créer des éléments DOM

La fonction `jQuery` (généralement appelée alias `$`) peut être utilisée à la fois pour sélectionner des éléments et pour créer de nouveaux éléments.

```
var myLink = $('<a href="http://stackexchange.com"></a>');
```

Vous pouvez éventuellement passer un deuxième argument avec des attributs d'élément:

```
var myLink = $('<a>', { 'href': 'http://stackexchange.com' });
```

`'<a>'` -> Le premier argument spécifie le type d'élément DOM que vous souhaitez créer. Dans cet exemple, il s'agit d'une ancre mais pourrait être n'importe quoi sur cette liste . Voir la spécification d'une référence de l' `a` élément.

`{ 'href': 'http://stackexchange.com' }` -> le second argument est un objet JavaScript contenant des paires nom / valeur d'attribut.

les paires 'name': 'value' apparaîtront entre le `< >` du premier argument, par exemple `<a name:value>` qui, dans notre exemple, serait `<a href="http://stackexchange.com"></a>`

Manipulation des classes d'éléments

En supposant que la page inclut un élément HTML tel que:

```
<p class="small-paragraph">
  This is a small <a href="https://en.wikipedia.org/wiki/Paragraph">paragraph</a>
  with a <a class="trusted" href="http://stackexchange.com">link</a> inside.
</p>
```

jQuery fournit des fonctions utiles pour manipuler les classes DOM, notamment `hasClass()` , `addClass()` , `removeClass()` et `toggleClass()` . Ces fonctions modifient directement l'attribut de `class` des éléments correspondants.

```
$('p').hasClass('small-paragraph'); // true
$('p').hasClass('large-paragraph'); // false

// Add a class to all links within paragraphs
$('p a').addClass('untrusted-link-in-paragraph');

// Remove the class from a.trusted
$('a.trusted.untrusted-link-in-paragraph')
.removeClass('untrusted-link-in-paragraph')
.addClass('trusted-link-in-paragraph');
```

Basculer une classe

Étant donné le balisage d'exemple, nous pouvons ajouter une classe avec notre premier `.toggleClass()` :

```
$(".small-paragraph").toggleClass("pretty");
```

Maintenant ceci retournerait `true` : `$(".small-paragraph").hasClass("pretty")`

`toggleClass` fournit le même effet avec moins de code que:

```
if($(".small-paragraph").hasClass("pretty")){
   $(".small-paragraph").removeClass("pretty");}
else {
   $(".small-paragraph").addClass("pretty"); }
```

basculer deux classes:

```
$(".small-paragraph").toggleClass("pretty cool");
```

Booléen pour ajouter / supprimer des classes:

```
$(".small-paragraph").toggleClass("pretty",true); //cannot be truthy/falsey

$(".small-paragraph").toggleClass("pretty",false);
```

Fonction de basculement de classe (voir exemple plus bas pour éviter un problème)

```
$( "div.surface" ).toggleClass(function() {
  if ( $( this ).parent().is( ".water" ) ) {
    return "wet";
  } else {
    return "dry";
  }
});
```

Utilisé dans des exemples:

```
// functions to use in examples
function stringContains(myString, mySubString) {
  return myString.indexOf(mySubString) !== -1;
}
function isOdd(num) { return num % 2;}
var showClass = true; //we want to add the class
```

Exemples:

Utilisez l'index d'élément pour basculer les classes impaires / paires

```
$( "div.gridrow" ).toggleClass(function(index,oldClasses, false), showClass ) {
  showClass
```

```
  if ( isOdd(index) ) {
    return "wet";
  } else {
    return "dry";
  }
});
```

Exemple de `toggleClass` plus complexe, avec un balisage simple

```
<div class="grid">
  <div class="gridrow">row</div>
  <div class="gridrow">row</div>
  <div class="gridrow">row</div>
  <div class="gridrow">row</div>
  <div class="gridrow">row</div>
  <div class="gridrow gridfooter">row but I am footer!</div>
</div>
```

Fonctions simples pour nos exemples:

```
function isOdd(num) {
  return num % 2;
}

function stringContains(myString, mySubString) {
  return myString.indexOf(mySubString) !== -1;
}
var showClass = true; //we want to add the class
```

Ajouter une classe impair / paire aux éléments avec une classe `gridrow`

```
$("div.gridrow").toggleClass(function(index, oldClasses, showThisClass) {
  if (isOdd(index)) {
    return "odd";
  } else {
    return "even";
  }
  return oldClasses;
}, showClass);
```

Si la ligne a une classe `gridfooter` , supprimez les classes impaires / paires, conservez le reste.

```
$("div.gridrow").toggleClass(function(index, oldClasses, showThisClass) {
  var isFooter = stringContains(oldClasses, "gridfooter");
  if (isFooter) {
    oldClasses = oldClasses.replace('even', ' ').replace('odd', ' ');
    $(this).toggleClass("even odd", false);
  }
  return oldClasses;
}, showClass);
```

Les classes renvoyées sont ce qui est effectué. Ici, si un élément ne possède pas de `gridfooter` , ajoutez une classe pour pair / impair. Cet exemple illustre le retour de la liste de classes OLD. Si cela d' `else return oldClasses;` est supprimé, seules les nouvelles classes sont ajoutées, ainsi la

ligne avec une classe `gridfooter` aura toutes les classes supprimées si nous n'avions pas retourné ces anciennes - elles auraient été basculées (supprimées) sinon.

```
$("div.gridrow").toggleClass(function(index, oldClasses, showThisClass) {
  var isFooter = stringContains(oldClasses, "gridfooter");
  if (!isFooter) {
    if (isOdd(index)) {
      return "oddLight";
    } else {
      return "evenLight";
    }
  } else return oldClasses;
}, showClass);
```

Autres méthodes API

jQuery offre une variété de méthodes qui peuvent être utilisées pour la manipulation DOM.

Le premier est la méthode .empty () .

Imaginez le balisage suivant:

```
<div id="content">
  <div>Some text</div>
</div>
```

En appelant `$('#content').empty();` , la div interne serait supprimée. Cela pourrait également être réalisé en utilisant `$('#content').html('');` .

Une autre fonction pratique est la fonction .closest () :

```
<tr id="row_1">
  <td><button type="button" class="delete">Delete</button>
</tr>
```

Si vous souhaitez trouver la ligne la plus proche d'un bouton sur lequel vous avez cliqué dans l'une des cellules de ligne, vous pouvez le faire:

```
$('.delete').click(function() {
  $(this).closest('tr');
});
```

Comme il y aura probablement plusieurs lignes, chacune avec ses propres boutons de `delete` , nous utilisons `$(this)` dans la fonction .click () pour limiter la portée au bouton sur lequel nous avons cliqué.

Si vous voulez obtenir l' `id` de la ligne contenant le bouton `Delete` sur lequel vous avez cliqué, vous pouvez le faire comme suit:

```
$('.delete').click(function() {
  var $row = $(this).closest('tr');
```

```
  var id = $row.attr('id');
});
```

Il est généralement considéré comme une bonne pratique à des variables de préfixe contenant des objets jQuery avec un `$` (signe dollar) pour préciser ce que la variable est.

Une alternative à `.closest()` est la méthode .parents () :

```
$('.delete').click(function() {
  var $row = $(this).parents('tr');
  var id = $row.attr('id');
});
```

et il y a aussi une fonction .parent () aussi:

```
$('.delete').click(function() {
  var $row = $(this).parent().parent();
  var id = $row.attr('id');
});
```

`.parent()` ne monte que d'un niveau de l'arborescence DOM, donc il est assez rigide, si vous deviez changer le bouton de suppression pour qu'il soit contenu dans un `span` par exemple, alors le sélecteur jQuery serait cassé.

.html ()

Vous pouvez utiliser cette méthode pour remplacer tout le code HTML du sélecteur. En supposant que vous avez un élément HTML comme celui-ci

```
<div class="row">
    <div class="col-md-12">
        <div id="information">
            <p>Old message</p>
        </div>
    </div>
</div>
```

Vous pouvez utiliser `.html()` . supprimer et ajouter un texte d'alerte ou d'information pour alerter les utilisateurs avec une seule ligne.

```
$("#information").html("<p>This is the new alert!</p>");
```

Éléments de tri

Pour trier efficacement les éléments (en une seule fois et avec une interruption minimale du DOM), nous devons:

1. **Trouver** les éléments
2. **Trier en** fonction d'une condition définie
3. **Insérer** dans le DOM

```
<ul id='my-color-list'>
    <li class="disabled">Red</li>
    <li>Green</li>
    <li class="disabled">Purple</li>
    <li>Orange</li>
</ul>
```

1. Les trouver - `.children()` ou `.find()`

   Cela nous redonnera un objet de type Array pour jouer avec.

   ```
   var $myColorList = $('#my-color-list');

   // Elements one layer deep get .children(), any deeper go with .find()
   var $colors = $myColorList.children('li');
   ```

2. `Array.prototype.sort()` -les - `Array.prototype.sort()`

   Ceci est actuellement configuré pour renvoyer les éléments dans l'ordre croissant en fonction du contenu HTML (aka leurs couleurs).

   ```
   /**
    * Bind $colors to the sort method so we don't have to travel up
    * all these properties more than once.
    */
   var sortList = Array.prototype.sort.bind($colors);

   sortList(function ( a, b ) {

       // Cache inner content from the first element (a) and the next sibling (b)
       var aText = a.innerHTML;
       var bText = b.innerHTML;

       // Returning -1 will place element `a` before element `b`
       if ( aText < bText ) {
           return -1;
       }

       // Returning 1 will do the opposite
       if ( aText > bText ) {
           return 1;
       }

       // Returning 0 leaves them as-is
       return 0;
   });
   ```

3. Insérez-les - `.append()`

   Notez que nous n'avons pas besoin de détacher les éléments en premier - `append()` déplacera les éléments qui existent déjà dans le DOM, nous n'aurons donc pas de copies supplémentaires

   ```
   // Put it right back where we got it
   $myColorList.append($colors);
   ```

Le rendre mignon

Ajouter un bouton de tri

```
<!-- previous HTML above -->
<button type='button' id='btn-sort'>
    Sort
</button>
```

Définir la valeur initiale du sens de tri

```
var ascending = true;
```

`sortList()` nos éléments DOM et `sortList()` pour minimiser notre traitement DOM

```
var $myColorList = $('#my-color-list');
var $colors = $myColorList.children('li');
var sortList = Array.prototype.sort.bind($colors);
```

Emballez le tout dans une fonction `doSort()`

```
// Put the sortList() and detach/append calls in this portable little thing
var doSort = function ( ascending ) {

    sortList(function ( a, b ) {
        // ...
    });

    $myColorList.append($colors);
};
```

Ajouter un gestionnaire de clic pour `$('#btn-sort')`

```
$('#btn-sort').on('click', function () {
    // Run the sort and pass in the direction value
    doSort(ascending);

    // Toggle direction and save
    ascending = !ascending;
});
```

Tous ensemble maintenant

```
var ascending = true;

var $myColorList = $('#my-color-list');
var $colors = $myColorList.children('li');
var sortList = Array.prototype.sort.bind($colors);

var doSort = function ( ascending ) {

    sortList(function ( a, b ) {

        var aText = a.innerHTML;
        var bText = b.innerHTML;

        if ( aText < bText ) {
            return ascending ? -1 : 1;
        }

        if ( aText > bText ) {
            return ascending ? 1 : -1;
        }

        return 0;
    });

    $myColorList.append($colors);

};

$('#btn-sort').on('click', function () {
    doSort(ascending);
    ascending = !ascending;
});
```

Prime

Tri à plusieurs niveaux (regroupement d'éléments triés)

```
// ...

var doSort = function ( ascending ) {

    sortList(function ( a, b ) {

        // ...initial sorting...

    }).sort(function ( a, b ) {

        // We take the returned items and sort them once more
        var aClass = a.className;
        var bClass = b.className;

        // Let's group the disabled items together and put them at the end

        /**
         * If the two elements being compared have the same class
         * then there's no need to move anything.
         */
        if ( aClass !== bClass ) {
```

```
            return aClass === 'disabled' ? 1 : -1;
        }
        return 0;
    });

    // And finalize with re-insert
    $myColorList.append($colors);
};

// ...
```

Pouvez-vous aller plus loin?

Ajouter un autre bouton pour basculer le tri de groupe désactivé

MDN - Array.prototype.sort ()

Lire Manipulation DOM en ligne: https://riptutorial.com/fr/jquery/topic/512/manipulation-dom

Chapitre 12: Méthode jQuery .animate ()

Syntaxe

1. (sélecteur) .animate ({styles}, {options})

Paramètres

Paramètre	Détails
Propriétés	Un objet de propriétés et de valeurs CSS vers lequel l'animation se déplacera
durée	(par défaut: 400) Chaîne ou nombre déterminant la durée d'exécution de l'animation
assouplir	(par défaut: swing) Chaîne indiquant la fonction d'accélération à utiliser pour la transition
Achevée	Une fonction à appeler une fois l'animation terminée, appelée une fois par élément correspondant.
début	spécifie une fonction à exécuter lorsque l'animation commence.
étape	spécifie une fonction à exécuter pour chaque étape de l'animation.
queue	une valeur booléenne spécifiant s'il faut ou non placer l'animation dans la file d'attente des effets.
le progrès	spécifie une fonction à exécuter après chaque étape de l'animation.
terminé	spécifie une fonction à exécuter à la fin de l'animation.
échouer	spécifie une fonction à exécuter si l'animation échoue.
specialEasing	une carte d'une ou plusieurs propriétés CSS du paramètre styles et leurs fonctions d'accélération correspondantes.
toujours	spécifie une fonction à exécuter si l'animation s'arrête sans se terminer.

Examples

Animation avec rappel

Parfois, nous devons changer la position des mots d'un endroit à un autre ou réduire la taille des

mots et changer automatiquement la couleur des mots pour améliorer l'attrait de notre site Web ou de nos applications Web. JQuery aide beaucoup avec ce concept en utilisant `fadeIn(), hide(), slideDown()` mais ses fonctionnalités sont limitées et il n'a fait que la tâche spécifique qui lui est assignée.

Jquery résout ce problème en fournissant une méthode étonnante et flexible appelée `.animate()` . Cette méthode permet de définir des animations personnalisées qui sont utilisées avec des propriétés css permettant de survoler les frontières. par exemple si on donne la propriété de style css en `width:200;` et la position actuelle de l'élément DOM est 50, la méthode animate réduit la valeur de la position actuelle à partir de la valeur css donnée et anime cet élément à 150.

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script>
    $("#btn1").click(function(){
        $("#box").animate({width: "200px"});
    });
</script>

<button id="btn1">Animate Width</button>
<div id="box" style="background:#98bf21;height:100px;width:100px;margin:6px;"></div>
```

Liste des propriétés de style css qui permettent la méthode `.animate()` .

```
backgroundPositionX, backgroundPositionY, borderWidth, borderBottomWidth, borderLeftWidth,
borderRightWidth, borderTopWidth, borderSpacing, margin, marginBottom, marginLeft,
marginRight, marginTop, outlineWidth, padding, paddingBottom, paddingLeft, paddingRight,
paddingTop, height, width, maxHeight, maxWidth, minHeight, minWidth, fontSize, bottom, left,
right, top, letterSpacing,  wordSpacing, lineHeight, textIndent,
```

Vitesse spécifiée dans la méthode `.animate()` .

```
milliseconds (Ex: 100, 1000, 5000, etc.),
"slow",
"fast"
```

`.animate()` spécifiée dans la méthode `.animate()` .

"balançoire"
"linéaire"

Voici quelques exemples avec des options d'animation complexes.

Exemple 1:

```
$( "#book" ).animate({
  width: [ "toggle", "swing" ],
  height: [ "toggle", "swing" ],
  opacity: "toggle"
 }, 5000, "linear", function() {
```

```
    $( this ).after( "<div>Animation complete.</div>" );
});
```

Par exemple 2:

```
  $("#box").animate({
     height: "300px",
     width: "300px"
     }, {
     duration: 5000,
     easing: "linear",
     complete: function(){
        $(this).after("<p>Animation is complete!</p>");
     }
  });
```

Lire Méthode jQuery .animate () en ligne: https://riptutorial.com/fr/jquery/topic/5064/methode-jquery--animate---

Chapitre 13: Obtenir et régler la largeur et la hauteur d'un élément

Examples

Obtenir et définir la largeur et la hauteur (en ignorant la bordure)

Obtenez la largeur et la hauteur:

```
var width = $('#target-element').width();
var height = $('#target-element').height();
```

Définir la largeur et la hauteur:

```
$('#target-element').width(50);
$('#target-element').height(100);
```

Obtenir et définir innerWidth et innerHeight (en ignorant le remplissage et la bordure)

Obtenez la largeur et la hauteur:

```
var width = $('#target-element').innerWidth();
var height = $('#target-element').innerHeight();
```

Définir la largeur et la hauteur:

```
$('#target-element').innerWidth(50);
$('#target-element').innerHeight(100);
```

Obtenir et définir outerWidth et outerHeight (y compris le remplissage et la bordure)

Obtenez la largeur et la hauteur (hors marge):

```
var width = $('#target-element').outerWidth();
var height = $('#target-element').outerHeight();
```

Obtenez la largeur et la hauteur (y compris la marge):

```
var width = $('#target-element').outerWidth(true);
var height = $('#target-element').outerHeight(true);
```

Définir la largeur et la hauteur:

```
$('#target-element').outerWidth(50);
$('#target-element').outerHeight(100);
```

Lire Obtenir et régler la largeur et la hauteur d'un élément en ligne:
https://riptutorial.com/fr/jquery/topic/2167/obtenir-et-regler-la-largeur-et-la-hauteur-d-un-element

Chapitre 14: Plugins

Examples

Plugins - Mise en route

L'API jQuery peut être étendue en ajoutant à son prototype. Par exemple, l'API existante dispose déjà de nombreuses fonctions telles que `.hide()` , `.fadeIn()` , `.hasClass()` , etc.

Le prototype jQuery est exposé via `$.fn` , le code source contient la ligne

```
jQuery.fn = jQuery.prototype
```

L'ajout de fonctions à ce prototype permettra à ces fonctions d'être disponibles pour être appelées depuis n'importe quel objet jQuery construit (ce qui se fait implicitement avec chaque appel à jQuery, ou chaque appel à $ si vous préférez).

Un objet jQuery construit contiendra un tableau interne d'éléments basé sur le sélecteur qui lui est transmis. Par exemple, `$('.active')` va créer un objet jQuery contenant des éléments avec la classe active, au moment de l'appel (comme dans, il ne s'agit pas d'un ensemble d'éléments).

La valeur `this` intérieur de la fonction plugin fera référence à l'objet jQuery construit. Par conséquent, `this` est utilisé pour représenter l'ensemble correspondant.

Plugin de base :

```
$.fn.highlight = function() {
    this.css({ background: "yellow" });
};

// Use example:
$("span").highlight();
```

jsFiddle exemple

Chaînabilité et réutilisabilité

Contrairement à l'exemple ci - dessus , les **plug-ins** jQuery sont censés être **Chainable** . Cela signifie qu'il est possible de chaîner plusieurs méthodes dans une même collection d'éléments, comme `$(".warn").append("WARNING! ").css({color:"red"})` (voir comment nous avons utilisé le `.css()` méthode `.css()` après le `.append()` , les deux méthodes s'appliquent à la même collection `.warn`)

Autoriser l'utilisation du même plug-in sur différentes collections en passant par différentes options de personnalisation joue un rôle important dans la **personnalisation / réutilisation**

```
(function($) {
  $.fn.highlight = function( custom ) {

    // Default settings
    var settings = $.extend({
        color : "",              // Default to current text color
        background : "yellow"    // Default to yellow background
    }, custom);

    return this.css({     // `return this` maintains method chainability
        color : settings.color,
        backgroundColor : settings.background
    });

  };
}( jQuery ));


// Use Default settings
$("span").highlight();    // you can chain other methods

// Use Custom settings
$("span").highlight({
    background: "#f00",
    color: "white"
});
```

jsfiddle démo

Liberté

Les exemples ci-dessus concernent la compréhension de la création de plug-ins de base. N'oubliez pas de ne pas restreindre un utilisateur à un ensemble limité d'options de personnalisation.

Disons par exemple que vous voulez construire un `.highlight()` où vous pouvez passer une chaîne de **texte** souhaitée qui sera mise en évidence et autoriser une liberté maximale en ce qui concerne les styles:

```
//...
// Default settings
var settings = $.extend({
   text : "",          // text to highlight
   class : "highlight" // reference to CSS class
}, custom);

return this.each(function() {
   // your word highlighting logic here
});
//...
```

l'utilisateur peut maintenant transmettre un **texte** souhaité et contrôler complètement les styles ajoutés en utilisant une classe CSS personnalisée:

```
$("#content").highlight({
    text : "hello",
```

```
    class : "makeYellowBig"
});
```

jsFiddle exemple

Méthode jQuery.fn.extend ()

Cette méthode étend l'objet prototype jQuery ($ .fn) pour fournir de nouvelles méthodes personnalisées pouvant être chaînées à la fonction jQuery ().

Par exemple:

```
<div>Hello</div>
<div>World!</div>

<script>
jQuery.fn.extend({
  // returns combination of div texts as a result
  getMessage: function() {
    var result;
    // this keyword corresponds result array of jquery selection
    this.each(function() {
        // $(this) corresponds each div element in this loop
        result = result + " " + $(this).val();
    });
    return result;
  }
});

// newly created .getMessage() method
var message = $("div").getMessage();

// message = Hello World!
console.log(message);
</script>
```

Lire Plugins en ligne: https://riptutorial.com/fr/jquery/topic/1805/plugins

Chapitre 15: Pré-ajouter

Examples

Ajouter un élément à un conteneur

Solution 1:

```
$('#parent').prepend($('#child'));
```

Solution 2:

```
$('#child').prependTo($('#parent'));
```

Les deux solutions ajoutent l'élément `#child` (en ajoutant au début) à l'élément `#parent` .

Avant:

```
<div id="parent">
  <span>other content</span>
</div>
<div id="child">

</div>
```

Après:

```
<div id="parent">
  <div id="child">

  </div>
  <span>other content</span>
</div>
```

Méthode Prepend

`prepend()` - Insère le contenu, spécifié par le paramètre, au début de chaque élément de l'ensemble d'éléments correspondants.

1. `prepend( content [, content ] )`

```
// with html string
jQuery('#parent').prepend('<span>child</span>');
// or you can use jQuery object
jQuery('#parent').prepend($('#child'));
// or you can use comma seperated multiple elements to prepend
jQuery('#parent').prepend($('#child1'),$('#child2'));
```

2. `prepend(function)`

`version: 1.4` JQuery *`version: 1.4`* partir de la *`version: 1.4`* , vous pouvez utiliser la fonction de rappel comme argument. Où vous pouvez obtenir des arguments en tant que position d'index de l'élément dans l'ensemble et l'ancienne valeur HTML de l'élément. Dans la fonction, `this` fait référence à l'élément actuel de l'ensemble.

```
jQuery('#parent').prepend(function(i,oldHTML){
     // return the value to be prepend
     return  '<span>child</span>';
});
```

Lire Pré-ajouter en ligne: https://riptutorial.com/fr/jquery/topic/1909/pre-ajouter

Chapitre 16: Sélecteurs

Introduction

Un sélecteur jQuery sélectionne ou recherche un élément DOM (document object model) dans un document HTML. Il est utilisé pour sélectionner des éléments HTML basés sur l'ID, le nom, les types, les attributs, la classe, etc. Il est basé sur les sélecteurs CSS existants.

Syntaxe

- Tag: Aucun marqueur, utilisez le tag directement
- Id: `#id`
- Classe: `.className`
- Attribut: `[attributeName]`
- Attribut avec valeur: `[attributeName ='value']`
- L'attribut commence par la valeur `^=` : `[attributeName ^= 'value']`
- L'attribut se termine par la valeur `$=` : `[attributeName $='value']`
- L'attribut contient la valeur `*=` : `[attributeName *= 'value']`
- Pseudo-sélecteur `:pseudo-selector`
- Tout descendant: espace entre les sélecteurs
- Direct enfants: `>` entre les sélecteurs
- Frère adjacent après le premier: `+`
- Frère non adjacent après le premier: `~`
- Ou: `,` (virgule) entre sélecteur

Remarques

Lors de l'écriture de `selectors` pour la `class` ou l' `id` ou l' `attribute` qui contient des caractères spéciaux comme

```
! " # $ % & ' ( ) * + , . / : ; < = > ? @ [ \ ] ^ { | } ~
```

les caractères doivent être échappés en utilisant deux barres obliques inverses `\\` .

par exemple.

```
<span id="temp.foobar"><span>
```

les sélecteurs seront encadrés comme,

```
$('#temp\\.foobar')
```

Examples

Types de sélecteurs

Dans jQuery, vous pouvez sélectionner des éléments dans une page en utilisant différentes propriétés de l'élément, notamment:

- Type
- Classe
- ID
- Possession d'attribut
- Valeur d'attribut
- Sélecteur indexé
- Pseudo-état

Si vous connaissez les sélecteurs CSS, vous remarquerez que les sélecteurs dans jQuery sont les mêmes (à quelques exceptions près).

Prenez le code HTML suivant, par exemple:

```
<a href="index.html"></a>                  <!-- 1 -->
<a id="second-link"></a>                   <!-- 2 -->
<a class="example"></a>                    <!-- 3 -->
<a class="example" href="about.html"></a>  <!-- 4 -->
<span class="example"></span>              <!-- 5 -->
```

Sélection par type:

Le sélecteur jQuery suivant sélectionnera tous les éléments `<a>` , y compris 1, 2, 3 et 4.

```
$("a")
```

Sélection par classe

Le sélecteur jQuery suivant sélectionnera tous les éléments de l' `example` de classe (y compris les éléments non-a), à savoir 3, 4 et 5.

```
$(".example")
```

Sélection par ID

Le sélecteur jQuery suivant sélectionnera l'élément avec l'ID donné, qui est 2.

```
$("#second-link")
```

Sélection par possession d'attribut

Le sélecteur jQuery suivant sélectionne tous les éléments avec un attribut `href` défini, y compris 1 et 4.

```
$("[href]")
```

Sélection par valeur d'attribut

Le sélecteur jQuery suivant sélectionne tous les éléments pour lesquels l'attribut `href` existe avec la valeur `index.html` , qui est juste 1.

```
$("[href='index.html']")
```

Sélection par position *indexée* (*sélecteur indexé*)

Le sélecteur jQuery suivant sélectionnera seulement 1, le second `<a>` ie. le `second-link` parce que l'index fourni est `1` comme `eq(1)` (notez que l'index commence à `0` donc le second a été sélectionné ici!).

```
$("a:eq(1)")
```

Sélection avec exclusion indexée

Pour exclure un élément en utilisant son index `:not(:eq())`

Ce qui suit sélectionne les éléments `<a>` , sauf celui avec l' `example` classe, qui est 1

```
$("a").not(":eq(0)")
```

Sélection avec exclusion

Pour exclure un élément d'une sélection, utilisez `:not()`

Ce qui suit sélectionne les éléments `<a>` , sauf ceux avec l' `example` classe, qui sont 1 et 2.

```
$("a:not(.example)")
```

Sélection par pseudo-état

Vous pouvez également sélectionner dans jQuery en utilisant des pseudo-états, y compris `:first-child` `:last-child` `:first-of-type` `:last-of-type` , etc.

Le sélecteur jQuery suivant sélectionne uniquement le premier élément `<a>` : numéro 1.

```
$("a:first-of-type")
```

Combiner les sélecteurs jQuery

Vous pouvez également augmenter votre spécificité en combinant plusieurs sélecteurs jQuery; vous pouvez en combiner n'importe quel nombre ou les combiner tous. Vous pouvez également sélectionner plusieurs classes, attributs et états en même temps.

```
$("a.class1.class2.class3#someID[attr1][attr2='something'][attr3='something']:first-of-
type:first-child")
```

Cela sélectionnerait un élément `<a>` qui:

- A les classes suivantes: `class1, class2, and class3`
- A l'identifiant suivant: `someID`
- A l'attribut suivant: `attr1`
- A les attributs et valeurs suivants: `attr2` avec valeur `something` , `attr3` avec valeur `something`
- A les états suivants: `first-child` et `first-of-type`

Vous pouvez également séparer différents sélecteurs par une virgule:

```
$("a, .class1, #someID")
```

Cela sélectionnerait:

- Tous les `<a>` éléments
- Tous les éléments qui ont la classe `class1`
- Un élément avec l'id `#someID`

Sélection des enfants et des frères et sœurs

Les sélecteurs jQuery sont généralement conformes aux mêmes conventions que CSS, ce qui vous permet de sélectionner les enfants et les frères et sœurs de la même manière.

- Pour sélectionner un enfant non direct, utilisez un espace
- Pour sélectionner un enfant direct, utilisez un `>`
- Pour sélectionner un frère adjacent après le premier, utilisez un `+`
- Pour sélectionner un frère non adjacent après le premier, utilisez un `~`

Sélection de caractères génériques

Il peut y avoir des cas où nous voulons sélectionner tous les éléments mais il n'ya pas de propriété commune à sélectionner (classe, attribut, etc.). Dans ce cas, nous pouvons utiliser le sélecteur `*` qui sélectionne simplement tous les éléments:

```
$('#wrapper *')    // Select all elements inside #wrapper element
```

Combinaison de sélecteurs

Envisagez de suivre la structure DOM

```
<ul class="parentUl">
    <li> Level 1
        <ul class="childUl">
            <li>Level 1-1 <span> Item - 1 </span></li>
            <li>Level 1-1 <span> Item - 2 </span></li>
        </ul>
    </li>
    <li> Level 2
```

```
        <ul class="childUl">
            <li>Level 2-1 <span> Item - 1 </span></li>
            <li>Level 2-1 <span> Item - 1 </span></li>
        </ul>
    </li>
</ul>
```

Sélecteurs descendants et enfants

Étant donné un parent `<ul>` - `parentUl` trouve ses descendants (`<li>`),

1. Simple **`$('parent child')`**

   >> `$('ul.parentUl li')`

   Cela obtient tous les descendants de l'ancêtre correspondant indiqué *vers* le *bas tous* les *niveaux.*

2. `>` **- `$('parent > child')`**

   >> `$('ul.parentUl > li')`

   Cela trouve tous les enfants correspondants (*seulement le 1er niveau inférieur*).

3. Sélecteur basé sur le contexte - **`$('child','parent')`**

   >> `$('li','ul.parentUl')`

   Cela fonctionne comme 1. ci-dessus.

4. `find()` **- `$('parent').find('child')`**

   >> `$('ul.parentUl').find('li')`

   Cela fonctionne comme 1. ci-dessus.

5. `children()` **- `$('parent').find('child')`**

   >> `$('ul.parentUl').children('li')`

   Cela fonctionne comme 2. ci-dessus.

Autres combinateurs

Sélecteur de groupe: ","

Sélectionnez tous les éléments `<ul>` ET tous les éléments `<li>` ET tous les éléments `<span>` :

```
$('ul, li, span')
```

Sélecteur multiple: "" (aucun caractère)

Sélectionnez tous les éléments `<ul>` avec la classe `parentUl` :

```
$('ul.parentUl')
```

Sélecteur de frères et sœurs adjacents: "+"

Sélectionnez tous les éléments `<li>` placés immédiatement après un autre élément `<li>` :

```
$('li + li')
```

Sélecteur général de frères et sœurs: "~"

Sélectionnez tous les éléments `<li>` qui sont des frères et soeurs d'autres éléments `<li>` :

```
$('li ~ li')
```

Vue d'ensemble

Les éléments peuvent être sélectionnés par jQuery à l'aide de jQuery Selectors . La fonction renvoie un élément ou une liste d'éléments.

Sélecteurs de base

```
$("*")                      // All elements
$("div")                    // All <div> elements
$(".blue")                  // All elements with class=blue
$(".blue.red")              // All elements with class=blue AND class=red
$(".blue,.red")             // All elements with class=blue OR class=red
$("#headline")              // The (first) element with id=headline
$("[href]")                 // All elements with an href attribute
$("[href='example.com']")   // All elements with href=example.com
```

Opérateurs relationnels

```
$("div span")          // All <span>s that are descendants of a <div>
$("div > span")        // All <span>s that are a direct child of a <div>
$("a ~ span")          // All <span>s that are siblings following an <a>
$("a + span")          // All <span>s that are immediately after an <a>
```

Sélecteurs de mise en cache

Chaque fois que vous utilisez un sélecteur dans jQuery, le DOM est recherché pour les éléments

correspondant à votre requête. Faire cela trop souvent ou à plusieurs reprises diminuera les performances. Si vous faites référence à un sélecteur spécifique plusieurs fois, vous devez l'ajouter au cache en l'affectant à une variable:

```
var nav = $('#navigation');
nav.show();
```

Cela remplacerait:

```
$('#navigation').show();
```

La mise en cache de ce sélecteur peut s'avérer utile si votre site Web doit souvent afficher / masquer cet élément. S'il y a plusieurs éléments avec le même sélecteur, la variable deviendra un tableau de ces éléments:

```
<div class="parent">
    <div class="child">Child 1</div>
    <div class="child">Child 2</div>
</div>

<script>
    var children = $('.child');
    var firstChildText = children[0].text();
    console.log(firstChildText);

    // output: "Child 1"
</script>
```

REMARQUE: L'élément doit exister dans le DOM au moment de son affectation à une variable. S'il n'y a pas d'élément dans le DOM avec une classe appelée `child` vous stockerez un tableau vide dans cette variable.

```
<div class="parent"></div>

<script>
    var parent   = $('.parent');
    var children = $('.child');
    console.log(children);

    // output: []

    parent.append('<div class="child">Child 1</div>');
    children = $('.child');
    console.log(children[0].text());

    // output: "Child 1"
</script>
```

N'oubliez pas de réaffecter le sélecteur à la variable après avoir ajouté / supprimé des éléments dans le DOM avec ce sélecteur.

Note : Lors de la mise en cache des sélecteurs, de nombreux développeurs démarreront le nom de la variable avec un `$` pour indiquer que la variable est un objet jQuery comme ceci:

```
var $nav = $('#navigation');
$nav.show();
```

Eléments DOM comme sélecteurs

jQuery accepte une grande variété de paramètres, dont l'un est un élément DOM réel. Si vous passez un élément DOM à jQuery, la structure de type jQuery sous-jacente sous forme de tableau contiendra cet élément.

jQuery détectera que l'argument est un élément DOM en inspectant son nodeType.

L'utilisation la plus courante d'un élément DOM est dans les rappels, où l'élément actuel est transmis au constructeur jQuery afin d'accéder à l'API jQuery.

Comme dans `each` rappel (note: chacun est une fonction d'itérateur).

```
$(".elements").each(function(){
    //the current element is bound to `this` internally by jQuery when using each
    var currentElement = this;

    //at this point, currentElement (or this) has access to the Native API

    //construct a jQuery object with the currentElement(this)
    var $currentElement = $(this);

    //now $currentElement has access to the jQuery API
});
```

Chaînes HTML en tant que sélecteurs

jQuery accepte une grande variété de paramètres en tant que "sélecteurs", et l'un d'entre eux est une chaîne HTML. Si vous transmettez une chaîne HTML à jQuery, la structure sous-jacente de l' objet jQuery ressemblera à celle du HTML généré.

jQuery utilise regex pour déterminer si la chaîne transmise au constructeur est une chaîne HTML et également qu'elle *doit* commencer par `<` . Cette regex est définie comme étant `rquickExpr = /^(?:\s*(<[\w\W]+>)[^>]*|#([\w-]*))$/` (explication sur regex101.com).

L'utilisation la plus courante d'une chaîne HTML en tant que sélecteur se produit lorsque des ensembles d'éléments DOM doivent être créés dans du code uniquement, ce qui est souvent utilisé par les bibliothèques pour des éléments tels que les fenêtres pop-up modales.

Par exemple, une fonction qui a renvoyé une balise d'ancrage dans un div en tant que modèle

```
function template(href,text){
    return $("<div><a href='" + href + "'>" + text + "</a></div>");
}
```

Renverrait un objet jQuery contenant

```
<div>
```

```
    <a href="google.com">Google</a>
</div>
```

si appelé comme `template("google.com","Google")` .

Lire Sélecteurs en ligne: https://riptutorial.com/fr/jquery/topic/389/selecteurs

Chapitre 17: Traversée DOM

Examples

Sélectionnez les enfants de l'élément

Pour sélectionner les enfants d'un élément, vous pouvez utiliser la méthode `children()` .

```
<div class="parent">
    <h2>A headline</h2>
    <p>Lorem ipsum dolor sit amet...</p>
    <p>Praesent quis dolor turpis...</p>
</div>
```

Changez la couleur de *tous* les enfants de l'élément `.parent` :

```
$('.parent').children().css("color", "green");
```

La méthode accepte un argument de `selector` facultatif pouvant être utilisé pour filtrer les éléments renvoyés.

```
// Only get "p" children
$('.parent').children("p").css("color", "green");
```

Itération sur la liste des éléments jQuery

Lorsque vous devez parcourir la liste des éléments jQuery.

Considérez cette structure DOM:

```
<div class="container">
    <div class="red one">RED 1 Info</div>
    <div class="red two">RED 2 Info</div>
    <div class="red three">RED 3 Info</div>
</div>
```

Pour imprimer le texte présent dans tous les éléments `div` avec une classe de `red` :

```
$(".red").each(function(key, ele){
    var text = $(ele).text();
    console.log(text);
});
```

Astuce: `key` est l'index de l'élément `div.red` nous `div.red` actuellement, dans son parent. `ele` est l'élément HTML, nous pouvons donc créer un objet jQuery à l'aide de `$()` ou de `jQuery()` , comme ceci: `$(ele)` . Après, nous pouvons appeler n'importe quelle méthode jQuery sur l'objet, comme `css()` ou `hide()` etc. Dans cet exemple, nous tirons simplement le texte de l'objet.

Sélection des frères et sœurs

Pour sélectionner des frères et sœurs d'un élément, vous pouvez utiliser la méthode `.siblings()` .

Un exemple typique où vous voulez modifier les frères et sœurs d'un élément est dans un menu:

```
<ul class="menu">
    <li class="selected">Home</li>
    <li>Blog</li>
    <li>About</li>
</ul>
```

Lorsque l'utilisateur clique sur un élément de menu, la classe `selected` doit être ajoutée à l'élément cliqué et supprimée de ses *frères et sœurs* :

```
$(".menu").on("click", "li", function () {
    $(this).addClass("selected");
    $(this).siblings().removeClass("selected");
});
```

La méthode prend un argument de `selector` facultatif, qui peut être utilisé si vous devez limiter les types de frères et sœurs que vous souhaitez sélectionner:

```
$(this).siblings("li").removeClass("selected");
```

Méthode la plus proche ()

Renvoie le premier élément qui correspond au sélecteur en commençant à l'élément et en parcourant l'arborescence DOM.

HTML

```
<div id="abc" class="row">
    <div id="xyz" class="row">
    </div>
    <p id="origin">
      Hello
    </p>
</div>
```

jQuery

```
var target = $('#origin').closest('.row');
console.log("Closest row:", target.attr('id') );

var target2 = $('#origin').closest('p');
console.log("Closest p:", target2.attr('id') );
```

SORTIE

```
"Closest row: abc"
```

```
"Closest p: origin"
```

Méthode first (): la première méthode renvoie le premier élément de l'ensemble correspondant.

HTML

```
<div class='.firstExample'>
   <p>This is first paragraph in a div.</p>
   <p>This is second paragraph in a div.</p>
   <p>This is third paragraph in a div.</p>
   <p>This is fourth paragraph in a div.</p>
   <p>This is fifth paragraph in a div.</p>
</div>
```

JQuery

```
var firstParagraph = $("div p").first();
console.log("First paragraph:", firstParagraph.text());
```

Sortie:

```
First paragraph: This is first paragraph in a div.
```

Obtenir l'élément suivant

Pour obtenir l'élément suivant, vous pouvez utiliser la méthode `.next()` .

```
<ul>
    <li>Mark</li>
    <li class="anna">Anna</li>
    <li>Paul</li>
</ul>
```

Si vous êtes sur l'élément "Anna" et que vous voulez obtenir l'élément suivant, "Paul", la méthode `.next()` vous permettra de le faire.

```
// "Paul" now has green text
$(".anna").next().css("color", "green");
```

La méthode prend un argument de `selector` facultatif, qui peut être utilisé si l'élément suivant doit être un certain type d'élément.

```
// Next element is a "li", "Paul" now has green text
$(".anna").next("li").css("color", "green");
```

Si l'élément suivant ne fait pas partie du `selector` type `selector` un ensemble vide est renvoyé et les modifications ne feront rien.

```
// Next element is not a ".mark", nothing will be done in this case
$(".anna").next(".mark").css("color", "green");
```

Obtenir l'élément précédent

Pour obtenir l'élément précédent, vous pouvez utiliser la méthode `.prev()` .

```
<ul>
    <li>Mark</li>
    <li class="anna">Anna</li>
    <li>Paul</li>
</ul>
```

Si vous êtes sur l'élément "Anna" et que vous voulez obtenir l'élément précédent, "Mark", la méthode `.prev()` vous permettra de le faire.

```
// "Mark" now has green text
$(".anna").prev().css("color", "green");
```

La méthode prend un argument de `selector` facultatif, qui peut être utilisé si l'élément précédent doit être un certain type d'élément.

```
// Previous element is a "li", "Mark" now has green text
$(".anna").prev("li").css("color", "green");
```

Si l'élément précédent ne fait pas partie du `selector` type `selector` un ensemble vide est renvoyé et les modifications ne feront rien.

```
// Previous element is not a ".paul", nothing will be done in this case
$(".anna").prev(".paul").css("color", "green");
```

Filtrer une sélection

Pour filtrer une sélection, vous pouvez utiliser la méthode `.filter()` .

La méthode est appelée sur une sélection et renvoie une nouvelle sélection. Si le filtre correspond à un élément, il est ajouté à la sélection renvoyée, sinon il est ignoré. Si aucun élément n'est mis en correspondance, une sélection vide est renvoyée.

Le HTML

C'est le HTML que nous allons utiliser.

```
<ul>
    <li class="zero">Zero</li>
    <li class="one">One</li>
    <li class="two">Two</li>
    <li class="three">Three</li>
</ul>
```

Sélecteur

Le filtrage à l'aide de sélecteurs est l'un des moyens les plus simples de filtrer une sélection.

```
$("li").filter(":even").css("color", "green"); // Color even elements green
$("li").filter(".one").css("font-weight", "bold"); // Make ".one" bold
```

Fonction

Filtrer une sélection à l'aide d'une fonction est utile s'il n'est pas possible d'utiliser des sélecteurs.

La fonction est appelée pour chaque élément de la sélection. S'il renvoie une valeur `true` , l'élément sera ajouté à la sélection renvoyée.

```
var selection = $("li").filter(function (index, element) {
    // "index" is the position of the element
    // "element" is the same as "this"
    return $(this).hasClass("two");
});
selection.css("color", "green"); // ".two" will be colored green
```

Éléments

Vous pouvez filtrer par éléments DOM. Si les éléments DOM sont dans la sélection, ils seront inclus dans la sélection renvoyée.

```
var three = document.getElementsByClassName("three");
$("li").filter(three).css("color", "green");
```

Sélection

Vous pouvez également filtrer une sélection par une autre sélection. Si un élément figure dans les deux sélections, il sera inclus dans la sélection renvoyée.

```
var elems = $(".one, .three");
$("li").filter(elems).css("color", "green");
```

méthode find ()

La méthode .find () nous permet de rechercher parmi les descendants de ces éléments dans l'arborescence DOM et de construire un nouvel objet jQuery à partir des éléments correspondants.

HTML

```
     <div class="parent">
        <div class="children" name="first">
            <ul>
                <li>A1</li>
                <li>A2</li>
```

```
                <li>A3</li>
            </ul>
        </div>
        <div class="children" name="second">
            <ul>
                <li>B1</li>
                <li>B2</li>
                <li>B3</li>
            </ul>
        </div>
       </div>
```

jQuery

```
$('.parent').find('.children[name="second"] ul li').css('font-weight','bold');
```

Sortie

- A1
- A2
- A3

- **B1**
- **B2**
- **B3**

Lire Traversée DOM en ligne: https://riptutorial.com/fr/jquery/topic/1189/traversee-dom

Chapitre 18: Visibilité de l'élément

Paramètres

Paramètre	Détails
Durée	Une fois passé, les effets de `.hide()` , `.show()` et `.toggle()` sont animés; les éléments vont progressivement disparaître.

Examples

Vue d'ensemble

```
$(element).hide()             // sets display: none
$(element).show()             // sets display to original value
$(element).toggle()           // toggles between the two
$(element).is(':visible')     // returns true or false
$('element:visible')          // matches all elements that are visible
$('element:hidden')           // matches all elements that are hidden

$('element').fadeIn();           // display the element
$('element').fadeOut();           // hide the element

$('element').fadeIn(1000);           // display the element using timer
$('element').fadeOut(1000);           // hide the element using timer

// display the element using timer and a callback function
$('element').fadeIn(1000, function(){
 // code to execute
});

// hide the element using timer and a callback function
$('element').fadeOut(1000, function(){
   // code to execute
});
```

Possibilité de basculer

Cas de `toggle()` simple `toggle()`

```
function toggleBasic() {
  $(".target1").toggle();
}
```

Avec *durée* spécifique

```
function toggleDuration() {
  $(".target2").toggle("slow"); // A millisecond duration value is also acceptable
}
```

... et *rappel*

```
function toggleCallback() {
  $(".target3").toggle("slow",function(){alert('now do something');});
}
```

... ou avec *assouplissement* et rappel.

```
function toggleEasingAndCallback() {
  // You may use jQueryUI as the core only supports linear and swing easings
  $(".target4").toggle("slow","linear",function(){alert('now do something');});
}
```

... ou avec une variété d' *options* .

```
function toggleWithOptions() {
  $(".target5").toggle(
    { // See all possible options in: api.jquery.com/toggle/#toggle-options
      duration:1000, // milliseconds
      easing:"linear",
      done:function(){
        alert('now do something');
      }
    }
  );
}
```

Il est également possible d'utiliser une *diapositive* comme animation avec `slideToggle()`

```
function toggleSlide() {
  $(".target6").slideToggle(); // Animates from top to bottom, instead of top corner
}
```

... ou fondu en changeant d'opacité avec `fadeToggle()`

```
function toggleFading() {
  $( ".target7" ).fadeToggle("slow")
}
```

... ou basculer une classe avec `toggleClass()`

```
function toggleClass() {
  $(".target8").toggleClass('active');
}
```

Un cas courant est d'utiliser `toggle()` pour afficher un élément en cachant l'autre (même classe)

```
function toggleX() {
  $(".targetX").toggle("slow");
}
```

Tous les exemples ci-dessus peuvent être trouvés ici

Lire Visibilité de l'élément en ligne: https://riptutorial.com/fr/jquery/topic/1298/visibilite-de-l-element

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec jQuery	A.J, acdcjunior, amflare, Anil, bwegs, Community, DGS, empiric, Fueled By Coffee, hairboat, Hirshy, Iceman, Igor Raush, J F, jkdev, John C, Kevin B, Kevin Katzke, Kevin Montrose, Luca Putzu, Matas Vaitkevicius, mico, Mottie, Neal, ni8mr, Prateek, RamenChef, Rion Williams, Roko C. Buljan, secelite, Shaunak D, Stephen Leppik, Suganya, Sverri M. Olsen, the12, Travis J, user2314737, Velocibadgery, Yosvel Quintero
2	Ajax	Alon Eitan, amflare, Andrew Brooke, Ashkan Mobayen Khiabani, Athafoud, atilacamurca, Ben H, Cass, Community, csbarnes, Dr. J. Testington, Edathadan Chief aka Arun, empiric, hasan, joe_young, John C, kapantzak, Kiren Siva, Lacrioque, Marimba, Nirav Joshi, Ozan, shaN, Shaunak D, Teo Dragovic, Yosvel Quintero
3	Ajouter	Ashkan Mobayen Khiabani, bipon, Community, Darshak, Deryck, empiric, Flyer53, J F, JF it, Paul Roub, Pranav C Balan, Proto, Shaunak D
4	Case à cocher Tout sélectionner avec cocher / décocher automatiquement les autres cases à cocher	user1851673
5	Chaque fonction	bipon, Renier
6	événement prêt pour le document	Adjit, Alon Eitan, amflare, charlietfl, Emanuel Vintilă, Igor Raush, J F, jkdev, Joram van den Boezem, Liam, Mark Schultheiss, Melanie, Nhan, Nico Westerdale, Scimonster, secelite, TheDeadMedic, the-noob, URoy
7	Événements	Adjit, charlietfl, DelightedD0D, doydoy44, empiric, Gone Coding, Horst Jahns, Jatniel Prinsloo, Kevin B, Louis, Luca Putzu, Marimba, NotJustin, SGS Venkatesh, Stephen Leppik, Sunny R Gupta, Washington Guedes, WMios, Zakaria Acharki
8	jQuery Objets différés et promesses	Alex, Ashiquzzaman, Gone Coding

9	Les attributs	A.J, acdcjunior, ban17, ochi, Scimonster
10	Manipulation CSS	abaracedo, Ashkan Mobayen Khiabani, Brandt Solovij, J F, j08691, Jonathan Michalik, Kevin B, Petroff, Roko C. Buljan, ScientiaEtVeritas, Shlomi Haver, Sorangwala Abbasali, Stephen Leppik, Sverri M. Olsen
11	Manipulation DOM	Angelos Chalaris, Assimilater, Brock Davis, DawnPaladin, DefyGravity, Deryck, Marimba, Mark Schultheiss, martincarlin87 , Neal, Paul Roub, Shaunak D, still_learning
12	Méthode jQuery .animate ()	RamenChef, Rust in Peace, Simplans, VJS
13	Obtenir et régler la largeur et la hauteur d'un élément	Ashkan Mobayen Khiabani
14	Plugins	hasan, Roko C. Buljan, Travis J
15	Pré-ajouter	Ashkan Mobayen Khiabani, empiric, J F, Pranav C Balan
16	Sélecteurs	alepeino, Alon Eitan, Brock Davis, Castro Roy, David, DelightedD0D, devlin carnate, dlsso, hasan, Iceman, James Donnelly, JLF, John Slegers, kapantzak, Kevin B, Keyslinger, Matas Vaitkevicius, Melanie, MikeC, Nux, Petr R., rbashish, Scimonster, Shaunak D, Shekhar Pankaj, Sorangwala Abbasali, ssb, Sverri M. Olsen, Travis J, whales, WOUNDEDStevenJones , Zaz
17	Traversée DOM	A.J, charlietfl, Community, dlsso, mark.hch, rmondesilva, SGS Venkatesh, sucil, Sverri M. Olsen, The_Outsider, Zaz
18	Visibilité de l'élément	Alex Char, Paul Roub, Rupali Pemare, The_Outsider, Theodore K., user2314737, Zaz