無料電子ブック

学習

jQuery

Free unaffiliated eBook created from
Stack Overflow contributors.

#jquery

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

1: jQueryをいめる

jQueryは、DOM、イベント、AJAX、およびアニメーションをするJavaScriptライブラリです。また、となるDOMやJavaScriptエンジンでくのブラウザのをします。

jQueryのバージョンは、されたされたとされていないのでhttps://code.jquery.com/jquery/からダウンロードできます。

バージョン

バージョン	ノート	
1.0	の	2006-08-26
1.1		2007-01-14
1.2		2007-09-10
1.3	シズルがコアにされました	2009-01-14
1.4		2010-01-14
1.5	コールバック、Ajaxモジュールのきえ	2011-01-31
1.6	`attr()` `val()`メソッドと`val()`メソッドでなパフォーマンス	2011-05-03
1.7	しいイベントAPI `on()`と`off()`。	2011-11-03
1.8	シズル、きされたアニメーション、 `$(html, props)`。	2012-08-09
1.9	のインターフェイスのとコードクリーンアップ	2013-01-15
1.10	1.9と2.0ののベータサイクルからされたバグとのみみ	2013-05-24
1.11		2014-01-24
1.12		201618
2.0	パフォーマンスのとサイズののためのIE 6-8サポートの	2013418
2.1		2014-01-24
2.2		201618
3.0	いくつかのjQueryカスタムセレクタのな	2016-06-09
3.1	サイレントエラーはしません	2016-07-07

Examples

jQuery ネームスペース "jQuery" と "$"

`jQuery`は、あらゆるjQueryコードをくためのです。これは、`jQuery(...)`または`jQuery.foo`としてできます。

`$`は`jQuery`のエイリアスであり、はにできます `jQuery.noConflict();`がされているをく - 「 のの 」を。

このHTMLスニペットがあるとすると、

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

このdivにテキストコンテンツをするためにjQueryをすることができます。これをうには、jQueryの`text()`をします。これは`jQuery`か`$`をってくことができます。すなわち、

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

または -

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

ともじなHTMLをもたらすでしょう -

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

`$`は`jQuery`よりもであるため、にjQueryコードをくのにましいです。

jQueryはCSSセレクタをし、のではIDセレクタがされていました。 jQuery のセレクタのについては、セレクタのタイプをしてください。

ののの`hello.html`ファイルをします。

```
<!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>
```

JSBinのライブデモ

このファイルをWebブラウザできます。そのHello, World!というテキストのページがされます
Hello, World!

コードの

1. jQueryのCDNからjQueryライブラリをロードします。

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

   これは$グローバル、 jQueryとのエイリアスをします。

   > *jQuery*をインクルードするによくあるいの1つは、それにするかするのスクリプトやライブラリよりもにライブラリをロードすることにしていることにしてください。

2. jQueryによってDOM Document Object Model が「」であるとされたときにされるをディファレンスします。

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

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

3. DOMのができたら、jQueryはのコールバックをします。たちののには、2つのなことをするびしが1つしかありません

   1. idがhello セレクターの #hello にしいをします。されたとしてセレクタをすることは、jQueryのとのです。にdocument.querySelectorAll MDNのからしました。

   2. したのtext()をHello, World!しtext() Hello, World!。

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

については、 jQuery - Documentationページをしてください。

HTMLページのにスクリプトタグをめる

のCDNからjQueryをロードするには、jQueryのWebサイトにアクセスしてください 。できるさまざまなバージョンとフォーマットのリストがされます。

、ロードしたいjQueryのバージョンのソースをコピーします。たとえば、 **jQuery 2.X**をロードする、 またはタグをクリックすると、のようにされます。

フルコードをコピーまたはコピーアイコンをクリックし、htmlの`<head>`または`<body>`にりけます。

ベストプラクティスは、 `async`をつheadタグにJavaScriptライブラリをロードすることです。ここにデモがあります

```
<!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>
```

`async`をするは、JavaScriptライブラリがにロードされ、なりすぐにされるので、してください。2つのライブラリがのライブラリにする2つのライブラリがまれているは、2のライブラリがロードされてのライブラリのにされると、エラーがし、アプリケーションがれるがあります。

のをける

jQueryのライブラリでも`$`をエイリアスとしてできます。これにより、これらのライブラリとjQueryとのでがするがあります。

のライブラリでするために`$`をリリースするには

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

このをびした、 `$`は`jQuery`エイリアスではなくなりました。ただし、`jQuery`をしてjQueryにアクセスすることはできます。

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

にじて、のをjQueryのとしてりてることができます。

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

に、のライブラリがjQueryをしないようにするには 、 ちにびされるIIFEに jQueryコードをラップし、として`jQuery`をします。

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

このIIFEのでは、 `$`はjQueryのエイリアスです。

jQueryの**$**エイリアスをし、DOMがであることをするもうつのな

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

する、

- `jQuery.noConflict()` `$`はもはやjQueryをするのではなく、`jQuery`がします。
- `var jQuery2 = jQuery.noConflict()` - `$`はもはやjQueryをしませんが、`jQuery`は`jQuery2`ます。

、3のシナリオがします**`jQuery2`**のみできるようにしたいはどうなりますかつかいます、

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

これは、 `jQuery`をする`$`と`jQuery`どちらもしません。

これは、のバージョンのjQueryをじページにロードするにです。

```
<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/

jQueryをっていないページにコンソールでjQueryをロードしています。

`jQuery`をしていないページでするがあるがありますが、ほとんどのは`jQuery`にしています。

このようなでは、 `Chrome Developer Tools`コンソール `F12` をして、の`jQuery`をして、ロードされたページに`jQuery`をでできます。

```
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);
```

なバージョンが `1.12.4` となる、 ここになバージョンのリンクがされます 。

jQueryオブジェクト

jQueryがびされるたびに、 `$()`または`jQuery()` 、に`jQuery new`インスタンスがされます。これはしいインスタンスをすソースコードです

```
// 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 );
}
```

にjQueryはそのプロトタイプを.fnとしてしており、jQueryオブジェクトをにインスタンスするためにここでされているスタイルは、びしがnewにしなくてもそのプロトタイプをできるようにします。

インスタンスjQuery API .each、 children、 filterなどがどのようにされているかにえて、にjQueryはセレクタのとするのようなをしますもされていないもの、 undefined、 null、またはのものがとしてされました。のアイテムの、こののようなはそのアイテムだけをします。

なデモンストレーションは、idをつをつけて、jQueryオブジェクトにアクセスしてとなるDOMをすことですこれは、のがまたはするにもします。

```
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
```

のjQueryプラグインのみみ

、プラグインをみむときは、ずjQueryのにプラグインをめてください。

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

のバージョンのjQueryをするがあるは、なバージョンのjQueryのにプラグインをロードしてから、 jQuery.noConflict(true)をするコードをjQuery.noConflict(true)。 のバージョンのjQueryとそれにするプラグインをロードします。

```
<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>
```

プラグインをするときは、するjQueryバージョンをするがあります

```
<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>
```

のプラグインをするために1つのドキュメントしかできませんが、ドキュメントのなjQueryコードのやをけるために、をけておくがいでしょう。

オンラインでjQueryをいめるをむ https://riptutorial.com/ja/jquery/topic/211/jqueryをいめる

2: Ajax

- var jqXHR = $ .ajaxurl [、settings]
- var jqXHR = $ .ajax[settings]
- jqXHR.donefunctiondata、textStatus、jqXHR{};
- jqXHR.failfunctionjqXHR、textStatus、errorThrown{};
- jqXHR.alwaysfunctionjqXHR{};

パラ メ—タ—

パラ メ—タ	
URL	リクエストをするURLをします。
	のにするのをむオブジェクト
タイプ	にされるHTTPメソッド
データ	によってされるデータ
	がしたにびされるコ—ルバック
エラ—	エラ—をするためのコ—ルバック
statusCode	にするコ—ドがあるにびされるHTTPコ—ドとのオブジェクト
データ·タイプ	サ—バ—からってくるデ—タの
contentType	サ—バ—にするデ—タのコンテンツ·タイプ。デフォルトは "application / x-www-form-urlencoded; charset = UTF-8"です。
コンテキスト	コ—ルバックでされるコンテキストをします。`this`これはのタ—ゲットをします。

AJAXは、**J** avaScript ND **X** MLのです。 AJAXをすると、ペ—ジをリロ—ドせずに、Webペ—ジでサ—バ—へのHTTPAJAXをしてをけることができます。

Examples

$ .ajaxによる**HTTP**レスポンスコ—ドの

がしたかどうかについてトリガされる`.done` 、 `.fail` 、および`.always` promiseコ—ルバックにえて

、のHTTPステータスコードがサーバーからされたときにをトリガするオプションがあります。これは、 `statusCode`パラメーターをしてうことができます。

```
$.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')
});
```

jQueryのドキュメントではのようにべています。

> がすると、ステータスコードはコールバックとじパラメータをります。エラー3xxリダイレクトをむがした、 `error`コールバックとじパラメータがされます。

Ajaxをしてフォームをする

によってはフォームがあり、ajaxをしてフォームをしたいことがあります。

このながあるとします。

```
<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>
```

のjQueryコードをすることができます `$(document).ready`びし -

```
$('#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
      }
  });
});
```

- `var $form = $(this)` - のためにキャッシュされたフォーム
- `$('#ajax_form').submit(function(event){` - ID "ajax_form"のフォームがされると、このをし、そのイベントをパラメータとしてします。
- `event.preventDefault();` - フォームがにされないようにするあるいは、じをもたらす `ajax({});`ステートメントのに`return false`を`return false`こともできます
- `url: $form.attr('action'),` - フォームの "action"のをし、それを "url"プロパティにします。
- `data: $form.serialize(),` - フォームのを、サーバーへのにしたにします。この、"name=Bob&email=bob@bobsemailaddress.com"のようなものがされます。

JSONデータの

jQueryはjSON レスポンスのにしますが、リクエストでJSONでデータをしたいは、もうしがです。

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

するデータにしい`contentType`をしていることをしてください。これはないであり、あなたがしているAPIがするかもしれませんが、 `contentType`が`%20`から`+`へのデフォルトのをわないようにjQueryにするというもあり`contentType application/x-www-form-urlencoded`のデフォルトのままにします。なんらかのでcontentTypeをデフォルトのままにしなければならないは、 `processData`をfalseにして、これを`processData`てください。

`JSON.stringify`へのびしはここではけることができますが、これをすると、JavaScriptオブジェクトのでデータをできますしたがって、プロパティのにするなどのJSONエラーがすることはありません。

オールインワンの

Ajax Get

1

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

2

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

Ajax Loadシンプルさのためにされたのajax getメソッド

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

.loadはのデ―タでびすこともできます。デ―タは、またはオブジェクトとしてすることができます。

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

コ―ルバックメソッドで.loadがびされた、サ―バ―へのリクエストはポストになります

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

Ajax

1

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

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 JSONをする

1

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

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コールまたはリクエストをする

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

//リクエストをする

```
xhr.abort()
```

Ajaxファイルのアップロード

1.なな

このサンプルコードをして、しいファイルがわれるたびにユーザーがしたファイルをアップロードできます。

```
<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
        }
    });
});
```

はこれをし、にべてみましょう。

2.ファイルの

このMDNドキュメントWebアプリケーションからのファイルのは、ファイルのにするさまざまなについてよくんでいます。これらのメソッドのいくつかは、このでもされます。

ファイルをアップロードするに、アップロードするファイルをするをユーザーにするがあります

。このののために、 `file input`をし`file input`。 `multiple`プロパティで`multiple`ファイルをできます。ユーザーがに1つのファイルをするようにするには、のファイルをできます。

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

の`change event`をしてファイルをキャプチャします。

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

ハンドラのでは、ファイルのプロパティをしてファイルにアクセスします。これはオブジェクトのようなであるFileListをえます。

3. FormDataのとめみ

AjaxでファイルをアップロードするためにFormDataをします。

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

のステップでしたFileListはオブジェクトのようなで、forループ、for ... forループ、jQuery.eachなどのさまざまなメソッドをしてできます。このでは、jQueryをします。

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

FormDataのappendメソッドをして、フォームデータにファイルをします。

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

じでするのデータもできます。ユーザーからけったをファイルとともにしたいとします。これをフォームデータオブジェクトにすることができます。

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

4. Ajaxでファイルをする

```
$.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
    }
});
```

`processData`プロパティと`contentType`プロパティを`false`し`false`。これは、ファイルがサーバーにされ、サーバーによってしくされるようにわれます。

オンラインでAjaxをむ https://riptutorial.com/ja/jquery/topic/316/ajax

3: CSS

- .csscssProperty//レンダリングされたCSSプロパティをする
- .css[cssProperty、...]// cssPropertiesのからをする
- .csscssProperty、value//をする
- .css{cssPropertyvalue、...}//プロパティとをする
- .csscssProperty、function// cssPropertyをコールバックにする

レンダリングされた

なユニット `"auto"` 、 `"%"` 、 `"vw"`などをする、 `.css()`はのレンダリングされたを`px`します

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

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

プロパティとの

プロパティは、 の**CSS**フォーマットを**String**としてするか、 **camelCase**をしてできます

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

はでするがあります。はjQueryによってに`px`としてわれます。

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

jQuery 3 `.show()` 、 `.hide()`と`.hide()`をけました。

このjQuery Blogのによれば、オーバーヘッドやパフォーマンスののために、 `.show()`や`.hide()`をするはありません。

> スタイルシートに`display: none`されたがある、 `.show()`メソッドはそれをオーバーライドしなくなります。したがって、jQuery 3.0にするためのもなルールは、のとおりです。スタイルシートをして`display: none`デフォルトをし、 `.show()`または`.slideDown()`などのをすメソッドをしようとし`display: none`でください。 `.fadeIn()` - にします。 をデフォルトでにするがあるは、に「し」のようなクラスをし、そのクラスを`display: none`にスタイルシートですることをおめします。に、jQueryの`.addClass()`および`.removeClass()`メソッドをして、そのクラスをまたはして、をできます。のとして、あなたがつことができます`.ready()`ハンドラのコール`.hide()`それらはページにされるに、のを。または、にスタイルシートのデフォルトをするがあるは、 `.css("display", "block")`

またはなをしてスタイルシートをオーバーライドできます。

Examples

CSSプロパティをする

1つのスタイルのみをする

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

のスタイルをにする

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

CSSプロパティをする

のCSSプロパティをするには、 `.css(propertyName)` メソッドをできます。

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

インクリメント/デクリメントプロパティ

CSSプロパティは、それぞれ `.css()` メソッドをして、 `+=` および `-=` でまたはすることができます。

```
  // 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 - ゲッターとセッター

CSSゲッター

`.css()` **getter**は、のようにページのすべてのDOMにできます。

```
// 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");
```

このは、されたのされたをします。カッコでしたCSSプロパティは、この`$("selector")` DOMのプロパティのをします。しないCSSとして`undefined`なります。

また、ので**CSS**ゲッター—をびすこともできます。

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

そのをつすべてののオブジェクトをします。

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

CSSセッター—

`.css()` **setter**メソッドは、ページのすべてのDOMにすることもできます。

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

このステートメントは、 `$("selector")`の`width`を`500px`し、されたセレクター—にさらにくのメソッドをできるようにjQueryオブジェクトをします。

`.css()` セッター—はCSSプロパティとのObjectをすのにもうことができます

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

がったすべてのは、DOM`style`プロパティにされ、のスタイルにをえますただし、スタイルプロパティはにスタイルののので`!important`です。

オンラインでCSSをむ https://riptutorial.com/ja/jquery/topic/2732/css

4: DOMトラバース

Examples

のを

のをするには、 `children()` メソッドをします。

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

`.parent`のすべてののをします。

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

このメソッドは、されるをフィルタリングするためにできるオプションの`selector`をけります。

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

jQueryのリストをする

jQueryのリストをするがあるとき。

このDOMをえてみましょう

```
<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>
```

すべての`div`にするテキストを`red`クラスでするには

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

ヒント `key`は、のでの`div.red`のインデックスです。 `ele`はHTMLですので、 `$()`や`jQuery()`をって`$(ele)`ようにjQueryオブジェクトをすることができます。その、 `css()`や`hide()`などのオブジェクトのjQueryメソッドをびすことができます。このでは、オブジェクトのテキストをします。

の

アイテムのをするには、 `.siblings()` メソッドをできます。

アイテムのをするなはメニュ―にあります

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

ユ―ザ―がメニュ―をクリックすると、 `selected` クラスがクリックされたにされ、そのからされます。

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

このメソッドはオプションの `selector` をります。これは、したいのをりむがあるにできます。

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

closest メソッド

からまり、DOMツリ―をトラバ―スするセレクタにするのをします。

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') );
```

```
"Closest row: abc"
"Closest p: origin"
```

first メソッドのメソッドは、したのセットからのをします。

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());
```

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

のをる

のをするには、 `.next()` メソッドをします。

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

"Anna"のにっていて、の "Paul"をしたいは、 `.next()` メソッドをします。

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

このメソッドはオプションの`selector`をります。これは、のがののでなければならないにできます。

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

のが`selector`ない、のセットがされ、はもいません。

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

のをする

のをするには、 `.prev()` メソッドをします。

```
<ul>
    <li>Mark</li>
```

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

"Anna"にっていて、の "Mark"をしたいは、 `.prev()`メソッドをします。

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

このメソッドはオプションの`selector`をります。これは、のがののでなければならないにできます。

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

のが`selector`ない、のセットがされ、はもいません。

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

をフィルタリングする

をフィルタリングするには、 `.filter()`メソッドをできます。

このメソッドはでびされ、しいをします。フィルタがとする、それはされたにされます。そうでないはされます。がしない、のがされます。

HTML

これがたちがするHTMLです。

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

セレクタ

セレクタをしたフィルタリングは、をフィルタリングするなの1つです。

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

セレクタをすることができないは、 をしてをフィルタリングするとです。

このはのにしてびされます。 `true`をす、はされたにされます。

```
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
```

DOMでフィルタリングできます。 DOMがにある、DOMはされたにまれます。

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

のでをフィルタリングすることもできます。がのにある、それはされたにまれます。

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

findメソッド

.findメソッドをすると、DOMツリーのこれらののをし、するからしいjQueryオブジェクトをできます。

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');
```

- A1
- A2
- A3

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

オンラインでDOMトラバースをむ https://riptutorial.com/ja/jquery/topic/1189/domトラバース

5: DOM

Examples

DOMの

jQueryは$エイリアスは、のとしいののにできます。

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

オプションで、のをつ2のをすことができます。

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

'<a>' - >のは、するDOMのタイプをします。このでは、 アンカーですが、このリストのかになるがありますす 。 aのリファレンスについては、 をしてください。

{ 'href': 'http://stackexchange.com' } - > 2のは、とのペアをむJavaScriptオブジェクトです。

ののに 'name' 'value'のペアがされます <a name:value>このでは

クラスの

ページにのようなHTMLがまれているとします。

```
<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はDOMクラス、にhasClass()、 addClass()、 removeClass()およびtoggleClass()をするのになをします。これらのは、したのclassをします。

```
$('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');
```

クラスをりえる

サンプルのマークアップをえられ.toggleClass()、の.toggleClass()クラスをできます

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

これはtrueをしtrue $(".small-paragraph").hasClass("pretty")

toggleClassはのようにコードをらしてじをします

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

2つのクラスをりえる

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

クラスを/するブール

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

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

クラストグルのをけるためにさらにのを

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

でされます

```
// 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
```

インデックスをして、クラス/をりえます。

```
$( "div.gridrow" ).toggleClass(function(index,oldClasses, false), showClass ) {
  showClass
  if ( isOdd(index) ) {
    return "wet";
  } else {
    return "dry";
  }
```

```
});
```

よりな**toggleClass**、なグリッドマ―クアップ

```
<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>
```

たちののためのシンプルな

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

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

`gridrow`クラスをつに/クラスをする

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

に`gridfooter`クラスがあるは、/クラスをし、りのはします。

```
$("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);
```

されるクラスは、がわれるかです。ここで、に`gridfooter`がないは、/のクラスをします。このは、OLDクラスリストのりをしています。これ`else return oldClasses;`は`else return oldClasses;`しいクラスだけがされるので、 `gridfooter`クラスをつは、 `gridfooter`クラスをしていないはすべてのクラスをします。そうでないは、トグルされています。

```
$("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);
```

そのAPIメソッド

jQueryは、DOMにできるさまざまなメソッドをします。

は.emptyメソッドです。

のマ―クアップをしてみてください。

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

`$('#content').empty();`びし`$('#content').empty();`のdivがされます。 これは`$('#content').html('');`をってもでき`$('#content').html('');`。

もう1つのなは、 .closestです。

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

セルの1つのでクリックされたボタンにもいをしたいは、これをうことができます

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

おそらく、それぞれの`delete`ボタンをつのがあるため、にクリックしたボタンのをするために.clickで`$(this)`をします。

あなたがクリックした`Delete`ボタンをむの`id`をしたいは、のようにすることができます

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

、jQueryオブジェクトをむのに`$`ドルをけると、のをにすることをおめします。

`.closest()`のわりに`.closest()` メソッドをすることもできます。

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

.parentもあります

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

`.parent()`はDOMツリーの1つのレベルにしか`.parent()`ないため、にがありません。たとえば、ボタンを`span`に`.parent()`ようにすると、jQueryセレクタがれてしまいます。

.html

このメソッドをすると、セレクタのすべてのHTMLをきえることができます。このようなhtmlがあるとします

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

`.html()`うことができます。アラートやテキストをしてし、1ですべてのユーザーにします。

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

のべえ

にをべえるにはすべてのをにのDOMでべえるには、

1. をつける
2. されたについてソート
3. DOMにってする

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

1. それらをつける - `.children()`または`.find()`

   これは、するArrayのようなオブジェクトをします。

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

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

2. それらをする - `Array.prototype.sort()`

   これは、HTMLコンテンツについてをですようにされています。

```
/**
 * 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. それらをする`.append()`

   にをデタッチするはないことにしてください。 - *`append()`*は*DOM*ににするをするので、なコピーはありません

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

それをかわいくする

ソートボタンをする

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

ソートのをする

```
var ascending = true;
```

DOMのをにえるために、DOMと`sortList()`をここにキャッシュします。

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

すべてを`doSort()`でラップします。

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

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

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

`$('#btn-sort')`クリックハンドラをする

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

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

に

```
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;
});
```

ボーナス

マルチレベルソートソートされたをグループする

```
// ...

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);
};

// ...
```

もうそれをることができますか

なグループソートをりえるのボタンをする

MDN - Array.prototype.sort

オンラインでDOMをむ https://riptutorial.com/ja/jquery/topic/512/dom

6: jQuery .animate メソッド

1. セレクタ.animate{styles}、{options}

パラ メーター

パラ メータ	
プロパティ	アニメーションがかうCSSのプロパティとのオブジェクト
	デフォルト400アニメーションのをするまたは
イージング	デフォルトswingトランジションにどのイージングをするかをす
コンプリート	アニメーションがしたらびす。したごとに1びされます。
	アニメーションのにされるをします。
ステップ	アニメーションのステップでされるをします。
キュー	エフェクトキューにアニメーションをするかどうかをするブール。
	アニメーションのステップのにされるをします。
	アニメーションがしたときにされるをします。
します	アニメーションがしなかったにされるをします。
スペシャル.エイジング	スタイルパラメータからの1つまたはのCSSプロパティのマップ、およびするイージング。
に	アニメーションがせずにしたにされるをします。

Examples

コールバックきアニメーション

、のをしたり、のサイズをしたり、のをにして、ウェブサイトやウェブアプリのをさせるがあります。 JQueryは`fadeIn()`, `hide()`, `slideDown()`をしてこのコンセプトにくのけをしますが、そのはられており、それにりてるのタスクのみをっています。

Jqueryは`.animate()`というくほどなメソッドをすることでこのをしています。このメソッドをすると、をえてふをえるcssプロパティをするカスタムアニメーションをできます。たとえば、CSSスタイルのプロパティに`width:200;` DOMののは50です。アニメーションメソッドは、されたCSSか

らのをらし、そのを150にアニメートします。アニメーションエンジンがするので、このについてにするはありません。

```
<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>
```

.animate() メソッドでできる**CSS**スタイルのプロパティのリスト。

```
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,
```

.animate() メソッドでされた。

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

.animate() メソッドでされたィージング。

"スイング"
"リニア"

なアニメーションオプションのをいくつかします。

1

```
$( "#book" ).animate({
  width: [ "toggle", "swing" ],
  height: [ "toggle", "swing" ],
  opacity: "toggle"
 }, 5000, "linear", function() {
    $( this ).after( "<div>Animation complete.</div>" );
});
```

2

```
  $("#box").animate({
     height: "300px",
     width: "300px"
     }, {
```

```
    duration: 5000,
    easing: "linear",
    complete: function(){
       $(this).after("<p>Animation is complete!</p>");
    }
});
```

オンラインでjQuery .animateメソッドをむ https://riptutorial.com/ja/jquery/topic/5064/jquery--animate--メソッド

7: jQueryオブジェクトと

き

jQueryのは、ビルディングブロックでをさせるなです。これにより、コールバックのいスクールネストがきえられます。

Examples

なの

ここでは、「えられたがしたときにむことをする」のになをします。これは、しいDeferredオブジェクトをすることでいます。これはでされ、Deferredのをします

```
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();
}
```

そして、このようにう

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

のの

のタスクをにするがあるは、そのプロミスオブジェクトをさせるがあります。ここにながあります

```
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の、エラーVS .done、.fail

とエラー Ajaxリクエストのにびされるコールバック。

のにエラーがしたにびされるコールバック。

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

.doneと.fail

.ajax。donefunctiondata、textStatus、jqXHR{}; jQuery 1.8でされたメソッド.successをきえます。これはのsuccessコールバックのです。

.ajax。failfunctionjqXHR、textStatus、errorThrown{}; jQuery 1.8でされたメソッド.errorをきえます。これはのなコールバックのわりのです。

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

ののをする

デフォルトでは、プロミスのはにです。をしたオブジェクトがそれを/すると、のがされます。

```
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"
```

オンラインでjQueryオブジェクトとをむ https://riptutorial.com/ja/jquery/topic/8308/jqueryオブジェクトと

8: イベント

jQueryはにaddEventListenerをしてイベントをします。つまり、じDOMにしてじイベントにのをバインドすることはにです。

Examples

イベントハンドラのアタッチとデタッチ

イベントハンドラをアタッチする

バージョン1.7、jQueryにはイベントAPIの`.on()`ます。これにより、されているjQueryにのJavaScriptイベントまたはカスタムイベントをバインドできます。 `.click()`などのショートカットがありますが、 `.on()`はさらにくのオプションをします。

HTML

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

jQuery

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

イベントハンドラをデタッチする

もちろん、jQueryオブジェクトからイベントをりすこともできます。これは`.off( events [, selector ] [, handler ] )`をって`.off( events [, selector ] [, handler ] )`。

HTML

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

jQuery

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

ボタン`$(this)`をクリックすると、のjQueryオブジェクトがされ、されているすべてのイベントハンドラがされます。するイベントハンドラをすることもできます。

jQuery

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

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

この、 `mouseenter`イベントは、クリックしたもききします。

されたイベント

をてみましょう。ここでは、になHTMLのをします。

HTMLの

```
<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>
```

このでは、すべての`<a>`にイベントリスナーをします。は、こののリストがであることです。 `<li>`のとともにエレメントがおよびされます。しかし、ページはのにされないため、リンクオブジェクトつまり`$('a').click()` になクリックイベントリスナーをすることができ`$('a').click()`。

たちがっているは、りする<a>にイベントをするです。

- イベント

されたイベントは、イベントのイベントのバブルとばれることがいのためにのみです。イベントがすると、いつでもドキュメントルートにバブルされます。らは、そのの「」のイベント、にイベントのをします。

したがって、のでは、 <a>のリンクをクリックすると、のの「クリック」イベントがこのでトリガーされます。

- a
- リ
- ul
-
- html
- ドキュメントルート

どのようなイベントのバブリングがこっているかをることで、たちはHTMLをじてしているのイベントの1つをキャッチできます。

このでは、 はでないため、キャッチするのにしています。

```
$('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
});
```

- このイベントリスナーのである 'ul'があります
- のパラメータ 'click'は、しようとしているイベントをします。
- 2のパラメータ 'a'は、イベントの このイベントリスナのulのにあるすべてののにされます。
- に、3のパラメータは、1および2のパラメータのがたされたにされるコードです。

ソリューションのみの

1. ユーザーが<a>をクリックする
2. それは<a>のclickイベントをトリガーします。
3. このイベントは、ドキュメントルートにけてバブリングをします。
4. イベントはに<li>をバブルし、 をバブルします。
5. にイベントリスナーがけられているため、イベントリスナーがされます。

6. イベントリスナーは、にトリガーイベントをします。バブリングイベントは「クリック」で、リスナーは「クリック」しています。パスです。
7. リスナーチェックは、バブルチェーンのに2のパラメータ 'a'をさせようとします。チェーンののが 'a'であるため、これはフィルタにし、これもパスです。
8. のパラメータのコードは、それはだとしてマッチしたアイテムをしてされ`this`。このに`stopPropagation()`びしがまれていない、イベントはルート `document` にかってにしけます。

でしないながでない/でないは、 `document`をするがあり`document`。として、のから`'body'`をしないでください

- `body`は、マウスイベントがバブルしないことをするスタイリングとするバグがあります。これはブラウザにし、されたボディのさが0のたとえば、すべてのがをつなどにするがあります。マウスイベントはに`document`バブルし`document`。
- `document` にスクリプトにするため、されたハンドラを*DOM*のハンドラの`document`して、それがすることをしてください。

ドキュメントみみイベント.load

イメージやPDFなど、のリソースがロードされるまでスクリプトをさせたいは、 `.on( "load", handler)`ショートカットのためのショートカットである`.load()`できます。

HTML

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

jQuery

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

IDをせずにりしのイベント

のインスタンスでかをするためにイベントがしたかをるがあるのりしがページにあります。

- すべてのにのクラスをえる
- イベントリスナーをクラスにします。 `this`イベントハンドラは、イベントがしたセレクタです
- `this`インスタンスをすることによって、ものコンテナにします。
- そのコンテナの`find()`して、そのインスタンスにののをする

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');
    });
  });
});
```

オリジナル イベント

jQueryイベントではできないプロパティがすることがあります。になるプロパティにアクセスするには、 `Event.originalEvent` をします `Event.originalEvent`

スクロールをする

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

jQueryをしてのイベントをオンまたはオフにりえます。 きリスナー

にしたすべてのリスナーをオフにしたいがあります。

```
//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")
```

このメソッドのは、のプラグインなどによって`document`にバインドされたすべてのリスナーもされることです。

くの、たちだけがしたすべてのリスナーをりしたいとっています。

これをするために、きリスナーを、

```
//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");
```

これにより、のクリックリスナーがってされることがなくなります。

オンラインでイベントをむ https://riptutorial.com/ja/jquery/topic/1321/イベント

9: セレクタ

き

jQueryセレクタは、HTMLドキュメントのDOMドキュメントオブジェクトモデルをまたはします。 ID、、タイプ、、クラスなどについてHTMLをするためにされます。のCSSセレクタについています。

- タグマーカーなし、タグを
- Id `#id`
- クラス `.className`
- `[attributeName]`
- をつ `[attributeName ='value']`
- は`^=`まります `[attributeName ^= 'value']`
- は`$=`わります `[attributeName $='value']`
- に`*=`がまれています `[attributeName *= 'value']`
- セレクタ `:pseudo-selector`
- のセレクタのスペース
- の `>`セレクタの
- のの `+`
- にくしない `~`
- またはセレクタのに, カンマ

`class`や`id selectors`や、をむ`attribute`をくときなど

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

2つのバックスラッシュ`\\`をしてをエスケープするがあります。

えば。

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

セレクタはフレームのように、

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

Examples

セレクタの

jQueryでは、のくのさまざまなプロパティをしてページのをできます。

- タイプ
- クラス
- ID
- の
-
- インデックスきセレクタ
-

あなたがCSSセレクタをっているなら、jQueryのセレクタがじであることにくでしょうさなをいて。

のようなHTMLをにとります

```
<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 -->
```

タイプによる

のjQueryセレクタは、1、2、3、4をむすべての`<a>`をします。

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

クラスでする

のjQueryセレクタは、クラスの`example`すべてのnon-aをむを3,4,5にします。

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

IDでする

のjQueryセレクタは、されたID2をつをします。

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

のによる

のjQueryセレクタは、された`href`をつすべての1と4をむをします。

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

による

のjQueryセレクタは、 `href`がするすべてのを`index.html`というでします。 は1だけです。

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

インデックスきのによる インデックスきセレクタ

のjQueryセレクタは1つだけをし、2つの`<a>`はします。 `second-link`されたインデックスがあるため、 `1`のような`eq(1)`インデックスはからまることにしてください`0`ここでされてしまったので、。

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

インデックスきでする

インデックスをしてをするには`:not(:eq())`

の`example`では、 `<a>`をしていますが、クラスの`example`では1

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

をする

をからするには、のようにし`:not()`

の`example`では、 `<a>`をしていますが、クラスの`example`では1と2です。

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

による

`:first-child`、 `:last-child`、 `:first-of-type`、 `:last-of-type`など、をしてjQueryをすることもできます。

のjQueryセレクタはの`<a>`number 1のみをします。

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

jQueryセレクタのみわせ

また、のjQueryセレクタをみわせてをめることもできます。のをみわせたり、それらのすべてをみわせることができます。のクラス、、およびをにすることもできます。

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

これは`<a>`をします

- `class1, class2, and class3`クラスがあります。
- のIDをっている `someID`
- のを`attr1`ます `attr1`
- のとをちます `attr2`に`something`、 `attr3`に`something`
- `first-child`と`first-of-type`

なるセレクタをカンマでることもできます。

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

これにより、

- すべての<a>
- クラスclass1をつすべての
- id #someIDの

との

jQueryセレクタは、CSSとじにしているため、じでとをすることができます。

- なをするには、スペ―スをします
- のをするには、a >
- のにくするをするには、 +
- にくりわせのをするには、 ~

ワイルドカ―ド

すべてのをしたいが、するのプロパティクラス、などがないがあります。その、 * selectorはすべてのをするだけです

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

セレクタのみわせ

DOMにうことをする

```
<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>
```

セレクタ

`<ul>` - `parentUl`がその `<li>` をつけると、

1. シンプルな**`$('parent child')`**

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

   これにより、されたのすべてのレベルがするがすべてになります。

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

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

   これにより、すべてのするがされます のレベルのみ 。

3. コンテキストベースセレクタ - **`$('child','parent')`**

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

   これは1.とじです。

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

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

   これは1.とじです。

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

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

   これは2.とじです。

のコンビネータ

グループセレクタ "、"

すべての`<ul>`とすべての`<li>`とすべての`<span>`をします。

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

のセレクタ ""なし

`parentUl`クラスのすべての`<ul>`をします。

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

するセレクタ "+"

の<li>のにされているすべての<li>をします。

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

なセレクタ "~"

の<li>のであるすべての<li>をします。

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

は、 jQueryセレクタをしてjQueryによってできます。このは、またはのリストのいずれかをします。

セレクタ

```
$("*")                      // 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
```

```
$("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>
```

キャッシングセレクタ

jQueryでセレクタをするたびに、DOMでクエリにするがされます。これをにまたはりしうと、パフォーマンスがします。のセレクタをする、そのセレクタをにりてることによってキャッシュにするがあります。

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

これはをきえます

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

ウェブサイトでこのをに/にするがあるは、このセレクタをキャッシュするとちます。じセレクタをつのがある、はののになります。

```
<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>
```

は、へのりてにDOMにするがあります。 `child`というクラスをつDOMにがないは、そのにのをします。

```
<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>
```

セレクタでDOMのを/した、セレクタをにりてすることをれないでください。

セレクタをキャッシュするとき、くのが`$`をってをし、がjQueryオブジェクトであることをします。

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

セレクタとしてのDOM

jQueryはさまざまなパラメータをけり、そのうちの1つはのDOMです。 DOMをjQueryにすと、jQueryオブジェクトのなのようながそのをします。

jQueryは、nodeTypeをべることによって、がDOMであることをします。

DOMのもなは、jQuery APIへのアクセスをるために、のがjQueryコンストラクターにされるコールバックです。

`each`コールバックのようにそれぞれはです。

```
$(".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
});
```

セレクタとしてのHTML

jQueryは、さまざまなパラメータを「セレクタ」としてけれ、そのうちの1つはHTMLです。HTMLをjQueryにすと、 jQueryオブジェクトのなのようなが、としてされたHTMLをします。

jQueryのは、は、コンストラクタにされるHTMLstringであり、またそれがでめなければならないことをするためにをしています。 そのは、 `rquickExpr = /^(?:\s*(<[\w\W]+>)[^>]*|#([\w-]*))$/` / regex101.com 。

HTMLをセレクタとしてするもなは、DOMのセットをコードでのみするがあるです。これは、Modal popoutなどのライブラリでされることがよくあります。

たとえば、divにラップされたアンカタグをテンプレートとしてした

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

jQueryオブジェクトをします。

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

`template("google.com","Google")`とばれる

オンラインでセレクタをむ https://riptutorial.com/ja/jquery/topic/389/セレクタ

10: チェックボックスすべてのチェックボックスをオンにする

き

は々なStackoverflowのとえをって、 "select all"チェックボックスとチェック/グループチェックボックスのステータスがわったかどうかをチェックするについて、このになにました。select allグループをするには、 "select all" idがとするがあります。このでは、select all IDはcbGroup1です。もcbGroup1です。

コードはにいですが、くのifとリソースのではありません。

Examples

2すべてのチェックボックスをするグループチェックボックスでします

```
<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>
```

オンラインでチェックボックスすべてのチェックボックスをオンにするをむ
https://riptutorial.com/ja/jquery/topic/10076/チェックボックスすべてのチェックボックスをオンにする

11: プラグイン

Examples

プラグイン - はじめに

jQuery APIは、プロトタイプをすることでできます。えば、のAPIは、すでにそのようななくのをしている`.hide()` `.fadeIn()` `.hasClass()`

jQueryプロトタイプは`$.fn`でされています。ソースコードにはがまれています

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

このプロトタイプにをすることで、されたjQueryオブジェクトからこれらのをびすことができますこれはjQueryのびしでにされます。

されたjQueryオブジェクトは、されたセレクタにづいてのをします。たとえば`$('.active')` .active `$('.active')`はびしにアクティブなクラスをつをするjQueryオブジェクトをしますこれはライブのセットではありません。

プラグインの`this`は、されたjQueryオブジェクトをします。その、 `this`はしたセットをすためにされます。

プラグイン

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

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

jsFiddleの

と

のとはなり 、jQueryプラグインは**Chainable**であるとされます。
これは、 `$(".warn").append("WARNING! ").css({color:"red"})`ようなじのにのメソッドをするがあることをし`.css()` `.append()`の`.css()`メソッドは、のメソッドがじ`.warn`コレクションにされます

さまざまなカスタマイズオプションをすなるコレクションでじプラグインをできるようにすることは、 カスタマイズ/においてなをたします

```
(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デモ

のは、なプラグインのをするにあります。られたのカスタマイズオプションにするのではなく、してください。

えばハイライトされるテキストをすことができる`.highlight()`プラグインをし、スタイルにしてのを`.highlight()`せたいとします。

```
//...
// 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
});
//...
```

ユーザーはのテキストをし、カスタムCSSクラスをしてのスタイルをにをつことができます。

```
$("#content").highlight({
    text : "hello",
    class : "makeYellowBig"
});
```

jsFiddleの

jQuery.fn.extend メソッド

このメソッドはjQueryプロトタイプ$ .fnオブジェクトをし、jQueryにできるしいカスタムメソッドをします。

えば

```
<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>
```

オンラインでプラグインをむ https://riptutorial.com/ja/jquery/topic/1805/プラグイン

12:

Examples

をコンテナにする

1

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

2

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

のは、プリペンドさ`#child`ににを`#parent`。

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

</div>
```

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

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

`prepend()` - パラメータでされたを、するのセットののにします。

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)`

JQuery *`version: 1.4`*では、コールバックをとしてできます。ここでは、ののインデックスとのいHTMLとしてをできます。では、 `this`はセットののをします。

```
jQuery('#parent').prepend(function(i,oldHTML){
     // return the value to be prepend
```

```
    return  '<span>child</span>';
});
```

オンラインでをむ https://riptutorial.com/ja/jquery/topic/1909/

13:

Examples

```
// 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

HTML

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

スクリプト

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

このようにして、リストのについてメッセージがされる。

0マンゴー

1

オンラインでをむ https://riptutorial.com/ja/jquery/topic/10853/

14:

jQuery.attr() 、するのセットののののをします。または、するすべてのの1つのをします。

$("input").attr("type");これは、のをするときにセレクタとするの $("input").attr("type");するだけで、のの、ある

ただし、をすると、するすべてのにがされます。

Examples

HTMLのをする

のパラメータが.attr()にされると、されたのされたのがされます。

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

HTML

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

jQuery

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

dataの

jQueryはデータをうために.data()をしています。 .dataは、したのdataのをします。

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

Html

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

jQuery

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

jQueryのdataメソッドはdata-*にアクセスできますが、のをします。

HTMLの

あるにをするは、 attr(attributeName, attributeValue)をできます。えば

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

このでは、ページのすべてのリンクにマウスオーバーテキスト"Click me"をします。

じをしてのをします。

の

からをするには、 `.removeAttr(attributeName)` をします。えば

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

ID `home` から `title` がされます。

attrとpropのい

`attr()` は、DOM `getAttribute()` および `setAttribute()` をしてHTMLをまたはします。 `prop()` は、をせずにDOMプロパティをすることによってします。くの、2つはですが、にはもうがです。

チェックボックスをオンにしてするには

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

プロパティをするには、 `removeProp()` メソッドをします。に、 `removeAttr()` はをします。

オンラインでをむ https://riptutorial.com/ja/jquery/topic/4429/

15: イベント

Examples

ドキュメントとはですかどのようにしますか

jQueryコードはしばしば`jQuery(function($) { ... });`ラップされ`jQuery(function($) { ... });`そのため、DOMのロードがしたにのみされます。

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

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

jQueryとJavaScriptはにページにレンダリングされていないDOMをできないため、これはです。

```
<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>
```

カスタムハンドラを`.ready()`メソッドにすことで、jQueryネームスペースのエイリアスをできます。これは、のJSライブラリが*jQuery*とじされた`$`エイリアスをしているにです。このをするには、`$.noConflict();`びすがあります`$.noConflict();` - これにより、い`$`エイリアスのわりにデフォルトの*jQuery*だけがされます。
`.ready()`ハンドラにカスタムハンドラをすことによって、 *jQuery*をするためにエイリアスをすることができます。

```
$.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>')
});
```

にページのにjQueryコードをくのではなく、 `$(document).ready`をすると、すべてのHTMLがレン

ダリングされ、Document Object ModelDOMがJavaScriptコードをできるになります。

jQuery 2.2.3

これらはすべてです。ブロックのコードは、ドキュメントのがうとされます。

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

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

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

これらはであるため、はされるです。は、じをする`$`ではなく`jQuery`キーワードをしたバージョンです。

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

jQuery 3.0

jQuery 3.0、このフォームのみをします。

```
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
});
```

そののドキュメントハンドラはjQuery 3.0ではです。

jQuery 3.0、レディハンドラはににびされます 。これは、のコードでは、がにがっているかどうかにかかわらず、にログ'ハンドラ'がにされることをします。

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

>ハンドラ
>ハンドラ

$document.readyと$window.loadのい

`$(window).load()`は**jQuery**バージョン1.8ではされました **jQuery 3.0**からにされました。 このイベントについてのjQueryページには、のがされています

> ィメージとにされたのロードィベントの
>
> が`.load()`ショートカットをしてしようとするなは、またはのコレクションがにみまれたときにをすることです。すべきいくつかののがあります。これらは
>
> - それは、ブラウザでももなくしません
> - イメージ`src`がとじ`src`されている、WebKitではしくしません
> - DOMツリーをしくてない
> - ブラウザのキャッシュににされているのためにすることができます

それでもなお`load()`をしたいは、のようにかれています

`$(document).ready()`は、なDOMがになるまでっています.HTMLのすべてのがされ、ドキュメントにまれています。ただし、イメージなどのリソースがこのでにロードされていないがあります。すべてのリソースがロードされるまでつことがなは、 `$(window).load()` をし、このイベントのなをしているは、わりにをできます。

```
$(document).ready(function() {
  console.log($("#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
});
```

イベントをアタッチし、readyの**DOM**をする

`$(document).ready()`

1. イベントハンドラのアタッチ
   jQueryイベントハンドラをアタッチする

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

2. ページのに**jQuery**コードをする

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

3. ロードされた**DOM**をする

にページがみまれたときに`div`にし、ボタンのクリックイベントでする

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

jQueryfnとのコードのい

document-readyイベントをすると、300msのをうさなパフォーマンスのがあります。じた`</body>`タグのでコードをすると、じがすることがあります。

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

のをしますが、ドキュメントレディイベントトリガーがのようにされるのをたずにすぐにされます。

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

のは、じる`</body>`タグの、には`span`タグのにスクリプトのページとにするにしていることをしています。

オンラインでイベントをむ https://riptutorial.com/ja/jquery/topic/500/イベント

16: の

パラ メーター

パラメータ	
	された、.hide() .show()と.toggle()アニメーションされます。がにフェードインまたはフェードインします。

Examples

```
$(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
});
```

をりえる

な`toggle()`ケース

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

の

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

...とコールバック

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

...またはイージングとコールバックで。

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

...またはさまざまなオプションがあります。

```
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');
      }
    }
  );
}
```

slideToggle()アニメーションとしてスライドをすることもできます

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

...または**fadeToggle()**をしてフェードイン/アウトする

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

...または**toggleClass()**クラスをりえる

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

なケースは、あるをするために**toggle()**をし、のじクラスをすことです。

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

のはすべてここにあります

オンラインでのをむ https://riptutorial.com/ja/jquery/topic/1298/の

17: のとさのと

Examples

とさのとを

とさをする

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

とさをする

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

innerWidthとinnerHeightのとパディングとボーダーをする

とさをする

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

とさをする

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

outerWidthとouterHeightのとパディングとボーダーをむ

とさをするマージンをく

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

とさをするマージンをむ

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

とさをする

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

オンラインでのとさのとをむ https://riptutorial.com/ja/jquery/topic/2167/のとさのと

18: する

1. $セレクタ.appendcontent
2. $content.appendToセレクタ

パラ メーター

パラ メーター	
コンテンツ	な、HTML、テキスト、、オブジェクト、さらにはをす。

- .appendと.afterはコードをするがあります。これは、スクリプトタグをしたり、コードをするHTMLたとえば、をしてうことができます。これらのメソッドをして、URLクエリパラメータ、Cookie、フォームなどのできないソースからしたをしないでください。そうすることで、クロスサイトスクリプティングXSSのがされるがあります。ドキュメントにコンテンツをするに、ユーザーをまたはエスケープします。

- jQueryはにSVGをサポートしていません。 SVGでのjQueryメソッドの
  そのについてにされていないり、しないをきこすがあります。 SVGをサポートするメソッドの
  jQuery 3.0はaddClassとremoveClassです。

Examples

コンテナにをする

1

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

2

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

のは、さ`#child`にににを`#parent`。

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

</div>
```

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

  </div>
</div>
```

すでにドキュメントにしないコンテンツをすると、このコンテンツはのコンテナからされ、しいコンテナにされます。したがって、 `.append()`または`.appendTo()`をしてをすることはできません。クローンがなは`.clone()` - > [http://api.jquery.com/clone/][1]

な.appendの

める

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" }
];
```

ループにする

あなたはちょうどきなデータのをけりました。はそれをループしてページにレンダリングするときです。

あなたののえは、このようなことをしているかもしれません

```
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)
        )
    );
}
```

これはにで 、あなたがしていたものをにレンダリングしますが...

こんなことしないで。

300のデータをえていますか

それぞれがのスタイルとに、すべてのの、さびをするためにブラウザをする-それらはによってされているをき、レイアウト 、これはながらこのでは、それらはのであるように`<table>`、 らがすることはできません。

でがない、このパフォーマンスのはごくわずかです。しかし、々はミリごとにカウントしたい。

よりいオプション

1. のにし、ループがしたにする

```
/**
 * 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);
```

これらのオプションのうち、これはjQueryにもしています。

---

2. の**Array**。*メソッドの

```
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);
```

それのものとにで、みやすい。

---

3. **HTML**のをする**jQuery**みみメソッドのわりに

```
// ...
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(''));
```

にですが、もう、されません。これにより、jQueryはにのテキストをにするがあり、ありません。jQueryは、しくされたときのににれています。

---

4. でをし、のにする

```
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);
```

のなおにり 。これは、jQueryがよりいレベルでうことのなえをしています。

よりいダイビング

- jQueryソースビューア
- Array.prototype.join
- Array.prototype.map
- Array.prototype.reduce
- document.createDocumentFragment
- document.createTextNode

- Google Web Fundamentals - パフォーマンス

jQueryをする

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>
```

スクリプト

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

オンラインでするをむ https://riptutorial.com/ja/jquery/topic/1910/する

クレジット

S. No		Contributors
1	jQueryをいめる	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	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
4	DOMトラバース	A.J, charlietfl, Community, dlsso, mark.hch, rmondesilva, SGS Venkatesh, sucil, Sverri M. Olsen, The_Outsider, Zaz
5	DOM	Angelos Chalaris, Assimilater, Brock Davis, DawnPaladin, DefyGravity, Deryck, Marimba, Mark Schultheiss, martincarlin87, Neal, Paul Roub, Shaunak D, still_learning
6	jQuery .animateメソッド	RamenChef, Rust in Peace, Simplans, VJS
7	jQueryオブジェクトと	Alex, Ashiquzzaman, Gone Coding
8	イベント	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
9	セレクタ	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
10	チェックボックスすべてのチェックボックスをオンにする	user1851673
11	プラグイン	hasan, Roko C. Buljan, Travis J
12		Ashkan Mobayen Khiabani, empiric, J F, Pranav C Balan
13		bipon, Renier
14		A.J, acdcjunior, ban17, ochi, Scimonster
15	イベント	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
16	の	Alex Char, Paul Roub, Rupali Pemare, The_Outsider, Theodore K., user2314737, Zaz
17	のとさのと	Ashkan Mobayen Khiabani
18	する	Ashkan Mobayen Khiabani, bipon, Community, Darshak, Deryck, empiric, Flyer53, J F, JF it, Paul Roub, Pranav C Balan, Proto, Shaunak D