ななちゃんのIT教室

jQueryに挑戦の巻

by nara.yasuhiro@gmail.com

ななちゃんが
jQuery に挑戦するという お話

第 0.6 版　2018 年 2 月 17 日

フリー素材
http://freeillustration.net

もくじ

第1回 jQueryとは

なな: 今回のお題の「jQuery」って何？

先生: 2006年に開発、公開された JavaScript ライブラリよ。ライブラリは便利な関数群のこと。jQuery は、CSSに近い記述方法を採用しているのが特徴なの。

なな: どんな使い方をするの？

先生: jQuery ライブラリを jQuery のサイト（http://jquery.com/） からダウンロードします。たとえば jquery-3.2.1.min.js というファイル。そして、それを html/JavaScript ファイルの script 要素を使って読み込むの。<script src="jquery-3.2.1.min.js"></script> みたいな形で。また、直接読み込む方法もあります。<script src="http://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script> みたいに。

なな: これは準備作業よね。JavaScript プログラムはどんなふうに書くの？。

先生:
```
<div>1</div><div>2</div><div>3</div><div>4</div><div>5</div>
<script> $("div").css("color", "red"); </script>
```

なな: 変わった形をしているのね。

先生: html のどの部分を操作するかを指定（**セレクタ部**）し、その上で操作する内容（**操作部**）を書くのが基本よ。

なな: **セレクタ部**？

先生: 「$(...)」 の形をしています。「$」は、「jQuery」と書いても良いんだけど、頻繁に使うので短いほうを使います。具体的には、「（$("div")」 みたいになります。指定した要素、ここでは div 要素の全てが対象になります。JavaScript 的には jQuery （ $ ）という関数が、引数で指定したタグ要素の情報を集め、「jQuery オブジェクト」 として返していることになります。

なな: **操作部**は？

先生: 例えば、css("color", "red")」。色を赤に指定しています。JavaScript 的には、「jQuery オブジェクト」の css メソッドということになります。

なな: 1 2 3 4 5
<script> $("div").css("color", "red"); </script>
は、どういう動作をするの？

先生: 「1」～「5」をすべて赤い文字に変えます。

なな: 全体をまとめると、

```
<div>1</div><div>2</div><div>3</div><div>4</div><div>5</div>
<script src="jquery-3.2.1.min.js"></script>
<script>
  $("div").css("color", "red");
</script>
```

こんな感じになって、実行結果は

1
2
3
4
5

こんな感じになるのね。

第2回 セレクタ部

なな: 今回は、「**セレクタ部**」ね。

先生: セレクタ部の書き方は、なるべく CSS と同じになるように設計されているの。
つまり、「**要素**」(タグ)、「**ID**」、「**クラス**」の 3 通りの指定ができます。

なな: まず、「**要素セレクタ**」から教えてください。

先生: 特定の要素 (タグ) を一括して指定します。 つまり、指定した要素が複数存在すると、その全部が対象になります。記述方法は、「$("div")」のように、要素名を「"」で囲む形で指定します。

なな: 「**IDセレクタ**」は?

先生: <div **id=helloworld**>hello なんかの、id 属性 を指定します。特定の 1 つの要素のみを指定することになります。記述方法は、id 属性名 の前に「#」をつけます。たとえば「$("#helloworld")」。 要素セレクタと ID セレクタをつなげて「$("div#helloworld")」のように書くこともできます。操作部と合わせると「$("#helloworld").css("color", "red");」のようになります。

なな: 「**クラスセレクタ**」は?

先生: 3 なんかの class 属性 を指定します。複数存在すると、全部が対象になります。記述方法は、class 属性名 の前に「.」をつけます。たとえば「$(".article")」。 要素セレクタ と ID セレクタをつなげて「$("div.article")」のように書くこともできます。

なな: それ以外にも、何かあるの?

先生: 「**子孫セレクタ**」というのもあります。:ある要素の更に内側にある要素を絞り込むの。記述方法は、親要素と子孫要素をスペースで区切って連続して書きます。 たとえば、 abcdef に対して、「$("div strong")」、それから、<div class=article>abcdef</div> に対して、「$(".article strong")」。

それから、「**ユニバーサルセレクタ**」。これは、全ての要素を選択します。記述方法は「 * 」(アスタリスク)を使います。たとえば、div 要素の中の全ての子要素を指定するには、「$("div *")」とします。

「**グループセレクタ**」は、複数のセレクタを並列に指定するもの。記述方法は「 , 」で区切ります。たとえば、「$("#question1,#question2")」。

なな: なんとなく分かったような気もするけど、そもそも JavaScript的に、セレクタの正体は?

先生: セレクタは、JavaScript 的には、関数 jQuery ----「 $ 」と略記するけど ---- の呼び出しであって、返す値は「jQuery オブジェクト」です。そして、操作部は、jQuery オブジェクトのメソッドということになるの。

1 2 3 4 5

に対して、$("div")[0].innerHTML = "XXXXX"; のようなこともできます。1 が XXXXX に変化。また、

$(document.getElementsByTagName("div")).text("XXXXX");

のようなこともできます。1 ～ 5 すべてが XXXXX に変化。要素へのポインタの配列に似ているけど、

(document.getElementsByTagName("div")).text("XXXXX");

のようなことはできません。 $( ) で、jQuery オブジェクトに変換しないと、text("XXXXX") のような操作部 (= jQuery オブジェクトのメソッド) は使えません。

第3回 操作部（基本）

なな：今回は、「操作部」ね。

先生：操作部には、いくつかの種類があります。まず、**CSS を制御**するには、「**css(プロパティ名，値)**」の形にします。プロパティ名は、たとえば、「"color"」や「"background-color"」。値はそのプロパティに設定する値の文字列。かならず、「 “ 」で囲んでください。複数のプロパティを同時に変更することも可能ですが、この場合は、プロパティ名は CSS のプロパティ名とは異なり、JavaScript で CSS を指定する際に使われるプロパティ名 になります。つまり、「background-color」は「-」を削除し、次の頭文字を大文字にして、「backgroundColor」のように変えます。たとえば、$("div").css({ backgroundColor: "red", color: "blue" }); のようにします。

なな：他の種類は？

先生：**CSSのプロパティの値を得る**には、「**css(プロパティ名)**」の形にします。たとえば、color プロパティの値を調べるには、document.write("色は" + $("#article").css("color") + "です。"); のようにします。

なな：他には？

先生：**HTML要素の中身を書き換える**には、「**html(“書き換えるhtml文字列”)**」の形にします。**中身の HTML 記述を調べる**場合は、引数を空にします。たとえば、

```
<div id="weather">晴れ</div>
<script> $("#weather").html("<strong>雨</strong>時々<strong>曇り</strong>"); </script>
```

タグを含まない文字列を使う場合は、html() のかわりに、text() を使います。出力の場合、文字列にタグが含まれると、そのまま表示します。中身を調べる時には、タグを除いたものが得られます。

なな：他には？

先生：**input 要素からの値の取得**は「**val()**」、**input 要素への出力**は、「**val(“文字列”)**」。

```
<input id="text1" type="text" size="20" value="abc">
<script> alert($("input#text1").val());$("input#text1").val("def"); </script>
```

だったら、「abc」とアラートしてから、「def」に書き換えます。

なな：他には？

先生：**透明度を設定**する **fadeTo(速度，透明度)**; というのもあります。速度は、"slow"、"normal"、"fast" や、完了までの時間をミリ秒単位で指定します。例えば「"1500"」または「1500」とすれば、1.5秒かけてアニメーションが行われます。透明度は、「1」を100%の濃度（透過しない状態）、「0」を完全に透明な状態として指定します。例えば「0.33」なら、33%の見え方。その他、**fadeOut(速度)**、**fadeIn(速度)** をはじめとする、さまざまなフェーディング効果の操作が用意されています。

第4回　操作部（その他）

なな：今回は、「操作部」の追加ね。

先生：**HTML要素の挿入、移動**に関するものをまとめましょう。

sel1.prepend(sel2)	sel1で指定された要素の中の先頭に、sel2で指定された要素を移動する。
sel1.append(sel2)	sel1で指定された要素の中の末尾に、sel2で指定された要素を移動する。
sel1.before(sel2)	sel1で指定された要素の直前に、　sel2で指定された要素を移動する。
sel1.after(sel2)	sel1で指定された要素の直後に、　sel2で指定された要素を移動する。

メソッドは、sel1 を値として返す。sel2 のところに $(" 1 ") または、" 1 " などのように、html を書くこともできる。（html を書く場合は、新規生成＋移動なので、挿入というイメージになる）

sel2.prependTo(sel1)	sel1で指定された要素の中の先頭に、sel2で指定された要素を移動する。
sel2.appendTo(sel1)	sel1で指定された要素の中の末尾に、sel2で指定された要素を移動する。
sel2.insertBefore(sel1)	sel1で指定された要素の直前に、　sel2で指定された要素を移動する。
sel2.insertAfter(sel1)	sel1で指定された要素の直後に、　sel2で指定された要素を移動する。

メソッドは、移動後の要素を値として返す。sel2 のところに、$(" 1 ") などのように、html を書くこともできる。（html を書く場合は、新規生成＋移動なので、挿入というイメージになる）

たとえば、「$(" 1 ").appendTo("body");」は、body 要素の末尾に タグを追加します。

なな：他には？

先生：**メソッドチェーン**というのがあります。

```
<div id=here>4</div>
$("div#here").prepend("3");
$("div#here").prepend("2");
$("div#here").prepend("1");
```

は

```
$("div#here").prepend("3").prepend("2").prepend("1");
```

と書くことができます。

なな：他には？

先生：複数の要素が選択される可能性があるセレクタでは、**length プロパティ**で、**選択された要素の個数**が得られます。size() でも、同じ結果が得られます。

```
<div>1</div><div>2</div><div>3</div><div>4</div><div>5</div>
var num = $("div").length; // 5

var num = $("div").size(); // 5
```

なな：他には？

先生：each() というのもあります。

```
$("div").each(function(index,elm) { $(this).html(index + ":" + $(this).html())});
$("div").each(function(index,elm) { this.innerHTML = index + ":" + this.innerHTML });
$("div").each(function(index,elm) { elm.innerHTML = index + ":" + elm.innerHTML });
```

といった書き方で、複数ある div の中身に通し番号を付加することができます。

なな：いろいろ、便利な道具があるのね！

第5回 イベントモデル

なな：イベントモデルって何？

先生：セレクタで選んだ要素上でクリックなどのイベントが発生した時のイベントハンドラを指定することよ。たとえば、

```
<button>Click me</button>
<script> $("button").click(function() { alert("Click!!"); }); </script>
```

は、ボタンをクリックしたときに alert("Click!!") を実行します。関数の定義を分けて書くこともできます。

```
$("button").click(f);
function f() { alert("Click!!"); }
```

なな：操作部と同じような形をしているけど、**引数欄に関数呼び出しを書く**というのが異なっているのね。

先生：イベント指定を表にまとめておきます。

click()	要素がマウスでクリックされる。
dblclick()	要素がマウスでダブルクリックされる。
mousedown()	要素の上でマウスが押下される。
mouseup()	要素の上で押下されたマウスが放される。
mouseover()	要素の上にマウスが重ねられる。
mouseout()	要素の上に重ねられたマウスが外れる。
focus()	input要素などが選択され入力状態になる。
blur()	input要素などが選択が解除される。
change()	input要素などの値が変化する。
ready()	要素のロードが完了、要素が使えるようになった。
hover(マウスオーバーの処理，マウスアウトの処理)	

先生： $(document).ready(function(){...}); は、**ウェブページの表示が完了した時に発生するイベント**です。 $(function(){...}); と略記が可能です。 それから、「**onメソッド**」というのもあります。

調査範囲.on(イベント名，セレクタ，イベントで実行したい処理)

という書式になります。たとえば、

なな：click! という文字列をクリックすると、clicked! に書き換わるのね！

```
<body>
  <script src="jquery-3.2.1.min.js"></script>
  <script>
    $("<div>click!</div>").appendTo("body");
    $("body").on("click", "div", function () { $("div").text("clicked!"); });
  </script>
</body>
```

それから、イベントハンドラののの中で **this** を使うと、**イベントの発生した要素**を対象にできます。

```
<div id=area>click!</div>
$("div").click(function() { $(this).css("background-color", "red"); });
$("div").click(function() { alert(this.id); });
$("div").click(function() { this.innerHTML = "clicked!" });
```

第6回 $.ajaxによるJSONP読み込み

先生：**$.ajax()** という命令を使って、**JSONP の読み込み**が実現できます。 JSONP については、「ななちゃんのIT教室 マッシュアップの巻」を参照してください。

```
<input type=text id=zip>
<input type=button id=go value="検索">
<div id=out></div>
<br>※本ページは HeartRails Geo API のサービスを使用しています。<br>
<script src="jquery-3.2.1.min.js"></script>
<script>
$("#go").click(function() {
    $.ajax({
      type: 'GET',
      url: 'http://geoapi.heartrails.com/api/json?method=searchByPostal&postal='
            + $("#zip").val() + '&jsonp=callbackf',
      dataType: 'jsonp',
      jsonpCallback: 'callbackf'
    })
    .then (
      function(json) {
          $("#out").html(JSON.stringify(json) + "<br>" +
                                  json.response.location[0].city +
                                  json.response.location[0].town);
      },
      function() {
         $("#out").html('ERROR');
      }
    );
});
</script>
```

なな：jQuery を使わない場合より、ずっとシンプルになっているわね！

第7回 アニメーション

先生: jQuery には、アニメーション表示用の命令もあります。

形式1. セレクタ.animate(params, [duration], [easing], [callback])

params: { '対象プロパティ' : '目的値' }
duration: 継続時間。"slow", "normal", "fast", ミリ秒。デフォルトは "normal"。
easing: 変化関数。"linear", "swing"。デフォルトは "swing"。
callback アニメーション終了時に呼び出す関数

形式2. セレクタ.animate(params, options)

params: { '対象プロパティ' : '目的値' }
options: オブジェクト形式によるオプション。
duration 上記 duration と同じ
easing 上記 easing と同じ
complete 上記 vallback と同じ
step 動作ステップごとに呼び出す関数
queue true ならアニメーションを queue で実行(順次)、
false ならすぐに実行(同時、並行)。デフォルトは true。

な な: ちょっと分かりにくいな、使用例の形で教えてください。

先生: はい。 要素を左右に動かす例です。

```
<button id="left">&lt;&lt;</button> <button id="right">&gt;&gt;</button>
<div class="block">Hello</div>
<style> div { width:100px; border:1px solid; } </style>
<script src="jquery-1.12.2.js"></script>
<script>
$("#right").click(function() { $(".block").animate({"marginLeft": "+=50px"}, "slow"); });
$("#left").click(function()  { $(".block").animate({"marginLeft": "-=50px"}, "slow"); });
</script>
```

こういう書き方もできます。

```
<style> div { width:100px; border:solid 1px; position: absolute; left:10; } </style>
:
$("#right").click(function() { $(".block").animate({"left": "+=50px"}, "slow"); });
$("#left").click(function()  { $(".block").animate({"left": "-=50px"}, "slow"); });
```

“toggle”指定を用いて、div要素に対して”hide”と”show”を繰り返す例です。

```
<button id="animate">animate</button>
<div class="block">Hello</div>
<style> div { width:100px; border:1px solid; background-color:pink; } </style>
<script src="jquery-1.12.2.js"></script>
<script>
$("#animate").click(function(){
  $(".block").animate({ height: "toggle", opacity: "toggle" }, "slow" ); });
</script>
```

な な: 複数のプロパティを同時に変化させることもできるの？

先生：できますよ。

```
<button id="go">Run</button>
<div id="block">Hello!</div>
<style> div { background-color:#bca; width:100px; border:1px solid green; } </style>
<script src="jquery-1.12.2.js"></script>
<script>
$("#go").click(function(){
  $("#block").animate({ width: "70%", opacity: 0.4, marginLeft: "0.6in",
                        fontSize: "3em", borderWidth: "10px" }, 1500 ); });
</script>
```

なな：easing は、"linear" と "swing" しかないの？

先生：はい。でも、easing のユーザ定義も可能です。

```
jQuery.extend(jQuery.easing, { easeInQuart: function (x, t, b, c, d) { return c*(t/=d)*t*t*t + b; }, });
$(".block").animate({ width: "toggle", opacity: "toggle"}, "slow", "easeInQuart");
```

x: 経過時間 / 指定した時間　t: 経過時間
b: 開始値　c: 終了値　d: 指定した時間

こういう定義方法もあります。

```
$.easing.custom = function (x, t, b, c, d) {
    var s = 1.70158;
    if ((t/=d/2) < 1) return c/2*(t*t*(((s*=(1.525))+1)*t - s)) + b;
                      return c/2*((t-=2)*t*(((s*=(1.525))+1)*t + s) + 2) + b;  }
```

ちなみに、組み込み済みのものは下記の 2 つ。swing はブランコのように、はじめと終わりがゆっくり変化。

```
$.easing.linear   = function (x) { return x; }
$.easing.swing    = function (x) { return 0.5 - Math.cos(x * Math.PI) / 2; }
```

なな：一旦、右に移動してから、次に下に移動、みたいなことはできるの？

先生：できるわ。複数の animate 命令を順に実行することができます。「.animate()」を連結することもできます。

```
<div id="wrap"> <div id="box1"></div> <div id="box2"></div> </div>
<style> #wrap            { position: relative; }
        #box2, #box1 { position: absolute; width: 100px; height: 100px; background: pink; }
</style>
<script src="jquery-1.12.2.js"></script>
<script>
$('#box1').animate({ 'left': '500px' });
$('#box2').animate({ 'left': '300px', 'top': '200px' });
$('#box1').animate({ 'left': '500px', 'top': '300px' });
$('#box1').animate({'left':'500px'}).animate({'top':'500px'}).animate({'left':'0px'}).animate({'top':'0px'});
$('#box1').animate({ 'left': '500px'}, 1500);
</script>
```

なな：形式2 の使い方は？

先生：下記のように使います。

```
$('#box1').animate({ 'left': '300px' },{ 'duration': 1500 });
$('#box1').animate({ 'left': '500px' },{ 'duration': 600, 'easing': 'linear' });
$('#box1').animate({ 'left': '700px' },
                   { 'duration': 600, 'complete': function(){ alert('終わりました'); } });
```

第8回 プラグイン

先生: jQueryを拡張するライブラリをjQuery プラグインといいます。jQuery UI は、ユーザー・インターフェイス（UI）に関わる機能を提供するもので、開発チームに jQuery 本家のメンバーも参加しているオフィシャルな存在。

jQuery UIのコンポーネント

・Interactions: マウスによる汎用的な対話処理
・Widgets: 定型的なUI部品、
・Effects jQueryのアニメーションを拡張

・Interactions
- Draggable ドラッグ可能な要素
- Droppable ドロップ可能な要素
- Resizable サイズ変更可能な要素
- Selectable マウス操作で選択可能な要素
- Sortable 並べ替え可能な要素

・Widgets
- Accordion アコーディオン・パネル
- Autocomplete オート・コンプリート機能付きのテキストボックス
- Button ボタンやリンク、ラジオボタン
- Datepicker 日付入力ボックス
- Dialog 汎用的なダイアログ
- Menu 展開可能なリッチ・メニュー
- Progressbar 進捗バー
- Slider スライダー
- Spinner アップダウン・ボタンを伴う数値入力ボックス
- Tabs タブ・パネル
- Tooltip ツールチップ

・Effects
- Effect 基本的なエフェクトを提供
- Show/Hide/Toggle 要素の表示／非表示にエフェクトを適用
- Add/Remove/Toggle/Switch Class スタイル・クラスの適用／解除にエフェクトを適用
- Color Animation 色を徐々に変化させるエフェクトを提供

なな: どうやって使うの？

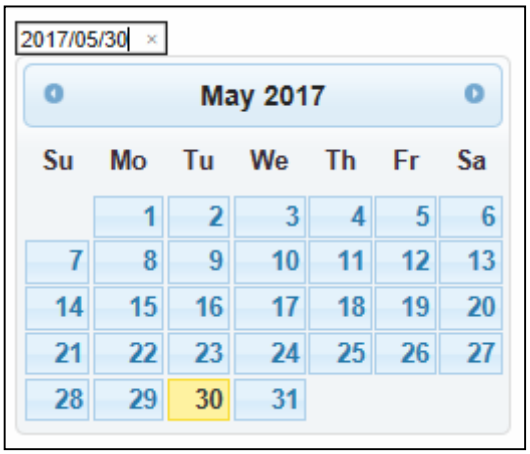
先生: <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/cupertino/jquery-ui.min.css">
<script src="http://code.jquery.com/jquery-1.9.1.min.js"></script>
<script src="http://code.jquery.com/ui/1.10.3/jquery-ui.min.js"></script>
というのを、html の<head> 部分に書き加えます。

<link>要素の「cupertino」の部分は、テーマの名前を表しているの。テーマは、ウィジェットのスタイルを決めるためのスタイルシートと関連する画像リソースの集合のこと。標準で24のテーマが用意されています。

black-tie	blitzer	cupertino	dark-hive	dot-luv	eggplan
texcite-bike	flick	hot-sneaks	humanity	le-frog	mint-choc
overcast	pepper-grinder	redmond	smoothness	south-street	start
sunny	swanky-purse	trontastic	ui-darkness	ui-lightness	vader

＜日付選択ボックス＞

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>DatePicker ウィジェット</title>
    <link rel="stylesheet"
          href="http://code.jquery.com/ui/1.10.3/themes/cupertino/jquery-ui.min.css">
    <script src="http://code.jquery.com/jquery-1.10.2.min.js"></script>
    <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.min.js"></script>
    <script>
        $(function() { $('#date').datepicker({ dateFormat: 'yy/mm/dd' }); });
    </script>
  </head>
  <body>
      <input type="text" id="date" size="10" onchange=alert(this.value)>
  </body>
</html>
```



＜アコーディオンメニュー＞

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Accordion</title>
    <link rel="stylesheet"
          href="http://code.jquery.com/ui/1.9.2/themes/base/jquery-ui.css">
    <script src="http://code.jquery.com/jquery-1.8.3.js"></script>
    <script src="http://code.jquery.com/ui/1.9.2/jquery-ui.js"></script>
    <style>
      #accordion { padding: 10px; width: 350px; height: 200px; }
    </style>
    <script>
      $(function() { $("#accordion").accordion(); });
    </script>
  </head>
  <body>
    <div id="accordion">
      <h3>セクション 1</h3>
        <ul><li><a href="javascript:alert('1-1')">メニュー 1-1</a></li>
          <li><a href="javascript:alert('1-2')">メニュー 1-2</a></li>
          <li><a href="javascript:alert('1-3')">メニュー 1-3</a></li>
        </ul>
      <h3>セクション 2</h3>
        <ul><li><a href="javascript:alert('2-1')">メニュー 2-1</a></li>
          <li><a href="javascript:alert('2-2')">メニュー 2-2</a></li>
          <li><a href="javascript:alert('2-3')">メニュー 2-3</a></li>
        </ul>
      <h3>セクション 3</h3>
        <ul><li><a href="javascript:alert('3-1')">メニュー 3-1</a></li>
          <li><a href="javascript:alert('3-2')">メニュー 3-2</a></li>
          <li><a href="javascript:alert('3-3')">メニュー 3-3</a></li>
        </ul>
    </div>
  </body>
</html>
```