

jQuery と Ajax

目次

1. jQuery の準備	1
1 jQuery を使う準備	1
2 jQuery の基本	3
2. jQuery による DOM 操作	7
1 要素の操作	7
2 メソッドチェーン	9
3. jQuery のメソッド	10
1 イベント	10
2 フォーム機能	14
4. Ajax の概要	22
1 Ajax とは	22
2 Ajax と JavaScript ライブラリ	22
3 HTTP	24
4 jQuery による Ajax	25
5. Ajax の実装	28
1 Ajax とデータベース	28
2 Eclipse と jQuery	36

1. jQuery の準備

jQuery は最も広く利用されている JavaScript ライブラリの 1 つです。jQuery の最大の特徴は、JavaScript とは異なる手軽な書き方で JavaScript の命令を記述できることにあります。JavaScript の記述には多くの手続きを必要としますが、jQuery を利用すると非常に短いコードで済みます。また、CSS に近い記述方法を採用しているため、CSS が分かる Web デザイナーやエンジニアが手軽に習得できるよう設計されています。

jQuery は MIT ライセンスで公開されているため、ライブラリ中の著作権表示を消さなければ商用、非商用を問わず誰でも自由に利用することができます。また、jQuery は非常に軽量なライブラリとなっており、ファイルサイズは約 100KB 程しかありません。そのため、Web ページの読み込み速度にほとんど影響を与えることなく利用できるのも大きな特徴と言えるでしょう。

jQuery は次の URL から入手することができます。

<http://jquery.com/>

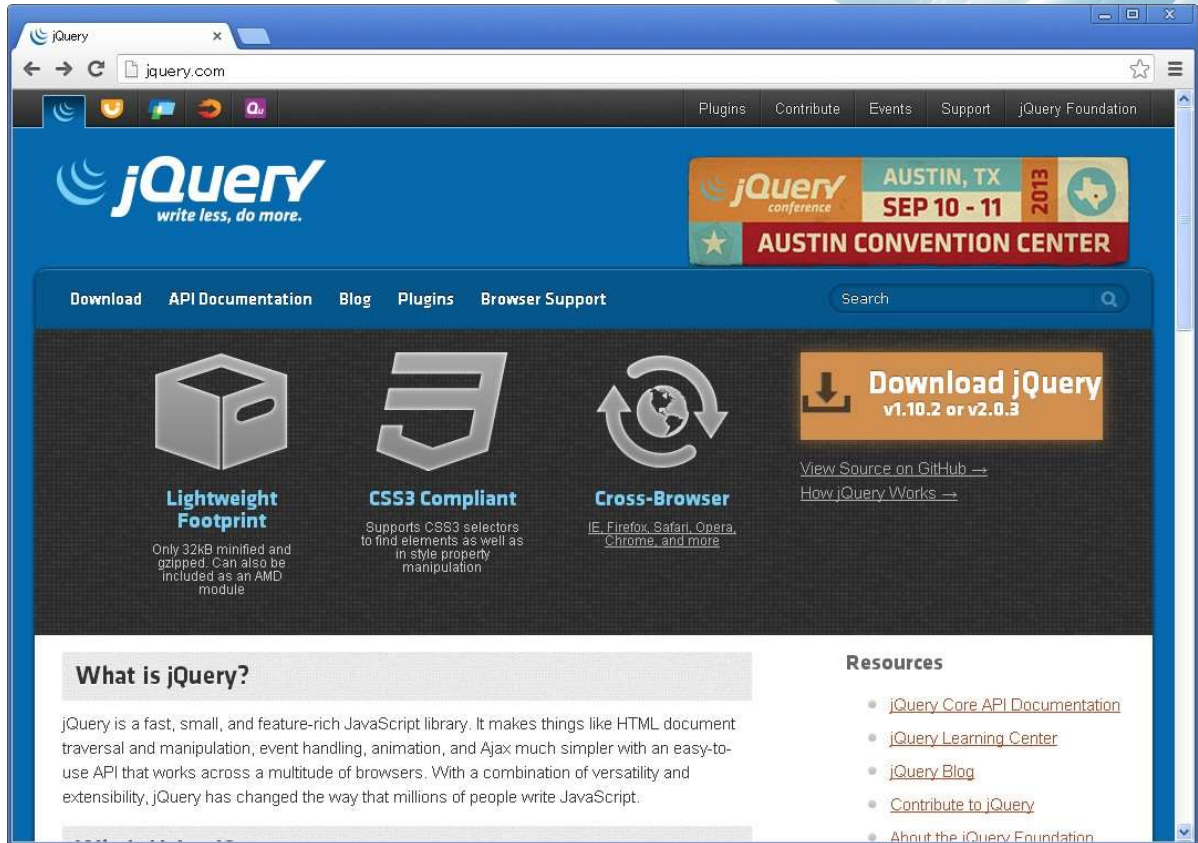
1 jQuery を使う準備

jQuery の Web サイトのトップページで「Download jQuery」ボタンをクリックすると、ダウンロードページが表示されます。ダウンロードページでは 2 種類のファイルをダウンロードできます。

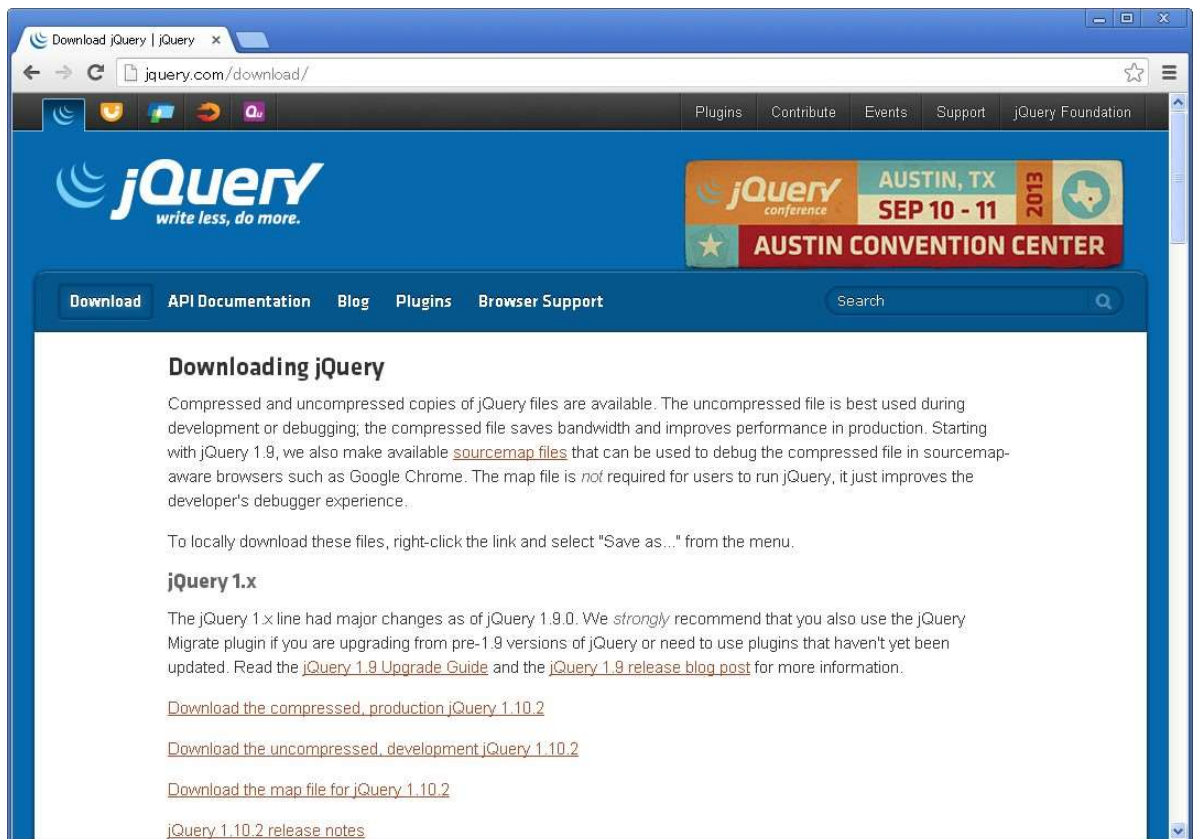
jQuery の配布物には以下の 2 種類のファイルが含まれています。

ファイル	ファイル名	説明
compressed	jquery-(バージョン番号).min.js	ファイルサイズを圧縮、最小化したもので、サイト運用時など転送量を減らす場合に使用する
uncompressed	jquery-(バージョン番号).js	ファイルサイズは大きいですが、開発時に jQuery のソースコードを参照、デバッグなどを行う場合に使用する

- 「Download the compressed, production jQuery 1.10.2」
ファイルサイズを圧縮するために最小化されたもの。
サイトの運用時に転送量を減らしたい場合に使用する。
- 「Download the uncompressed, development jQuery 1.10.2」
最小化されていないためファイルサイズは大きいですが、ソースは読みやすい。
開発時に jQuery のソースコードを参照し、デバッグを行う場合に使用する。



jQuery の Web サイトのトップページ



jQuery のダウンロードページ

ここでは uncompressed バージョンを使用します。「Download the uncompressed, development jQuery 1.10.2」のリンクを右クリックし、コンテキストメニューから「対象をファイルに保存」を選択して適当なフォルダに保存してください。

HTML ではダウンロードした JavaScript ファイルを次のように参照することで jQuery を使用することができます。

```
<script src="jquery-1.10.2.js"></script>
```

script 要素はページ内のどこに記述しても構いませんが、一般的には head 要素の内部に記述します。

```
<head>
  <meta charset="utf-8">
  <title>jQuery を読み込む</title>
  <script src="jquery-1.10.2.js"></script>
</head>
```

ダウンロードしたファイルを参照する代わりに、CDN(コンテンツデリバリーネットワーク)を使用することも可能です。

```
<script src="http://code.jquery.com/jquery-1.10.2.min.js"></script>
```

2 jQuery の基本

jQuery を含む JavaScript のコードは script 要素の中に記述する必要がありますが、Web ブラウザは HTML をファイルの先頭から 1 行ずつ読み込んでおり、script 要素が読み込まれた段階で、script 要素内記述された命令を実行します。以下 jQuery を使用していない簡単なサンプルです。

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="UTF-8">
    <title>sample</title>
    <script>
      alert("Hello world")
    </script>
  </head>
  <body>
    <h1>サンプルテキスト</h1>
  </body>
</html>
```


サンプルを実行すると「Hello world」というダイアログがブラウザ上に表示されます。「OK」ボタンを押すと body 要素に記述されているテキストが表示されますが、ボタンを押すまでは何も表示されません。「alert」という命令が実行された時点では、script 要素以降の HTML をブラウザは認識していないため、alert の命令を終了させた後にテキストが表示されるという仕組みになっています。

head 要素内の script 要素に body 要素内の HTML を操作するための命令を書いても、ブラウザがまだ script 要素までしか読み込んでいないので、本来意図した通りに動きません。

そこで登場するのが、スクリプトを実行するタイミングを制御する「ready」という jQuery のメソッドです。ready メソッドは以下のような形式で使います。

```
$(document).ready(function(){
    //jQuery の処理を記述
});
```

ready メソッドの内側(function(){...}内)に記述されている命令は、HTML の読み込みが終わった後に実行されます。ready という名前のとおり、「準備ができた段階でスクリプトを実行」という仕組みになっています。

また、ready メソッドは以下のような省略形を使用することができます。

```
$(function(){
    //jQuery の処理を記述
});
```

jQuery でスクリプトを記述するほとんどの場合、まず ready メソッドを記述し、その内側に実際の命令を記述していきます。jQuery の基本的な利用方法を学ぶために、jQuery を使用して簡単な DOM 操作を行うサンプルを見てみましょう。

jQuery サンプル.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>sample</title>
    <script src="jquery-1.10.2.min.js"></script> ❶
    <script>
      $(function(){ ❷
        $('#button').click(function(){ ❸
          var name = $('#name').val(); ❹
          alert('こんにちは、' + name + 'さん！');
        });
      });
    </script>
  </head>
```

```
<body>
  <h1>jQuery の基本</h1>
  お名前をどうぞ :
  <input type="text" id="name" />
  <input type="button" id="button" value="クリック" />
</body>
</html>
```

①では jQuery の JavaScript ファイルをインポートしています。

jQuery ではイベントハンドラの登録など HTML のロード後に実行される処理を、②のように `$(function(){...});` の中に記述します。

③のように `$()` 関数に CSS セレクタを指定することで要素を検索できます。また、`click()` などイベントに対応したメソッドでイベントハンドラを登録できます。ここでは `id` 属性が `button` の要素の `click` イベントにイベントハンドラを登録しています。

`$()` 関数で取得した jQuery オブジェクトには要素や DOM ツリーの操作を行なう際に便利な各種のメソッドが用意されています。ここでは `val()` メソッドでテキストフィールドの値を取得しています(④)。

jQuery が提供する主な機能について以下にまとめます。

●CSS セレクタ

通常の JavaScript で要素を取得するには、DOM ツリーを 1 つずつたどるか、`id` 属性の値を指定して `document.getElementById()` メソッドを使用などの方法しかありませんが、jQuery では `$()` 関数の引数に CSS セレクタを指定することで非常に柔軟な検索を行なうことができます。

```
// id 属性が myelement 配下の div 要素を取得
var element=$('#myelement div');
```

●DOM 操作

jQuery オブジェクトのメソッドを使用することで、煩雑な DOM 操作を手軽に行なうことができます。たとえば次のコードは `class` 属性が `image` の `li` 要素の子要素の先頭に `img` 要素を挿入しています。このように jQuery を使用することで、CSS セレクタに一致した複数の要素に対する操作を一度に行なうことができます。

以下に jQuery を使用した DOM 操作の例を示します。

```
// <li class="icon">...</li> の先頭に  を挿入
$('li.icon').prepend($('<img>').attr('src', 'icon.png'));
```

```
<ul>
  <li class="icon">HTML</li>
  <li class="icon">CSS</li>
  <li class="icon">JavaScript</li>
</ul>
```



```
<ul>
  <li class="icon">HTML</li>
  <li class="icon">CSS</li>
  <li class="icon">JavaScript</li>
</ul>
```

先頭に img 要素を挿入

● イベントハンドラの設定

JavaScript でイベント処理を行なう場合、ブラウザごとのイベントモデルの違いやイベントハンドラに渡されるイベントオブジェクトの差異に留意する必要があります。jQuery を使用することで、ブラウザ間のイベントモデルの差異を吸収し、ブラウザの違いを意識することなくイベントを処理することができます。

● その他の機能

jQuery にはその他にも、Ajax による通信を簡単に行なうための機能や、配列の操作などを手軽に行なうための便利なユーティリティが含まれています。

2. jQuery による DOM 操作

1 要素の操作

DOM 要素の操作を行うためのメソッドには次のようなものがあります

メソッド名/構文	説明
after(content)	引数で指定したコンテンツ(content)を要素の直後に追加する
before(content)	引数で指定したコンテンツ(content)を要素の直前に追加する
insertAfter(target)	引数に指定した要素(target)の直後にこのコンテンツを追加する
insertBefore(target)	引数に指定した要素(target)の直前にこのコンテンツを追加する
append(content)	引数に指定したコンテンツ(content)を要素の末尾に追加する
appendTo(target)	引数に指定した要素(target)の末尾にこのコンテンツを追加する
prepend(content)	引数で指定したコンテンツ(content)を要素の先頭に追加する
prependTo(target)	引数に指定した要素(target)の先頭にこのコンテンツを追加する
remove([selector])	要素を削除する引数にセレクタ(selector)を指定した場合はこの jQuery オブジェクトに含まれている要素の中からセレクタに一致した要素が削除される
detach([selector])	remove()メソッドと同じだが、要素に設定されているイベントハンドラやデータは削除しない。いったん削除した要素を再度 DOM に追加するときに便利
empty()	要素内のすべての子ノードを削除する
attr(name[.value])	属性(name)を取得または設定する
attr(hash)	複数の属性(hash)をまとめて設定する
removeAttr(name)	属性(name)を削除する
replaceWith(content)	引数にしていたコンテンツでこの要素を置換する
replaceAll(target)	引数で指定した要素(target)をこのコンテンツで置換する
wrap(parent)	引数で指定した要素(parent)でラップする
unwrap()	新要素を削除する
wrapAll(parent)	選択している要素(parent)をまとめて引数で指定した要素をラップする
wrapInner(parent)	子要素を引数で指定した要素(parent)でラップする
html([content])	要素内の HTML コンテンツ(content)も取得または設定する
text([content])	要素内のテキストコンテンツ(content)も取得または設定する
clone()	要素をコピーして新しいオブジェクトを作成する
css(name[.value])	指定した CSS プロパティの値を設定・取得する

●属性の追加/削除

attr()メソッドはJavaScriptでいうところのsetAttribute()メソッドとgetAttribute()メソッドの複合形であり、要素の属性値の取得および設定、removeAttr()メソッドで属性の削除を行うことができます。attr()メソッドの引数にオブジェクトを指定すると、複数の属性をまとめて設定することができます。


```
//a 要素の href 属性を取得
var url = $('a').attr('href');
//a 要素の href 属性を設定
$('a').attr('href', 'http://www.google.co.jp/');
//複数の属性をまとめて設定
$('a').attr({
  title: 'Google',
  href: 'http://www.google.co.jp/'
});
```

●要素内のコンテンツの取得/設定

text()メソッドは要素内のテキスト、html()メソッドは要素内の HTML を取得および設定できます。引数なしで呼び出すとコンテンツを取得し、引数を指定して呼び出すと引数に指定したコンテンツを設定します。

要素内のコンテンツの取得と設定.html

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="utf-8">
    <title>sample</title>
    <script src="jquery-1.10.2.min.js"></script>
  </head>
  <body>
    <h1>元々のテキスト</h1>
    <script>
      //h1 要素内のテキストを設定
      $('h1').text('Hello, jQuery!');
      //h1 要素内のテキストを取得
      var text = $('h1').text(); //→'Hello, jQuery!'
      alert(text);
      //h1 要素内の HTML を設定
      $('h1').html('<a href="http://jquery.com/">Hello, jQuery!</a>');
      //h1 要素内の HTML を取得
      var html = $('h1').html(); //→ '<a href="http://jquery.com/">
        Hello, jQuery!</a>'
      alert(html);
    </script>
  </body>
</html>
```

2 メソッドチェーン

ここまで HTML と CSS を操作する命令を一気に紹介してきました。jQuery では DOM 操作するときに便利な「メソッドチェーン」という記述を利用することができます。以下の例は prepend() メソッドと append() メソッドを利用し、p 要素の先頭と最後に sample 要素を挿入するサンプルです。

```
$("#p").prepend("<sample>先頭に挿入</sample>");  
$("#p").append("<sample>後方に挿入</sample>");
```

\$("#p") という同じセレクトアに対して、prepend() と append() の 2 つの命令が設定されています。このスクリプトをブラウザで実行すると、jQuery は p 要素を探し、prepend() を実行し、次にまた p 要素を探し、append() を実行します。p 要素を 2 回探しにいくことになるので非効率的です。

上記の内容を、メソッドチェーンを利用して書き直すと以下のようになります。

```
$("#p").prepend("<sample>先頭に挿入</sample>")  
    .append("<sample>後方に挿入</sample>");
```

コードは少し短くなりましたが、実行してみると結果は前と同じです。このように、jQuery では命令と命令を.(ドット)でつなげて記述することで、1 つのセレクトアに対して 2 つ以上の命令を連続して実行できます。この記述方法をメソッドチェーンと呼びます。単に記述が短くなるだけではなく、今回の場合は p 要素を一度検索した後に、各メソッドが実行されるので内部的な処理も効率が良くなります。同一のセレクトアに対して複数の命令を指定する場合は、メソッドチェーンで記述するようにしてください。

3. jQuery のメソッド

1 イベント

jQuery ではプログラムが実行されるきっかけを「イベント」と言います。イベントの発生を感知したタイミングで処理が実行される仕組みです。イベントを感知して処理を実行するにはいくつかの命令を使用します。主に次のようなものがあります。

メソッド名/構文	説明
click()	クリック時に処理を実行する
dblclick()	ダブルクリック時に処理を実行する
mousedown()	マウスのボタンが押された時に処理を実行する
mouseup()	マウスのボタンが離された時に処理を実行する
mouseover()	マウスオーバー時に処理を実行する
mouseout()	マウスアウト時に処理を実行する
mousemove()	マウスが移動したときに処理を実行する
one()	イベント発生時に一度だけ処理を実行する
on()	対象要素を絞ってイベントを登録する
off()	設定されているイベント処理を取り消す
hover()	マウスオーバー/マウスアウト時に処理を実行する

●特定の要素がクリックされたときに処理を実行

click()メソッドを使用することで特定の要素がクリックされた時に処理を実行することができます。dblclick メソッドを使用した場合は特定の要素が「ダブルクリック」された時に処理が実行されます。以下サンプルになります。

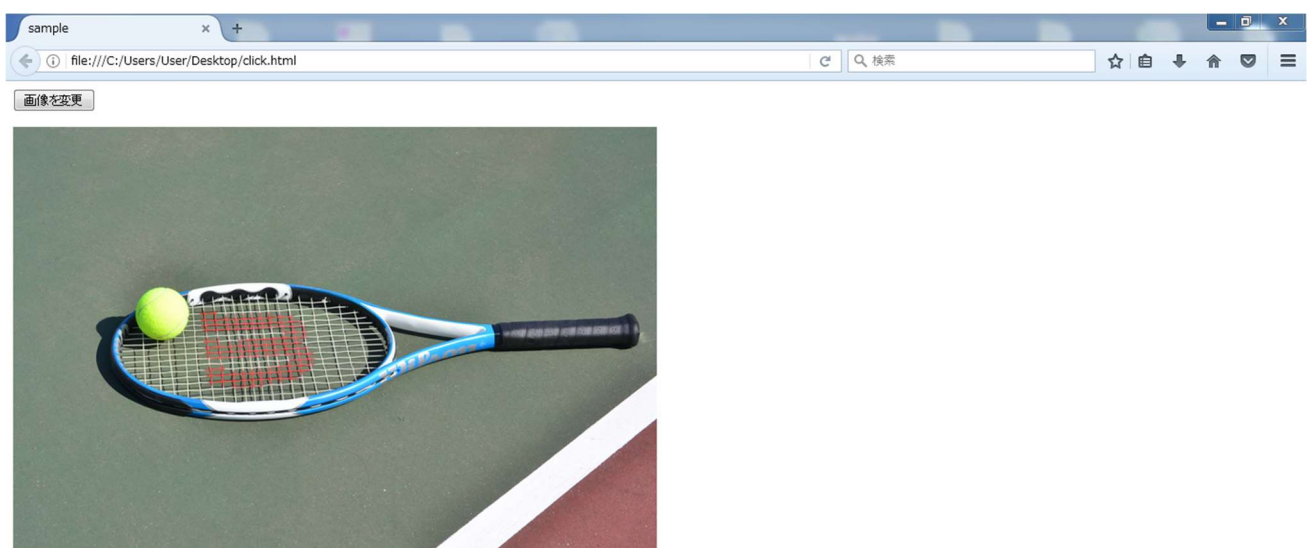
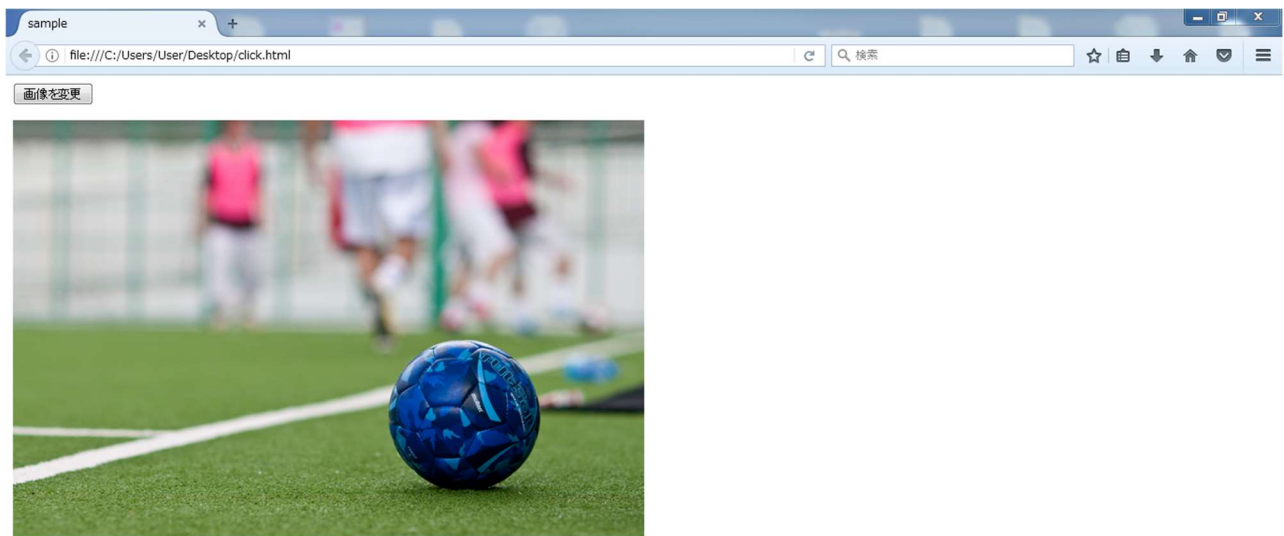
特定の要素がクリックされたときに処理を実行.html

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="UTF-8">
    <title>sample</title>
    <script src="jquery-1.10.2.min.js"></script>
    <script>
      $(function(){
        $("button").click(function(){
          $("img").attr("src","tennis.jpg").attr("alt","テニス");
        });
      });
    </script>
  </head>
  <body>
    <button>画像を変更</button>
```

```
<p></p>
</body>
</html>
```

上記サンプルはbutton要素がクリックされたらimg要素のsrc属性の値を「tennis.jpg」に、Alt属性の値を「テニス」に書き換えるスクリプトです。attr()メソッドはカンマ区切りで指定した属性値を変更する命令です。メソッドチェーンを利用し、src属性とalt属性を一度に書き換えています。また、あくまでsrc属性とalt属性を書き換えただけなので、style属性は書き換えていません。画像サイズが変わらないことにも注目してみてください。

実行結果



●複数の要素の取得

複数の要素に対してイベント設定する場合もあるでしょう。下記のサンプルは複数の a 要素に対してそれぞれ異なるクリックイベントを設定しています。複数の要素を指定した順番に取得するには「eq(equal)セレクト」を使用します。JavaScript は 0 から数値を数えるため、1 番目の要素は「0」二番目の要素は「1」になります。

複数の要素の取得.html

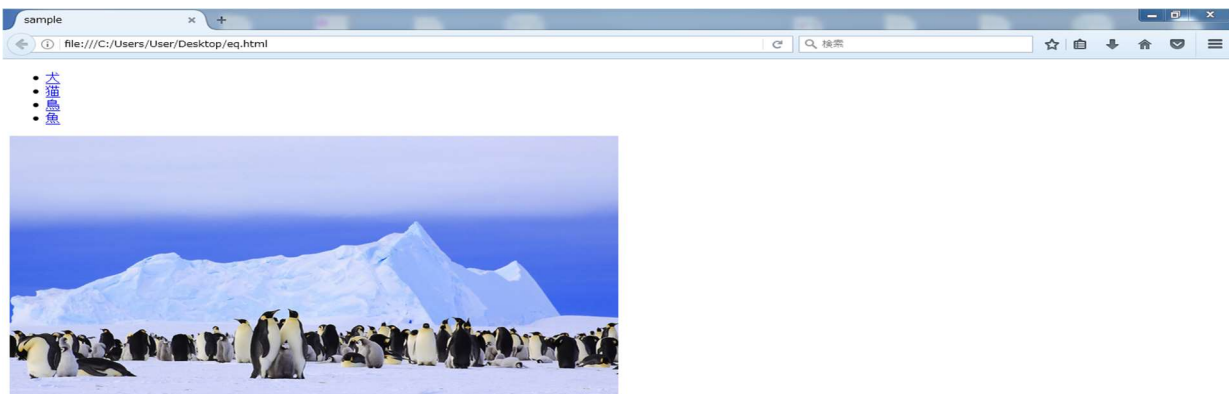
```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="UTF-8">
    <title>sample</title>
    <script src="jquery-1.10.2.min.js"></script>
    <script>
      $(function(){
        $("a:eq(0)").click(function(){
          $("img").attr("src","dog.jpg").attr("alt","犬");
          return false;
        });
        $("a:eq(1)").click(function(){
          $("img").attr("src","cat.jpg").attr("alt","猫");
          return false;
        });
        $("a:eq(2)").click(function(){
          $("img").attr("src","penguin.jpg").attr("alt","鳥");
          return false;
        });
        $("a:eq(3)").click(function(){
          $("img").attr("src","fish.jpg").attr("alt","魚");
          return false;
        });
      });
    </script>
  </head>
  <body>
    <ul>
      <li><a href="dog.jpg">犬</a></li>
      <li><a href="cat.jpg">猫</a></li>
      <li><a href="penguin.jpg">鳥</a></li>
      <li><a href="fish.jpg">魚</a></li>
    </ul>
    <p></p>
```



```
</body>
</html>
```

上記サンプルは click() メソッドで対象としているセレクトが a 要素ですが、もともと a 要素は「クリックされると href 属性に書かれたリンク先に移動する」働きがあります。そのままでは画面遷移をしてしまうので、click イベントの最後に「return false;」を追加します。この場合 a 要素がクリックされた際には jQuery の命令だけが実行され、href 属性に設定されたリンク先には移動しません。

実行結果



● イベントが発生した要素を取得する

画像を変更する機能は実装できましたが、このままだとコードがかなり長くなってしまい、画像が増えるたびにコードを追記しなくてはなりません。それを解消するために、イベントが発生した要素を取得するセレクト「\$(this)」が用意されています。イベントを設定している click(function(){・・・}) 内で \$(this) と書くと、イベントが発生した要素を取得できます。\$(this) を使用すると、先程のサンプルの jQuery 文を以下のように短く記述できます。

```
<script>
$(function(){
  $("a").click(function(){
    $("img").attr("src",$(this).attr("href")).attr("alt",$(this).text());
    return false;
  });
});
</script>
```

クリックされた要素の href 属性を \$(this).attr("href") で、クリックされた要素内のテキストを \$(this).text() で取得し、img 要素の src 属性、alt 属性にそれぞれ設定しています。仮に画像が 10 枚増えても、jQuery を書き換える必要がなくなり、HTML の修正だけで済みます。

イベントが発生した要素を取得する \$(this) は click イベント以外でも使用する機会がとても多いので覚えておきましょう。

2 フォーム機能

jQuery のフォーム機能を使用すると、入力内容に応じてフォームのデザインを変更したり、エラーメッセージを表示したりでき、使い勝手のいいフォームを作成できます。フォームの機能には主に次のようなものがあります。

メソッド名/構文	説明
val()	フォームに入力または選択されている値を取得・変更する
focus()	マウスやタブキーによって選択された状態を感知する
blur()	選択が外れたことを感知する
change()	フォームの内容変更を感知する
submit()	フォームの送信を感知する
checked()	チェックの入っているフォーム部品を選択する(チェックボックス)
selected()	選択されている要素を選択する(セレクトボックス)

●フォーム選択時に動的に操作を行う

focus()メソッドを使用し、input 要素などのフォーム部品がマウスやタブキーによって選択された状態(フォーカス状態)になったことを感知し、設定された処理を実行します。他のイベントと同じように、括弧内に function(){…} を記述し、その中に実行したい処理を記述します。また、フォーカスが外れたことを感知する blur()メソッド使用した例を紹介します。以下サンプルになります。

フォーム選択時に動的に操作を行う.html

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="utf-8">
    <title>sample</title>
    <script src="jquery-1.10.2.min.js"></script>
    <script>
      $(function(){
        //初期設定 背景を灰色に、val()メソッドを使い初期値を設定、文字色を黒に
        $('input[type="text"]').val("入力してください").css('color','#000000')
        .css('background','#CCC').focus(function() {
          //フォーカスされたら val()メソッドで初期化し、文字色を白に、背景色を赤にする
          $(this).val('').css('color','#ffaafa').css('background','#dc143c');
        }).blur(function() {
          //フォーカスが外れたら文字色を黒に背景色を青にする
          $(this).css('color','#000000').css('background','#6495ed');
        });
      });
    </script>
  </head>
```

```
<body>
  苗字 : <input type="text">
  名前 : <input type="text">
</body>
</html>
```

上記サンプルはメソッドチェーンを利用し、テキストボックスの入力欄に値と、文字色、背景色の三つを設定しています。テキストボックスを選択すると focus()メソッドが実行され、入力欄が変更されます。フォーカスを外すと blur()メソッドが実行されます。各イベントを組み合わせることで、様々な処理を動的に実装することができます。

実行結果



●選択されている要素を使用不可に変更する

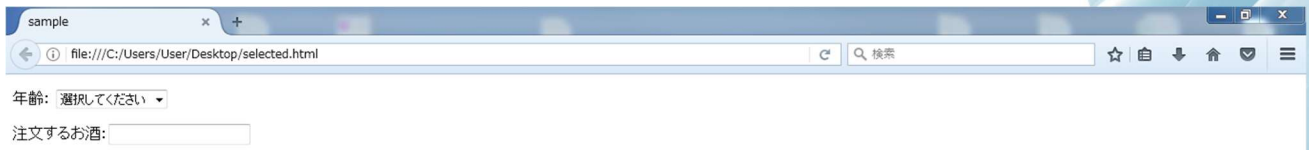
selected()メソッドを使用し、選択されているセレクトボックスの要素を取得し、value 属性の値によって処理を変更したい場合があります。以下サンプルになります。

選択されている要素を使用不可に変更する.html

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="utf-8">
    <title>sample</title>
    <script src="jquery-1.10.2.min.js"></script>
    <script>
      $(function(){
        $("select").change(function(){
          if($(":selected").attr("value")==="10"){
            $("input").val("お酒は 20 歳から !").attr("disabled", "disabled");
          }else{
            $("input").removeAttr("disabled");
          }
        });
      });
    </script>
  </head>
  <body>
    <p>年齢 :
    <select name="age">
      <option value="">選択してください</option>
      <option value="10">10 代</option>
      <option value="20">20 代</option>
      <option value="30">30 代</option>
      <option value="40">40 代以上</option>
    </select></p>
    <p>注文するお酒 : <input type="text"></p>
  </body>
</html>
```

上記サンプルは change()メソッドを使用し、セレクトボックスの値が変わった時に実行される文です。もし 10 代が選択された場合は value="10"なので、if 文の中身が実行されます。入力欄が選択不可になり、値が変更されます。10 代以外を選択した場合はフォームの入力ができるようになります。

実行結果

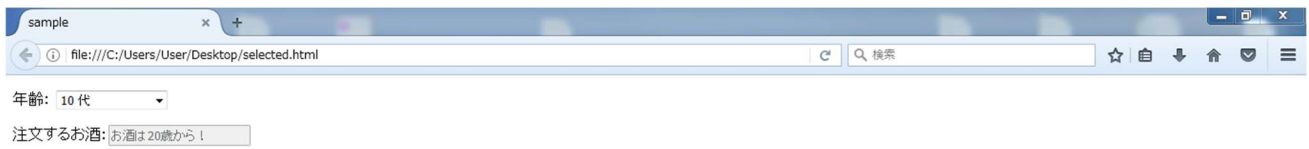


sample

file:///C:/Users/User/Desktop/selected.html

年齢: 選択してください

注文するお酒:



sample

file:///C:/Users/User/Desktop/selected.html

年齢: 10代

注文するお酒: お酒は20歳から！

●jQuery を用いたバリデーション機能

フォームに入力された内容に漏れやミスがないかチェックを行うバリデーション機能を実装すると、ユーザーは格段に使いやすくなります。エラー箇所を分かりやすく示して、使い勝手のいいページにすることができます。注意点としては、あくまでクライアント側で実行する簡易的な入力チェックです。セキュリティを確保するにはサーバ側での入力チェックが別途必要になりますので気を付けてください。

以下サンプルになります。

jQuery を用いたバリデーション機能.html

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="utf-8">
    <title>sample</title>
    <script src="jquery-1.10.2.min.js"></script>
    <link rel="stylesheet" href="style.css">
    <script>
      $(function(){
        $("form").submit(function(){
          //エラーの初期化
          $(".error").remove();
          $("dl dd").removeClass("error");
          $("input[type='text'].validate").each(function(){
            //必須項目のチェック
            if($(this).hasClass("required")){
              if($(this).val()==""){
                $(this).parent().prepend("<p class='error'>必須項目です</p>");
              }
            }
            //数値のチェック
            if($(this).hasClass("number")){
              if(isNaN($(this).val())){
                $(this).parent().prepend("<p class='error'>数値のみ入力可能です</p>");
              }
            }
          });
          //エラーの際の処理
          if($(".error").length > 0){
            $(".error").parent().addClass("error");
            return false;
          }
        });
      });
    </script>
  </head>
  <body>
    <div id="header">
      <h1>jQueryでバリデーション</h1>
    </div>
    <div id="main">
      <div id="form">
        <div id="comment">
          <input type="text" value="" />
          <br/>
          <input type="button" value="コメントする"/>
        </div>
        <div id="password">
          <input type="password" value="" />
          <br/>
          <input type="button" value="パスワードを登録する"/>
        </div>
      </div>
    </div>
    <div id="footer">
      <p>Copyright © 2014 All Rights Reserved.</p>
    </div>
  </body>
</html>
```

```
</script>
</head>
<body>
  <div id="container">
    <h1>入力フォーム</h1>
    <form action="complete.html" method="post">
      <dl>
        <dt>名前<span>※</span></dt>
        <dd><input type="text" name="name" class="validate required"></dd>
        <dt>郵便番号</dt>
        <dd><input type="text" name="zip1" class="validate number"> - <input
          type="text" name="zip2" class="validate number"></dd>
      </dl>
      <p><input type="submit" value="送信"></p>
    </form>
  </div>
</body>
</html>
```

style.css

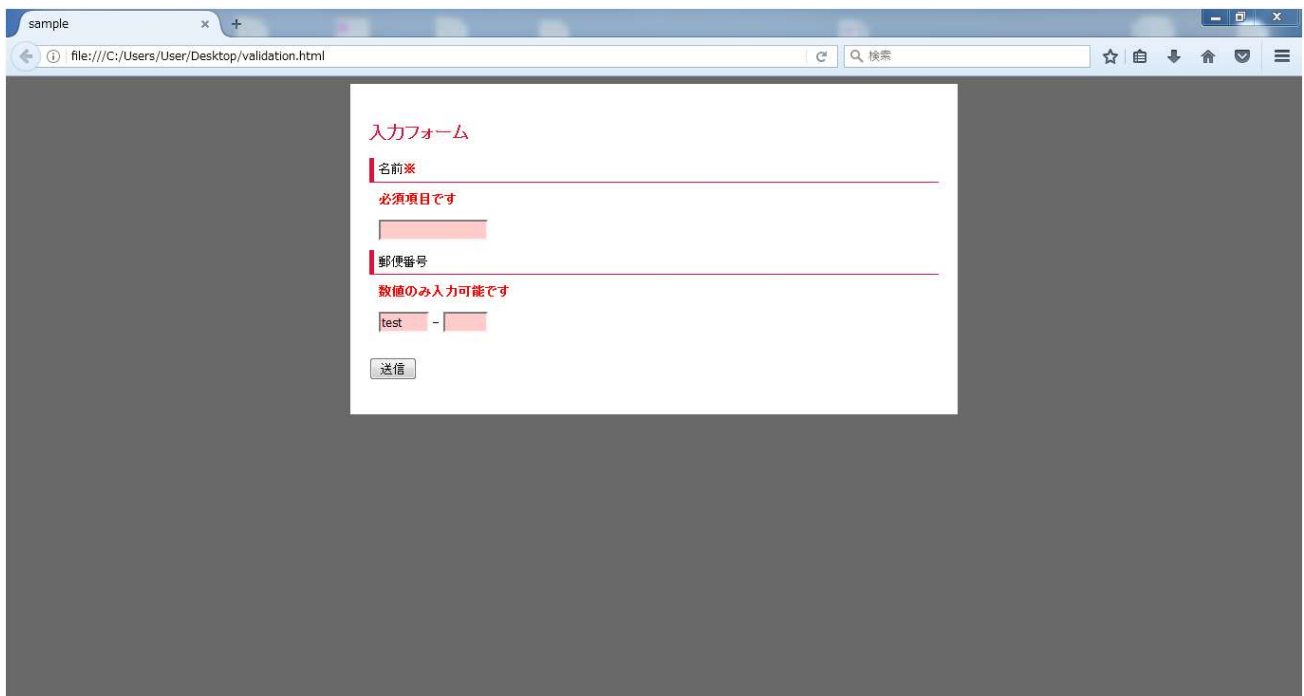
```
body{
  background:#696969;
}
#container{
  width:600px;
  margin:0 auto;
  padding:20px;
  background:white;
}
h1{
  margin-top:20px;
  font-size:large;
  color:#dc143c;
}
dl dt {
  border-left:5px solid #dc143c;
  border-bottom:1px solid #dc143c;
  font-size:small;
  margin:0;
  padding:5px;
}
dl dt span{
  color:red;
```

```
font-weight:bold;
}
dl dd{
  font-size:small;
  margin:0;
  padding:10px;
}
dl dd input{
  position:relative;
  z-index:2;
}
dl dd label{
  position:relative;
  padding:5px 5px 5px 25px;
  margin : 0 5px 0 -25px;
  margin-left:-25px;
  position:relative;
  z-index:1;
}
dl dd.error input , dl dd.error label {
  background:#FFCCCC;
}
* html dl dd.error label {
  background:none;
}
*+html dl dd.error label{
  background:none;
}
dl dd p.error{
  margin:0;
  color:red;
  font-weight:bold;
  margin-bottom:1em;
}
```

バリデーション機能としては、必須項目の名前の入力チェックを行い、郵便番号では入力されている値が数値かどうかチェックを行っています。HTML は、フォーム全体を dl/dt/dd 要素で記述し、dt 要素に「名前」などの項目名を、dd 要素に input 要素などのフォーム部品を配置しています。各フォーム部品(input 要素)にはバリデーションの条件を示す「validate、required」などの class 属性を付けています。CSS ではフォーム部品や項目名の基本スタイルとともに、バリデーションエラー用のスタイルも定義しています。バリデーションエラー時に dd 要素に class 属性「error」を追加します。エラーが発生した際は、メッセージを赤色で表示し、背景色をピンクに設定しています。

今回のサンプルは全て submit イベントがトリガーとなっています。送信ボタンがクリックされた際に、入力チェックが実行される仕組みです。また重要な点としましては、submit イベント内の処理では最初に、エラーを初期化します。送信ボタンが初めて押されたときはいいのですが、すでにボタンが押されてバリデーション処理が実行済みの場合、エラー時の処理が適用されている場合があります。初期化をしないとエラーメッセージが何度も追加されたり、修正済みの項目のエラーメッセージが消えなかったりしますので、removeClass()メソッドを使用し、「error」という class 属性を取り除き、remove()メソッドでエラーメッセージ(p 要素)の削除をします。

実行結果



The screenshot shows a web browser window with a single tab titled 'sample'. The address bar displays the file path: file:///C:/Users/User/Desktop/validation.html. The page content is a form titled '入力フォーム' (Input Form). It contains three input fields, each with a red error message above it:

- The first field is labeled '名前※' (Name ※) and has the error message '必須項目です' (Required item).
- The second field is labeled '郵便番号' (Postal Code) and has the error message '数値のみ入力可能です' (Only numerical input is possible).
- The third field contains the text 'test' and has a red error message, though the text is partially obscured.

At the bottom of the form is a button labeled '送信' (Send).

4. Ajax の概要

Ajax(エイジャックス)の登場は、一時期人気が衰えかかっていた JavaScript が再び脚光を浴びるきっかけとなりました。Ajax は新しい技術を使わずに既存の技術を組み合わせるだけで新しいサービスを生み出したという点でも非常に画期的でした。本資料では Ajax で用いられている技術と、Ajax による非同期処理について説明します。

Ajax を利用したプログラムについて説明する前に、Ajax の生い立ちと背景となる技術を見ていきましょう。Ajax を構成する技術は、いずれも Ajax のために開発されたものではなく、別の用途に使われていた技術を寄せ集めたものです。そのため、統一感がなく使いにくい場合もありますが、1 つ 1 つの技術をおさえていくことで、理解がしやすくなるでしょう。

1 Ajax とは

Ajax とは「Asynchronous JavaScript and XML」の略称で、JavaScript と XML の技術を使って非同期で通信を行なう開発手法のことを指します。Ajax を使うとページ遷移を行わずに Web ページの一部だけを更新できます。たとえば、Google Maps ではマウス操作で地図をどこまでもスクロールできます。また、Google 検索では入力候補のキーワードがリアルタイムに表示されます。

Ajax がこれほど注目されたのは、Web アプリケーションが苦手とされてきた操作性の悪さを、JavaScript を積極的に用いることで克服した点にあります。Ajax は「非同期通信を使用して Web ページの一部を置き換える」という技術によって、Web ページ全体を書き換えなくてはならないという Web アプリケーションの限界を打ち破り、従来のクライアント/サーバアプリケーションのような画面遷移が少ない操作性の高いアプリケーションの作成を可能としました。

2 Ajax と JavaScript ライブラリ

Ajax を利用した Web 画面は、JavaScript を多用するため、コード量が多くなる傾向にあります。そのため、JavaScript のプログラムで煩雑になりがちなクラスの定義や DOM の操作などの処理を簡単に行うための JavaScript ライブラリがいくつか提供されています。主なものとして、jQuery や Dojo、Prototype.js などがあり、これらのライブラリが提供するオブジェクトやメソッドを利用することにより、プログラムのコード量を削減できます。

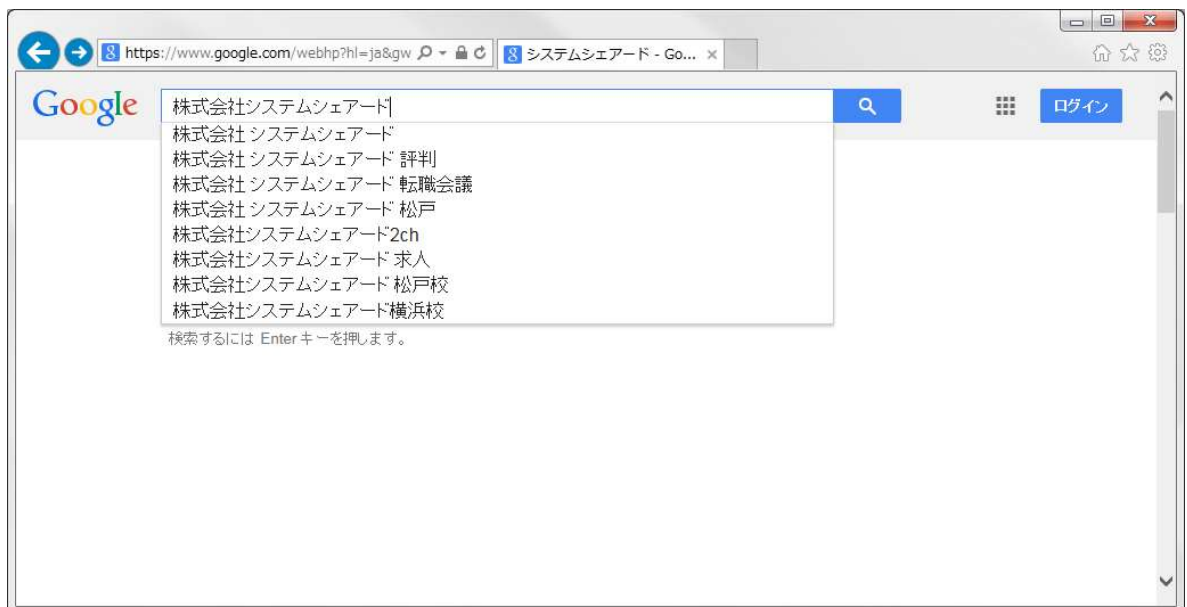
さらに、これらの JavaScript ライブラリの中には、ツールチップやタブページ、カレンダーによる日付入力などのクライアントアプリケーションに劣らない豊かな UI(画面部品)を JavaScript で実現しているものもあります。たとえば、jQuery UI のように表現性豊かな画面部品に強みを持ったライブラリもあり、JavaScript ライブラリを使うことで、豊かな表現力をもった Web アプリケーションが簡単に作成できるようになりました。現在では、これらのライブラリによって実現されている技術を総称して「Ajax」と呼ぶこともあります。

Ajax の有名な利用例としては、Google が提供する各種サービスが挙げられるでしょう。たとえば、地図サービスで有名な Google Maps では、ページ遷移せずに地図の移動や拡大縮小などを実現しています。



Google Maps (<http://maps.google.co.jp/>)

また、Google 検索では、Web ページを検索する際、検索語の候補をリアルタイムに表示し、入力補助を実現しています。



Google 検索での入力補助(<http://www.google.co.jp/>)

いずれも Web ブラウザさえあれば利用できるサービスであり大きな反響を呼びました。

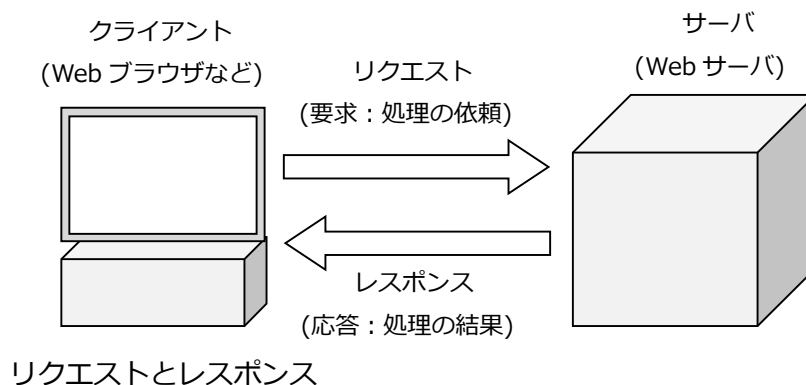
3 HTTP

Ajax で利用されている技術に、HTTP(HyperText Transfer Protocol)があります。HTTP は、Web ブラウザと Web サーバの間で HTML などのデータを送受信するための通信プロトコルで、Web アプリケーションにおける通信では一般的に用いられています。

HTTP は Java など Web アプリケーションの開発言語の多くでサポートされていますが、ほとんどの言語では HTTP の仕様が隠蔽された形でサポートされています。そのため、それらの言語を使用する場合は HTTP の仕組みを理解していなくてもかまいません。それに対して、Ajax で利用する XMLHttpRequest オブジェクトでは HTTP コマンドや HTTP のフォーマット(リクエストやレスポンスのデータ形式)を隠蔽せずにそのままの形式に近い形で扱うのが特徴です。したがって、jQuery などのライブラリを使用せずに Ajax を使用する場合は HTTP の知識が必要となります。

リクエストとレスポンス

HTTP はクライアント(Web ブラウザ)からサーバ(Web サーバ)に対して要求を送信し、その要求にサーバが応答するというシンプルな仕組みで構成されます。クライアントからの要求のことを「リクエスト」、サーバからの応答のことを「レスポンス」と呼びます(図 1)。1 つの処理は、必ず 1 回のリクエストと 1 回のレスポンスで構成され、レスポンスが返ってきた段階で処理が完結します。HTTP では 1 つのクライアントとの間の通信を維持できないため、ログイン状態を保持する場合など、複数のリクエストの関連を持ち続ける場合はサーバ側から何らかの仕組みを作り込む必要があります。



4 jQuery による Ajax

jQuery は Ajax のためのユーティリティとして下記のメソッドを提供しています。これらのメソッドにより手軽に Ajax を使用することができます。

jQuery が提供する Ajax 用のユーティリティメソッド

メソッド名/構文	説明
load(url[,data][,callback])	jQuery オブジェクトが選択している要素に Ajax で取得した HTML を挿入する
getScript(url[,callback])	JavaScript を取得して実行する
getJSON(url[,data][,callback])	JSON を取得する
get(url[,data][,callback])	GET リクエストを送信する
post(url[,data][,callback])	POST リクエストを送信する
ajax(settings)	jQuery が提供する Ajax 用のユーティリティのベースとなるメソッド。他のメソッドでは不可能な細かい処理が必要な場合に使用する

jQuery が提供する Ajax 用のユーティリティメソッドは、最終的に \$.ajax() メソッドを呼び出します。つまりこのメソッドが jQuery による Ajax サポートの基礎になっているということです。

通常、\$.ajax() メソッドを使用するよりも後述するユーティリティメソッドを使用するほうが簡単に処理を記述できますが、これらのユーティリティメソッドでは実現できない処理を行う必要がある場合は直接 \$.ajax() メソッドを使用します。

次に \$.ajax() メソッドの簡単な使用例を示します。なお、動作確認を行う場合は、「\$.ajax() メソッドを使用した Ajax の例.html」、「books.txt」を Eclipse の動的 Web プロジェクトの同階層に配備してから実行してください。

\$.ajax() メソッドを使用した Ajax の例.html

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="utf-8">
    <title>sample</title>
    <script src="jquery-1.10.2.min.js"></script>
    <script>
      //books.html を GET リクエストで取得する
      $.ajax({
        url: 'books.txt' ,
        type:'GET' ,
      }).done(function(data, status, req){
        // 成功したときに呼び出される
        alert(data);
      }).fail(function(req, status, error){
        // 失敗したときに呼び出される
```

```
        alert('失敗: ' + error);
    });
</script>
</head>
<body>
</body>
</html>
```

books.txt

This is pen!



実行結果



処理が成功した場合



処理が失敗した場合

一見警告ダイアログが表示されただけのように思えますが、実際には url に指定したリソースに対し、非同期通信を行っています。今回はプレーンテキストである books.txt に対しリクエストを送信しているため、固定文字列のみがレスポンスとして返されますが、servlet などを指定することも可能となっています。

なお、\$.ajax()メソッドの引数に指定可能なパラメータには次のようなものがあります。

\$.ajax()メソッドに指定可能な主なパラメータ

パラメータ名	説明
url	通信先の URL を指定する
async	非同期で通信を行うかどうかを true または false で指定する。false を指定した場合、通信が完了するまで処理をブロックする。省略する場合は true になる
cache	GET リクエストのレスポンスをキャッシュするかどうかを true または false で指定する。省略した場合は true になる。ただし、dataType に script または jsonp を指定した場合は false になる
contentType	リクエストのコンテンツタイプを指定する。省略した場合は application/x-www-form-urlencoded になる
context	コールバック関数内で this が示すオブジェクトを指定する
data	送信するパラメータをクエリ文字形式の文字列またはハッシュで指定する。
dataType	サーバから返却されるデータの型を次のいずれかから指定する。省略した場合は自動的に判別される xml html script json jsonp text
global	Ajax に関するイベントを発行するかどうかを true または false で指定する。デフォルトは true になる
ifModified	if-Modified-Since ヘッダと Last-Modified ヘッダを使用して更新チェックを行なうかどうかを true または false で指定する。省略した場合は false になる
jsonp	JSONP のコールバック名が callback 以外の場合にコールバック名を指定する
jsonpCallback	JSONP のコールバック関数名を指定する
password	HTTP 認証が必要な場合のパスワードを指定する
processData	data にオブジェクトを指定した場合にクエリ文字列形式に変換して送信するかどうかを true または false で指定する。省略した場合は true になる
scriptCharset	Ajax で取得したスクリプトを読み込む際の文字コードを指定する。dataType が json または jsonp で、かつ HTML と取得するスクリプトの文字コードが異なる場合のみ指定する必要がある
timeout	通信のタイムライン時間をミリ秒で指定する
type	GET または POST を指定する。省略した場合は GET になる
username	HTTP 認証が必要な場合のユーザー名を指定する

5. Ajax の実装

これまでに学習した Servlet/JSP を併用し、実際に非同期通信を行ってみましょう。Ajax では、サーバに対してパラメータ等の情報を Ajax リクエストとして送信し、ブラウザ等のクライアントは Ajax レスポンスとして、リクエストの結果を受け取り、受け取った Ajax レスポンスのデータを元にページに変更を加えます。

1 Ajax とデータベース

Ajax を利用し、データベースと接続し、画面上に非同期通信で値を表示するサンプルを紹介します。今までの Servlet/JSP と同様に Eclipse の動的 Web プロジェクトで作成していきます。(サンプルではプロジェクト名を AjaxSample としています)
以下、サンプルコードになります。また、サンプル用のデータベースも用意します。

- スキーマとテーブル

ユーザー名 : ajax_user

パスワード : systemsss

権限 : 「SELECT ANY DECTIONARY」 権限以外の全ての権限

ajax_sample テーブル

No	論理名称	物理名称	物理データ型(桁数)	PK	制約
1	ID	id	NUMBER(4)	○	
2	名前	name	VARCHAR2(30 CHAR)		NOT NULL
3	お金	money	NUMBER(8)		NOT NULL

SQL 文

```
--ユーザー名 system でログイン後、新規にスキーマを作成する。
CREATE USER ajax_user IDENTIFIED BY systemsss;
GRANT ALL PRIVILEGES TO ajax_user;

--一旦データベースとの接続を切断し、先程作成したユーザー名で接続する。
--接続後、ajax_sample テーブルを作成する。
CREATE TABLE ajax_sample (
  id NUMBER(4) PRIMARY KEY,
  name VARCHAR2(30 CHAR) NOT NULL,
  money NUMBER(8) NOT NULL
);

--テーブル作成後、各テーブルに動作テスト用レコードを挿入する。
INSERT INTO ajax_sample VALUES (1, '山田太郎', 10000);
INSERT INTO ajax_sample VALUES (2, '田中健一', 25000);
INSERT INTO ajax_sample VALUES (3, '鈴木花子', 500000);
COMMIT;
```

続いてクラスファイルを作成していきます。

動的 Web プロジェクトで作成します。(サンプルではパッケージ名を `jp.co.sss.ajaxSample` としています)

AccountDAO.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class AccountDAO {

    public static Connection getConnection() {
        Connection con = null;
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con = DriverManager.getConnection(
                "jdbc:oracle:thin:@//localhost:1521/xe", "ajax_user",
                "systemsss");
        } catch (Exception e) {
            e.printStackTrace();
        }
        return con;
    }

    public static String findById(String inputParam) throws SQLException {

        Connection con = null;
        PreparedStatement ps = null;
        try {
            con = getConnection();
            // テーブル名は ajax_sample、カラムは id、name、money の 3 つ
            ps = con.prepareStatement("SELECT * FROM ajax_sample WHERE id = ?");
            ps.setString(1, inputParam);
            ResultSet rs = ps.executeQuery();
            StringBuilder sb = new StringBuilder();
            if (rs.next()) {
                sb.append(" id: ");
                sb.append(rs.getInt("id"));
                sb.append(" name: ");
                sb.append(rs.getString("name"));
                sb.append(" money: ");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return sb.toString();
    }
}
```

```
        sb.append(rs.getInt("money"));
    }
    return sb.toString();
} finally {
    if (ps != null) {
        try {
            ps.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    if (con != null) {
        try {
            con.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
}
```

IndexAction.java

```
import java.io.IOException;
import java.sql.SQLException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class IndexAction extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        String inputParam = request.getParameter("id");
        try {
            String outputParam = AccountDAO.findById(inputParam);
            request.setAttribute("output", outputParam);
            RequestDispatcher dispatcher = request.getRequestDispatcher("/list.jsp");
            dispatcher.forward(request, response);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

続いて web.xml を修正します。

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
  id="WebApp_ID" version="2.5">
  <display-name>AjaxSample</display-name>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
  <servlet>
    <description></description>
    <display-name>IndexAction</display-name>
    <servlet-name>IndexAction</servlet-name>
    <servlet-class>jp.co.sss.ajaxSample.IndexAction</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>IndexAction</servlet-name>
    <url-pattern>/IndexAction</url-pattern>
  </servlet-mapping>
</web-app>
```

続いて JSP ファイルの作成を行います。

list.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
${output}
```

index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<script type="text/javascript"
    src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script type="text/javascript">
$(function() {
    $("input[name=idSend]").click(function() { ❶
        $.ajax({ ❷
            type : "GET", ❸
            url : "<%=request.getContextPath()%>/IndexAction", ❹
            data : { ❺
                id : $("input[name=accountId]").val()
            }
        }).done(function(data, status, req){ ❻
            $("h2").html(data);
        });
    });
});
</script>
</head>
<body>
<h1>検索したい id を入力してください</h1>
<form>
<input type="text" name="accountId" />
<input type="button" name="idSend" value="AJAX 通信" />
</form>
<h2></h2>
</body>
</html>
```


注目してほしいのは Ajax を使用している index.jsp です。

①では name="idSend" のボタンがクリックされたら、実行されます。
ここまでは今まで使用した jQuery 文と変わりません。

②で Ajax が実行されます。\$.ajax()メソッドの引数には色々なパラメータを指定することができますが、基本の種別があります。下記を参考にしてください。

名前	説明
url	リソースの URL を指定します。 Servlet/JSP(フレームワーク)の場合は web.xml で設定した path を指定します。
type	HTTP メソッドの種別を"GET"か"POST"で指定します。 Servlet/JSP の場合は doGet()メソッドを呼ぶか、 doPost()メソッドを呼ぶか指定することができます。
data	サーバに送信するフォームデータを指定します。 例: data: {id:"test", pw:"pass"} と指定すると id と pw という名前を紐づけてリクエストを実行します。
timeout	timeout に指定した時間(ミリ秒)が経過しても通信が完了しない場合はエラー処理(fail(),error)が実行されます。 スマートフォン向けのサイト等では、電波の影響で通信に時間が掛かる場合などにメッセージを表示するので現在では使用機会がとて多いオプションです。 ※Ajax のタイムアウト値は 10 秒~30 秒が目安になります。

③では HTTP メソッドを"GET"で指定しているので、url で指定した Servlet の doGet()メソッドをリクエストします。

④ではリソースの URL を指定しています。今回の場合は web.xml で記述した path"/IndexAction"を指定しています。

⑤ではサーバ側に送信するフォームデータを指定します。今回の場合、name="accountId"の値をサーバに送信します。<input type="text" name="test" />とした場合は data オプションに name="test"と指定します。

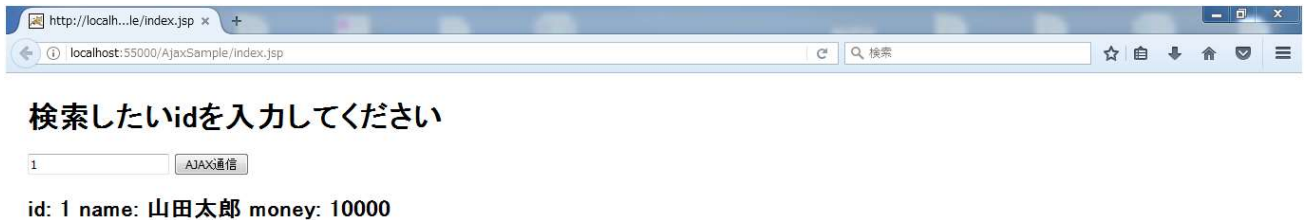
⑥の.done ()では ajax()メソッドの戻り値を引数として受け取ります。今回の場合は受け取った引数の値を画面に表示しています。

以下実行画面の例になります。画面遷移をしていないので、値を表示しても URL が変わっていないことに注目してみてください。

実行結果



検索したいidを入力してください



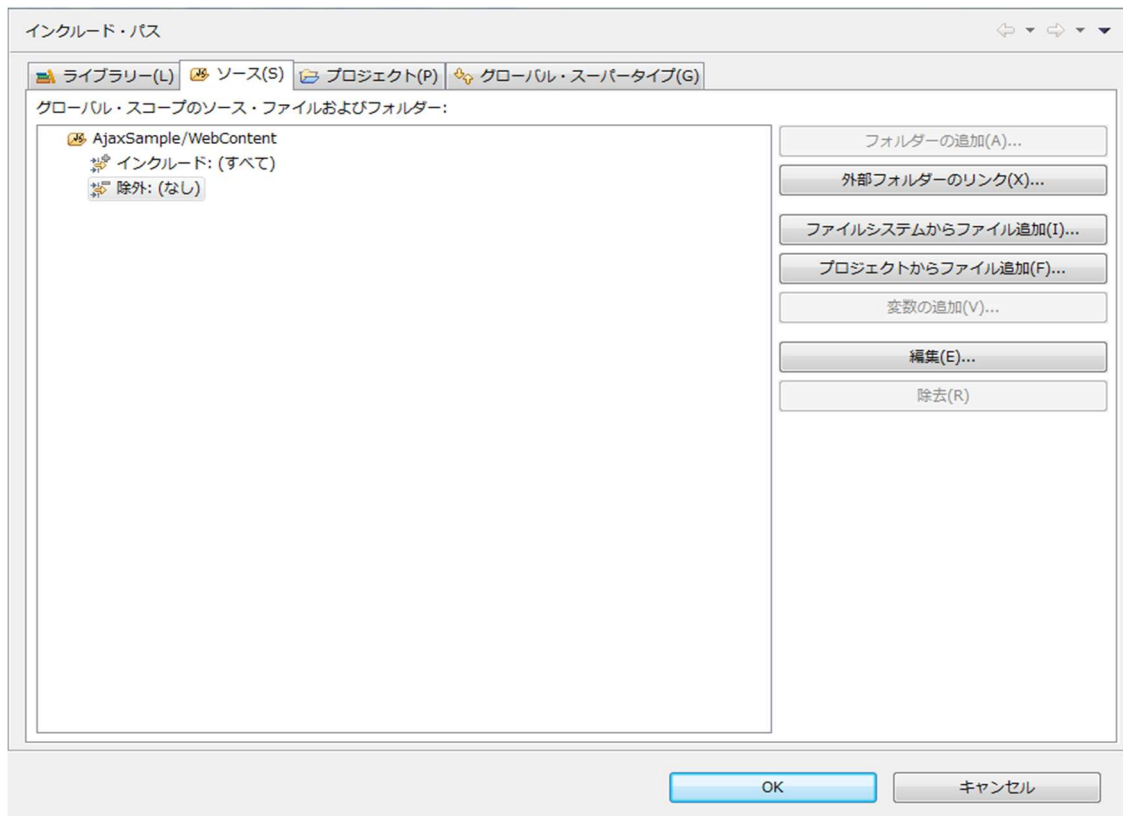
検索したいidを入力してください

id: 1 name: 山田太郎 money: 10000

2 Eclipse と jQuery

Eclipse で動的 Web プロジェクトを作成し、jQuery を使用すると、本体の js ファイルが構文エラーになってしまいます。そのままだでも問題なくプロジェクトの実行は可能ですが、プロジェクトに×印が付いてしまうので、Eclipse の除外設定を次の手順で行います。

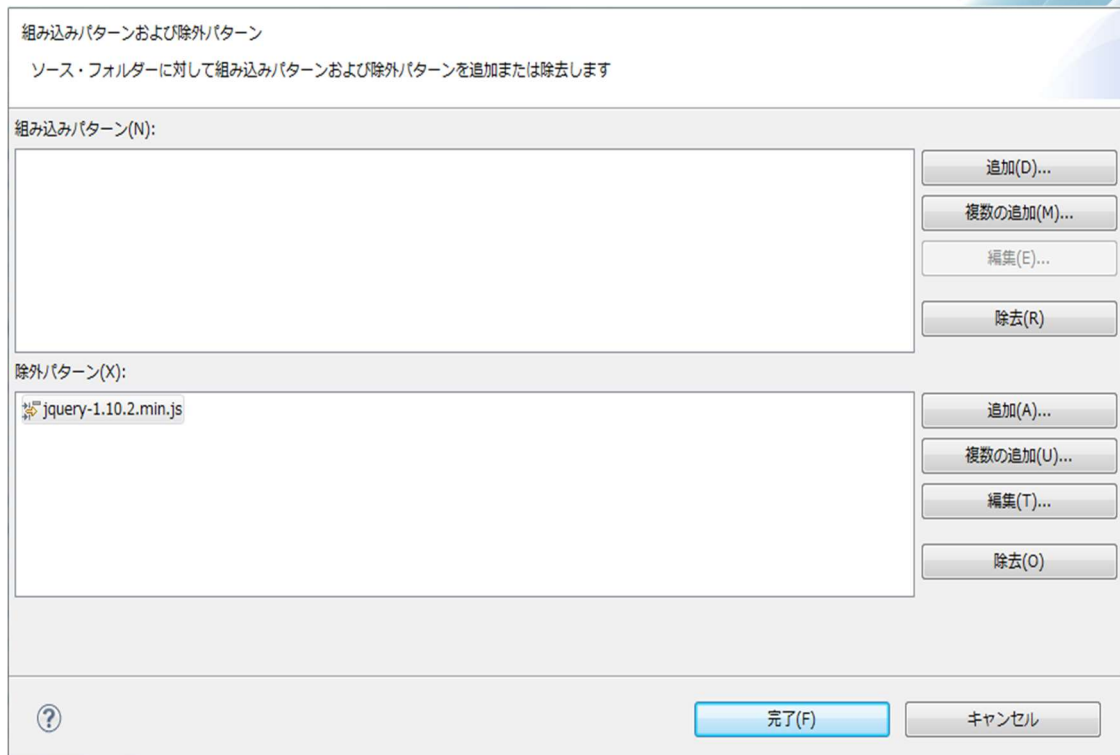
- 1.1 該当プロジェクトを右クリックし、プロパティを選択。
- 1.2 「JavaScript」を開く。
- 1.3 「インクルード・パス」を開き、ソースタブを選択。
- 1.4 フォルダを指定し、「除外」を選択後、右側の「編集」ボタンをクリック。



- 1.5 除外パターンテキストボックス横の「追加」ボタンをクリック。
- 1.6 「参照」ボタンをクリックし、`*/jquery*.js` を指定し、「OK」ボタンをクリック。
(例では `jquery-1.10.2.min.js` を指定)



1.7 「完了」ボタンをクリックする。



jQuery のプラグインだけではなく、外部の js ファイルは、問題なく起動するのに Eclipse では赤い警告が発生してしまうことがあります。Java ファイルのコンパイルエラーと見分けづらくなるので、その際は上記と同様に除外設定を行うようにしてください。