情報科学プロジェクト実験 コンピュータシステム研究室
XMLHttpRequestによるJavaScriptからのHTTPの制御

HTMLフォームの中でsubmitを用いると、HTTPによりフォームの内容をWebサーバに送ることができた。しかし、その結果、Webページはリロードされて、Webブラウザ上では別のページが表示される。これはJavaScriptからフォームのsubmit()を操作しても同じである。
XMLHttpRequestオブジェクトを使うと、ページをリロードすることなく、Webサーバからの結果を受け取ることができる。結果は、JavaScriptのDOM操作により表示しても良いし内部で使用しても良い。この技術はAjax(Asynchronous JavaScript + XML)と呼ばれる。

使い方の基本
リクエストの送信:

```
var x = new XMLHttpRequest(); // XMLHttpRequestオブジェクトのインスタンス化
x.onreadystatechange = function(){...} // 結果を受け取る場合の関数を作っておく
x.open("GET", "/cgi-bin/a.cgi"); // メソッドとURLを指定（この時点では送信しない）
x.send(null); // リクエストを送信
```

結果の受け取り:
send()メソッドの呼び出しにより、リクエストをサーバに送信するが、デフォルトでは結果を受信する前にsend()メソッド呼び出しから戻る。これを非同期通信と呼ぶ。
結果が到着すると、readystatechangeイベントが発生するので、このイベントに対応する関数を事前に設定しておく必要がある。これをコールバック関数と呼ぶ。
このコールバック関数では、イベント発生タイミングの把握とエラーの有無を確認してから、結果に対する処理を行う必要がある。
readystatechangeイベントは、正確には以下の５つの状態が変化する度に発生する。

状態	意味
0	open()が呼び出されていない
1	open()が呼び出された
2	ヘッダが指定された（後述）
3	結果を受信中
4	結果の受信が完了した

状態番号はreadyStateで確認できる。Webブラウザによって微妙にタイミングが異なるので、状態４以外は利用できないと考えてよい。結果として、結果の受け取りは、以下の様に指定する。

```
x.onreadystatechange = function() {
        if (x.readyState == 4 && x.status == 200)
                callback(x.responseText);
}
```

x.statusはHTTPの結果であり、200は成功を意味する。404ならばnot foundである。
x.responseText は結果をテキストのまま受け取る場合の指定で、callback()が何らかの処理を表す。

フォームの内容をPOSTメソッドで送る場合には、上記に加えて指定が必要である。
POSTで送る内容そのものとその種類をヘッダに指定する点である。以下に例を示す。

```
var x = new XMLHttpRequest(); // XMLHttpRequestオブジェクトのインスタンス化
x.onreadystatechange = function(){...} // 結果を受け取る場合の関数を作っておく
x.open("POST", "/cgi-bin/a.cgi"); // メソッドとURLを指定
x.setRequestHeader("Content-Type", // HTTPのヘッダ情報の指定
                   "application/x-www-form-urlencoded");
x.send(data); // リクエストを送信
```

telnetでHTTPを試した時のことを思い出してほしい。

```
% telnet 192.168.114.15  80
Trying 192.168.114.15...
Connected to 192.168.114.15.
Escape character is '^]'.
POST /~okam/cgi-bin/test.cgi HTTP/1.1
Host: cssv
Content-Length: 31

message1=abc&message2=xyz&ok=OK
```

ここで、Host: やContent-Length:がヘッダ情報であり、message1=abc&message2=xyz&ok=OKが送信内容である。
フォームの内容をPOSTで送る場合には、このようにA=Bの形で指定したinputの属性を&で区切って複数並べる。この形式は application/x-www-form-urlencoded と呼ばれる（URLエンコードとも呼ばれる）。
上記の例で、x.setRequestHeader()はこのヘッダ情報を設定するもので、x.send(data)のdataはこの形式の文字列になっていなければならない。

jQueryによるAjaxプログラム

これまでに説明した内容はパターン化しているために、
jQueryでも簡単に扱えるようになっている。

```
$.get(url, data, callback)
        GETメソッドでdataをurlに送り、
        サーバから戻された結果をcallbackで処理させる。

$.post(url, data, callback)
        POSTメソッドでdataをurlに送り、
        サーバから戻された結果をcallbackで処理させる。

var x = $('#form').serialize();
        フォームオブジェクトからinput要素のname属性と入力内容を取り出し、
        $.get()や$.post()のdataとして使用できる形式に変換する。
```

例：

```
var x = $('#form').serialize();
$.post('/cgi-bin/test.cgi', x, function(result) {
      // ここに結果に対する処理を書く
});
```

課題：
以下の機能を持つ html with JavaScript を1個のファイルで作る。
可能な限りhtmlタグを直に書かないでJavaScriptだけで書く。
文字コードはすべて UTF-8にすること。

入力画面: 以下のform入力を行う
郵便番号
住所
氏名
年齢
性別

[消去]ボタンによりform内容の消去
[送る]ボタンにより確認の表示

Ajaxにより郵便番号7桁から住所入力の補助を行う。
郵便番号7桁を入力したら住所欄を埋める

確認画面:htmlのテーブル形式で入力内容を表示
[編集]ボタン formにもどる
[送る]ボタン実際にサーバにデータを送る
formは非表示にするだけ

最終画面: 「ありがとうございました。」を表示

サーバ側では入力データをDBに格納する
dbディレクトリに置く
サーバ側のCGIは1または2個のプログラムとする

ヒント：
・画面ごとに タグでまとめると良い。
・htmlのテーブル形式は前回配った資料にある。