Changing <textarea> Size

機能概要 テキストエリアをリサイズ可能にする

ライセンス Creative Commons BY

配布サイトURL http://lowreal.net/logs/2006/03/22/1

作　者 cho45（さとう）

解　説 このスクリプトをインストールすると、全ての<textarea>要素がリサイズ可能になります。たとえば、Webcat Plus（http://webcatplus.nii.ac.jp/）にアクセスすると、トップページに大きなテキストエリアがあるので、このテキストエリアの右下の部分をドラッグしてみましょう。自在に大きさが変更できます（**図4-24**、**図4-25**）。

図4-24

図4-25

長文を入力しようとするとどうしてもテキストエリアの小ささが気になってしまいますが、このスクリプトを使えば好きな大きさに変更できるので、そのような悩みとはオサラバです。

使い方 テキストエリアの右下をドラッグするだけなので、使い方と呼ぶような事は何もありません。強いて言えば、クリックの判定が厳しいので慣れるまではうまくリサイズできないかもしれません。何度か触って感触をつかんでください。

この手の機能はかなり需要があるらしく、似た機能を持ったスクリプトがいくつかあるので、これらも併せてご紹介しましょう。どれも細かい点が違うので、好みのスクリプトを見付けてください。

Textarea_drag_resize

http://userscripts.org/scripts/show/5073

図4-26

このスクリプトをインストールすると、テキストエリアの右下にリサイズ用のアイコンが表示されます。表示されたアイコンをドラッグするだけなので、操作しやすいです。

Stretch-to-fit Textareas

http://userscripts.org/scripts/show/5090

図4-27

このスクリプトをインストールすると、テキストエリアの脇に小さなチェックボックスが出現します。こいつのチェックを入れておくと、入力中の内容に合わせて自動的にテキストエリアの大きさが変化します。いちいち自分で大きさをいじるのが面倒な場合はこのスクリプトがお勧めです。

GreaseMonkey.04 06.12.15 11:50 AM ページ252

TabArea

http://userscripts.org/scripts/show/4066

図4-28

このスクリプトをインストールすると、テキストエリアを[Ctrl]キーを押しながらダブルクリックするだけで、大きくしたり元に戻したりできるようになります。

実はリサイズの機能はおまけで、メインの機能はテキストエリア内にタブが挿入できるようになる事です。通常は[Tab]キーを押すと、次の入力フィールドに移動してしまいますが、このスクリプトをインストールしておくと、通常のテキストエディタと同じくタブが挿入できるようになります。

Move and Size TextAreas

http://userscripts.org/scripts/show/1553

図4-29

このスクリプトをインストールすると、ページの一番右下に半透明のパネルが追加されます。テキストエリアが表示されているページ上でそのパネルをクリックすると、テキストエリアの右下に、移動とリサイズ用のボタンが出現します。

赤いボタンをドラッグするとテキストエリアが移動します。どこでも好きな位置に動かす事ができます。青いボタンがリサイズです。これも好きな大きさに変更する事ができます。[shift]キーを押しながら青いボタンをドラッグすると、中央の位置を固定したままリサイズします。

One Point さて、Changing <textarea> Sizeのスクリプトに話しを戻しましょう。実はこのスクリプトで使われている$X関数が非常に絶妙なので、是非知っておいてください。$X関数を使えば、XPathを扱うのが物凄く楽になります。

$X関数のコードは次の通りです。

```
function $X(exp, context) {
  if (!context) context = document;
  var resolver = function (prefix) {
    var o = document.createNSResolver(context)(prefix);
    return o ? o : (document.contentType == "text/html") ? "" : "http://www.w3.org/1999/xhtml";
  }
  var exp = document.createExpression(exp, resolver);

  var result = exp.evaluate(context, XPathResult.ANY_TYPE, null);
  switch (result.resultType) {
    case XPathResult.STRING_TYPE : return result.stringValue;
    case XPathResult.NUMBER_TYPE : return result.numberValue;
    case XPathResult.BOOLEAN_TYPE: return result.booleanValue;
    case XPathResult.UNORDERED_NODE_ITERATOR_TYPE: {
      result = exp.evaluate(context, XPathResult.ORDERED_NODE_SNAPSHOT_TYPE, null);
      var ret = [];
      for (var i = 0, len = result.snapshotLength; i < len ; i++) {
        ret.push(result.snapshotItem(i));
      }
      return ret;
    }
  }
  return null;
}
```

なお、$X関数を含むChanging <textarea> SizeのライセンスはCreative CommonsのBYライセンス(http://creativecommons.org/licenses/by/2.1/jp/)なので、原著者であるcho45さんのクレジットさえ明記しておけば自由に再利用可能です。$X関数の部分だけを抜き出す事もできるので、cho45さんに感謝しつつ利用させてもらいましょう。

さて、$X関数を使うとXPathがどれくらい幸せになるのか実際のコードを見ながら追っていきましょう。

たとえば、今回のように全textareaを取得したい場合、普通にXPathで書くと次のようになると思います。

```
var nodes = document.evaluate(
    '//textarea', document, null,
    XPathResult.ORDERED_NODE_SNAPSHOT_TYPE, null
    );
```


この上、個別のデータを取得する場合はsnapshotItemメソッドを使う必要が有りますし、本書では省略していますがdocument.evaluateの引数に使える値も様々なので、一々全部の呪文を書いてたら日が暮れてしまいます。しかし、そんな不満も$X関数を使う事で全部解決してしまいます。$X関数を使って全textareaを取得する処理を書くと、次のように単純になります。

```
var textareas = $X("//textarea");
```

これだけです。

戻り値は配列なので、その後の取り扱いも非常に簡単です。たとえば、テキストエリアをクリックした際にアラートを表示するスクリプトは、次のようになります。

```
var nodes = $X("//textarea");

nodes.forEach(function (elm) {
  elm.addEventListener(
    'click',
    function() { alert('hehehe'); },
    false
  );
});
```

謎の呪文群が消え去ったので、スッキリとしました。

$X関数についてさらに詳しい事は、作者であるcho45さんのブログに詳しい情報が載っているので、そちらをご参照ください。

cho45さんのブログ

http://lowreal.net/logs/2006/03/16/1

Show Password onMouseOver

機能概要 パスワード欄に入力した内容を見れるようにする

ライセンス 不明

配布サイトURL http://userscripts.org/scripts/show/1893