Microsoft AJAX Library
～ 8年越しの開花 ～

DHTMLがもてはやされた時代がありました

- JavaScriptが一時もてはやされた時代がありました（Dynamic HTMLと言いまして・・・）
  - 98年~01年位
- 適度に利用できるくらいにNN4, IE4で拡張され、クライアントマシンにも少し余裕ができ利用できる地盤ができたころです。

- ただしJavaScriptはいつしかメインストリームではなくなっていきます。

DHTMLの凋落

- DHTMLはなぜ凋落したのか？
  - ブラウザ非互換
    - もともとIEとNNで違う実装であった
      - 作る側の知識が必要
  - セキュリティの問題
    - ブラウザクラッシャー
    - リンク先詐称
    - セキュリティ系のコミュニティでは無効を推奨
      - 私も推奨していますが(^^;

まずは単純なDHTMLの例を見てみましょう。

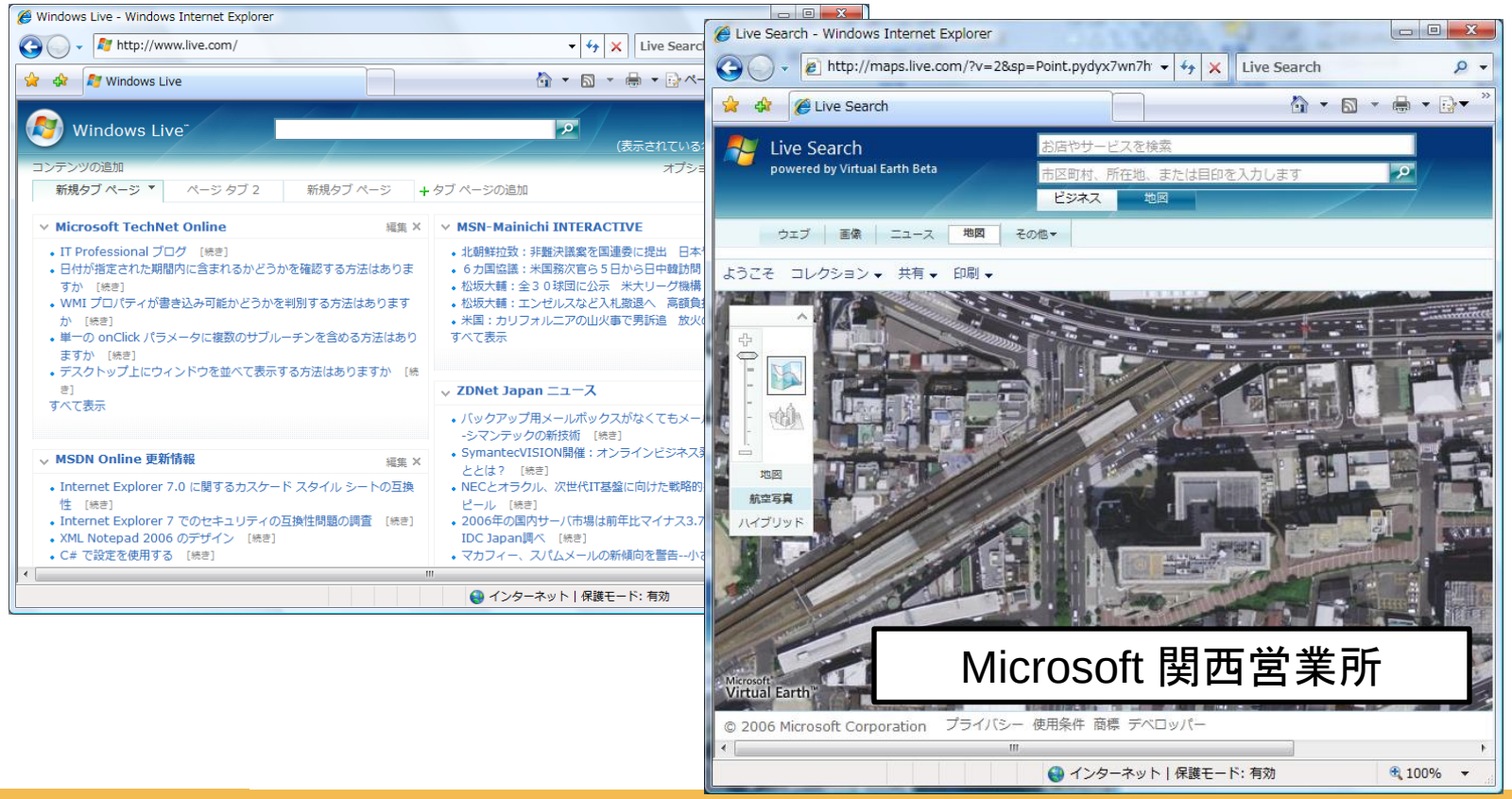
DEMO1

AJAXってなに？

- AJAXってなに？
- Asynchronous JavaScript + XMLの略
- JavaScriptとXMLを使った非同期技術くらいが適当な訳か…な?

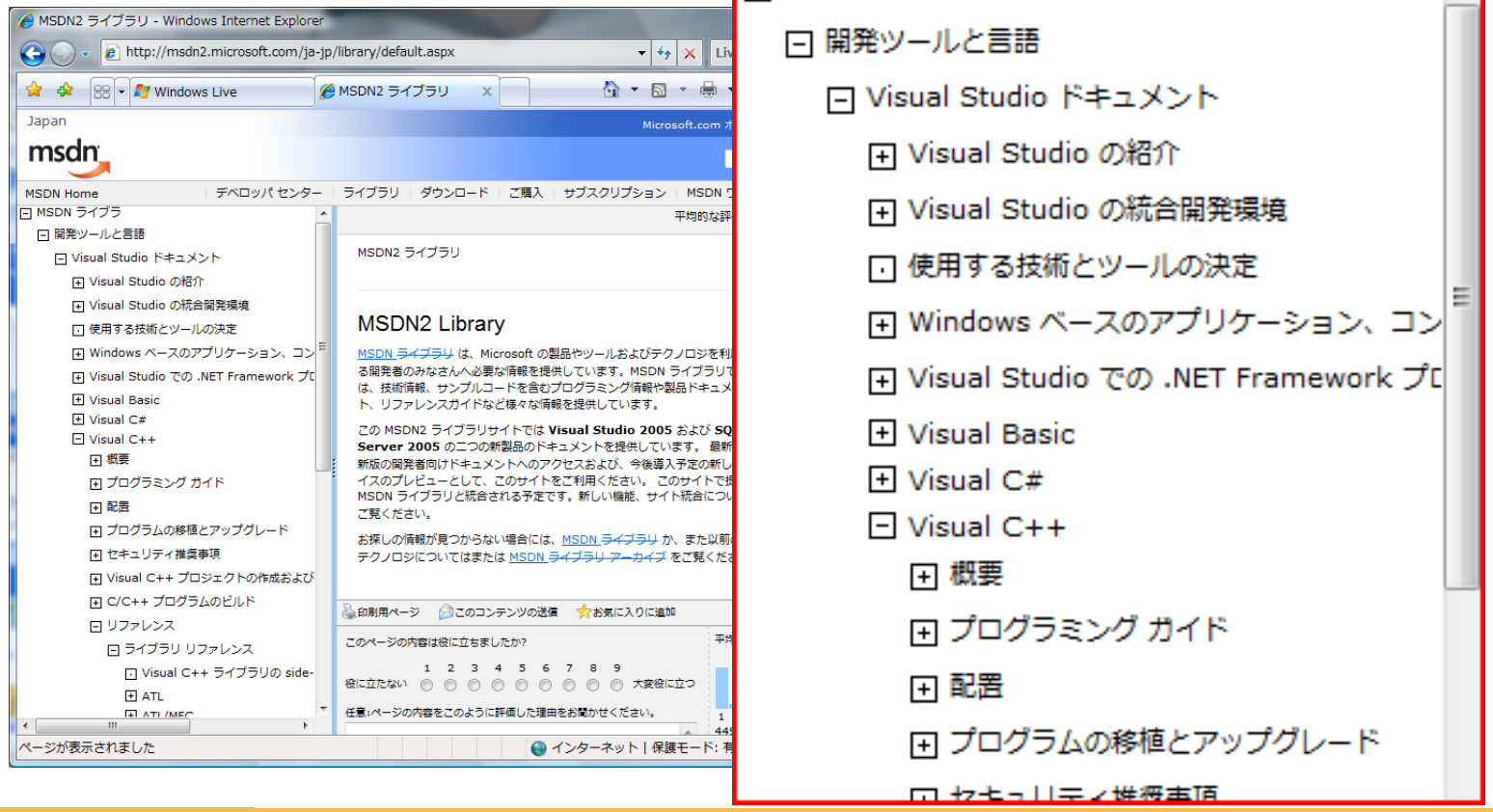
AJAXで復活？

- ここ1年ほどでAJAXが注目されJSも再びメインストリームに戻ってきたようです。



でも・・・・

- われわれIT技術者は見慣れているはずですよ！！！！



AJAXのポイント1

URLが変わらない

東京国際空港(羽田)

東京駅

AJAXのポイント1

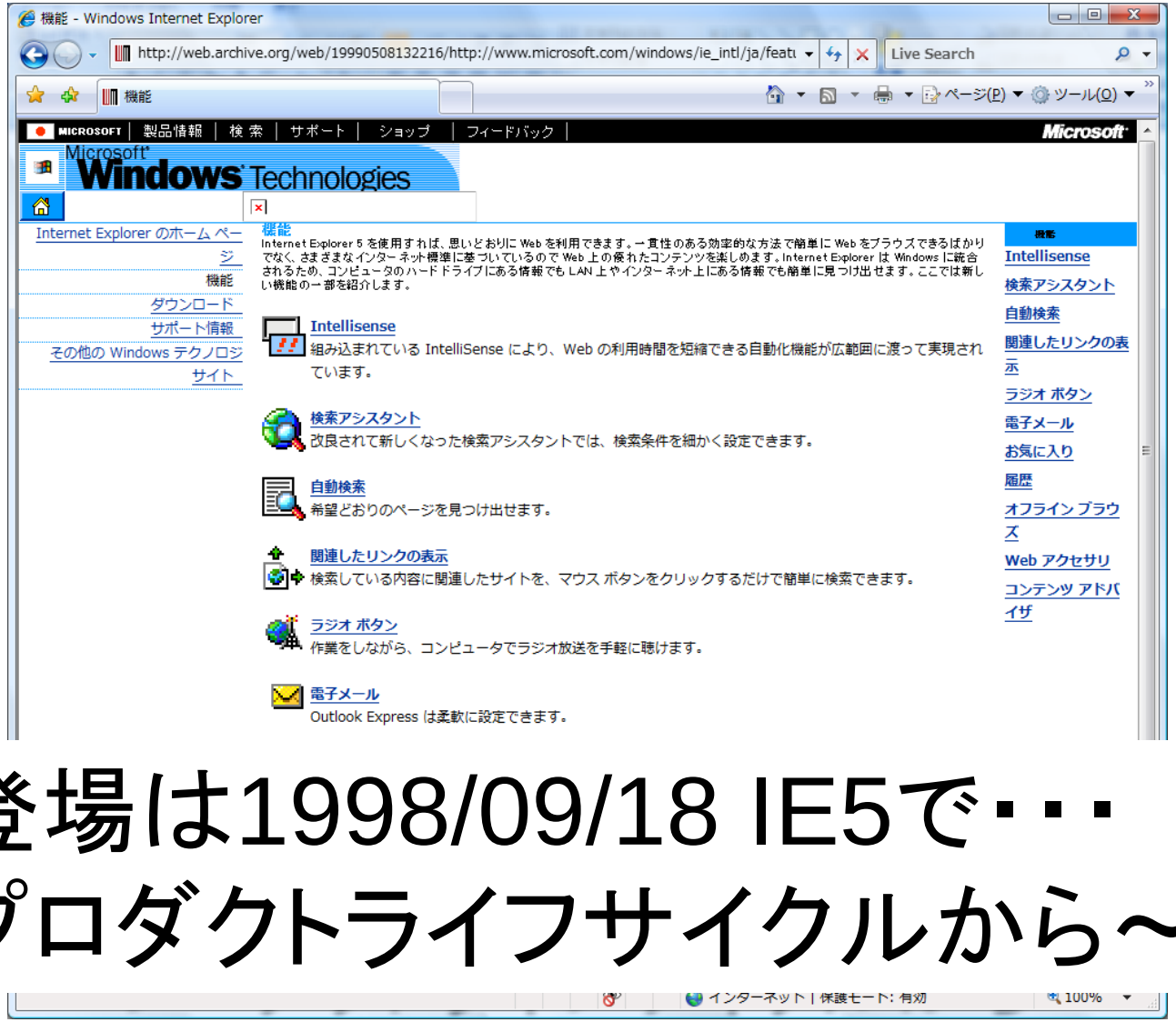
URLが変わらないことにより

- メリット
  - 同一ページをアピールしやすい
  - アプリケーションをアピールしやすい
- デメリット
  - ブックマークできない
  - サーチエンジンに引っ掛かりにくい

AJAXのポイント2

内部的に
MSXML.XMLHTTP(IE6以前)
か
XMLHttpRequest
　(IE7, その他ブラウザ)
を利用している。

IE 5.0



初登場は1998/09/18 IE5で・・・
～プロダクトライフサイクルから～

XMLHttpRequestって

- 属性
- onreadystatechange
- readyState
- responseBody
- responseText
- responseXML
- status
- statusText

- メソッド
- abort
- getAllResponseHeaders
- getResponseHeader
- open
- send
- setRequestHeader

非常に小さいHttpクライアントの実装

DEMO2

いまのデモはただの
XMLHttpRequestの
同期版のデモでした。

次は非同期

XMLHttpRequestの非同期DEMO

DEMO3

objXHR.responseXML.xmlの示しているオブジェクトはStringです。

このあとさらにXMLのパースを行い

表示しているHTMLに反映する必要があります。

めんどくさーーーーーい

- そのための

AJAXは主に3つのカテゴリから

ASP.NET AJAX Control Toolkit

ASP.NET AJAX Extension

.NET 2.0 Runtime / ASP.NET 2.0	Microsoft AJAX Library
IIS	IE, Firefox, Opera, Safari
Windows	Windows / Macintosh
サーバ	クライアント

AJAX Extensionは現時点で2バージョン

今日紹介する
- Timer
- UpdatePanel
- UpdateProgress
- JSからのXMLサービス呼び出し
- JSON

Webサービスブリッジ
- オートコンプリート
- さまざまなコントロール
- XMLスクリプト

先行リリース(2006年内予定)

次期リリース

ASP.NET AJAX Extension

http://ajax.asp.net/files/AspNet_AJAX_CTP_to_Beta_Whitepaper.aspx

ASP.NET 2.0 AJAX Extensions

まずは

- UpdatePanel、UpdateProgressをご覧ください。

DEMO4

Demo4のポイント1

- メリット
  - UpdatePanelでくくるだけ
  - UpdateProgressも置くだけ
  - 簡単にAJAXのメリットを享受できる
- デメリット
  - 今までページ単位に処理の整合性を見ていたのが、ブロック単位に検討する必要がある。
  - 2つのUpdatePanelを置く場合には非常に慎重に(途中で意図せずキャンセルは発生する)

Demo4のポイント2

- 中身を変更してもURLが変わらない。
- ASP.NETのポストバック機能との親和性が高い

ページ間移動は今まで通り

Response.Redirectを多用する

Timer

DEMO5

Demo5のポイント

- メリット
  - 定期処理っぽいものが実現できる(決して定期処理ではない)
- デメリット
  - サーバ負荷をよく考えないと大変なものを簡単に作製可能
  - リクエストが来ないことも検討しなくてはいけない

Webサービス

DEMO6

ポイント1

```
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
[Microsoft.Web.Script.Services.ScriptService]
public class WebService : System.Web.Services.WebService {
```

Microsoft.Web.Script.Services.ScriptService属性を付ける

ポイント2

```
<httpHandlers>
  <remove verb="*" path="*.asmx"/>
  <add verb="*" path="*.asmx" validate="false" type="Microsc
  <add verb="GET" path="ScriptResource.axd" type="Microsoft.
</httpHandlers>
```

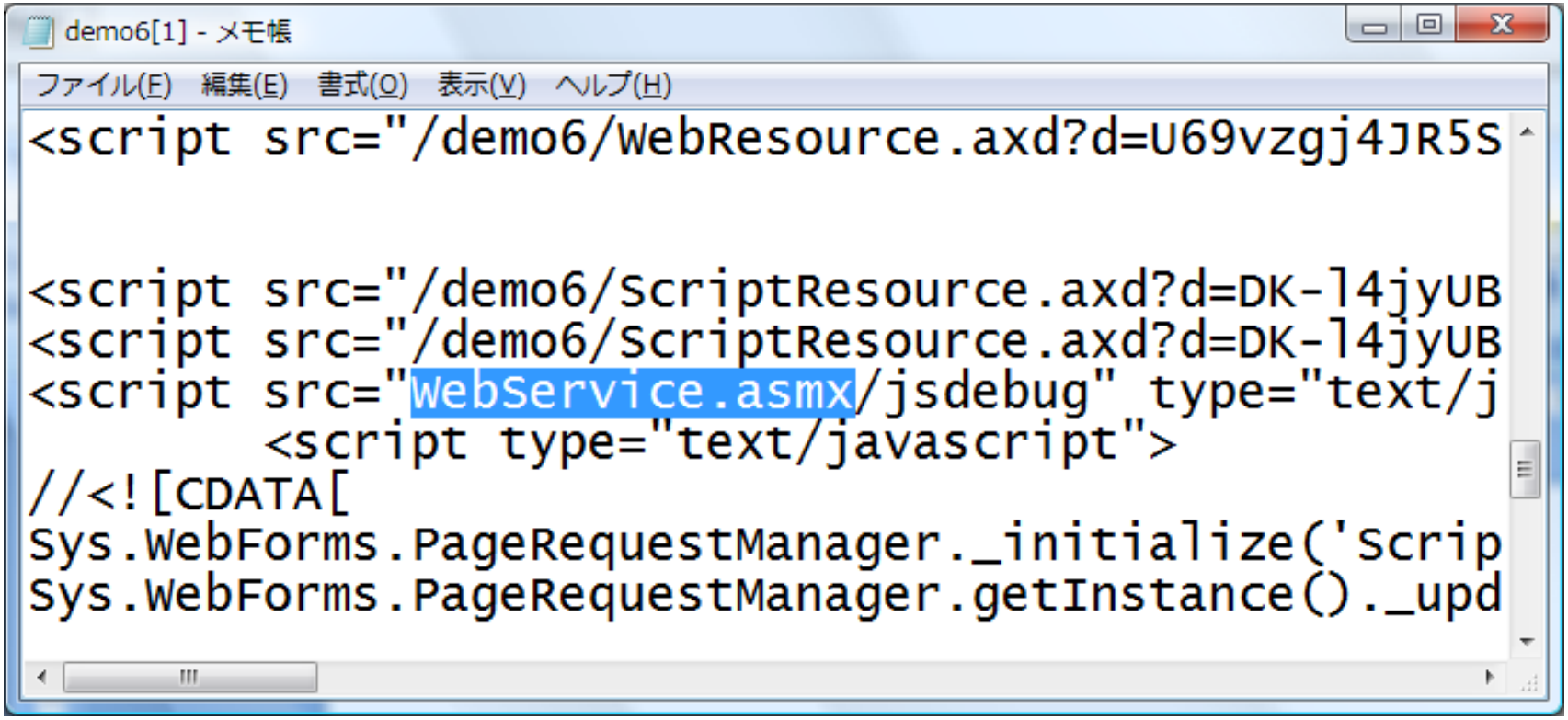
asmxの拡張子を乗っ取っている

System.Web.Services.Protocols.WebServiceHandlerFactory

→

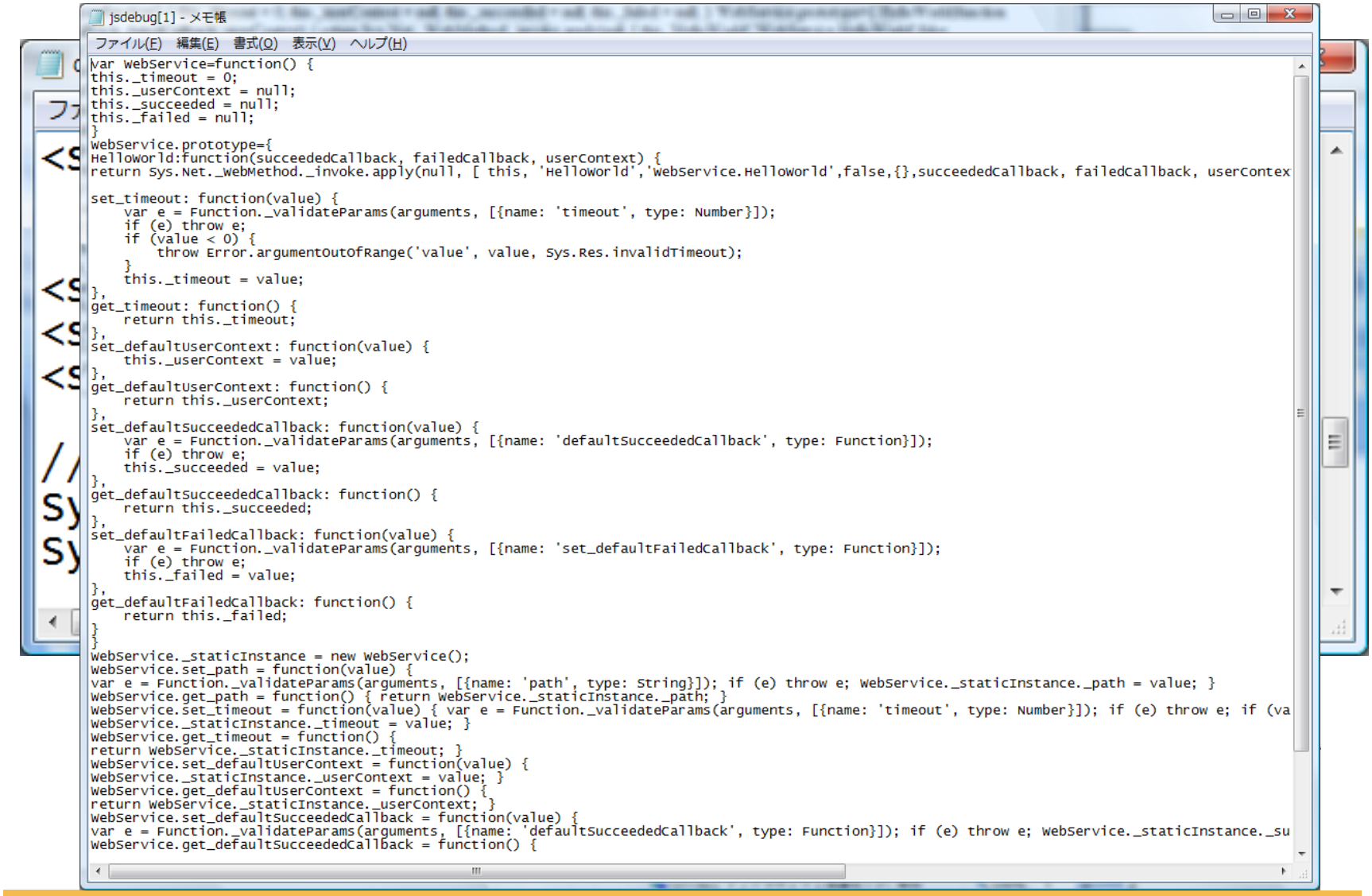
Microsoft.Web.Script.Services.ScriptHandlerFactory

ポイント3



WebServiceがJavaScriptのファイルを供給している

ポイント3

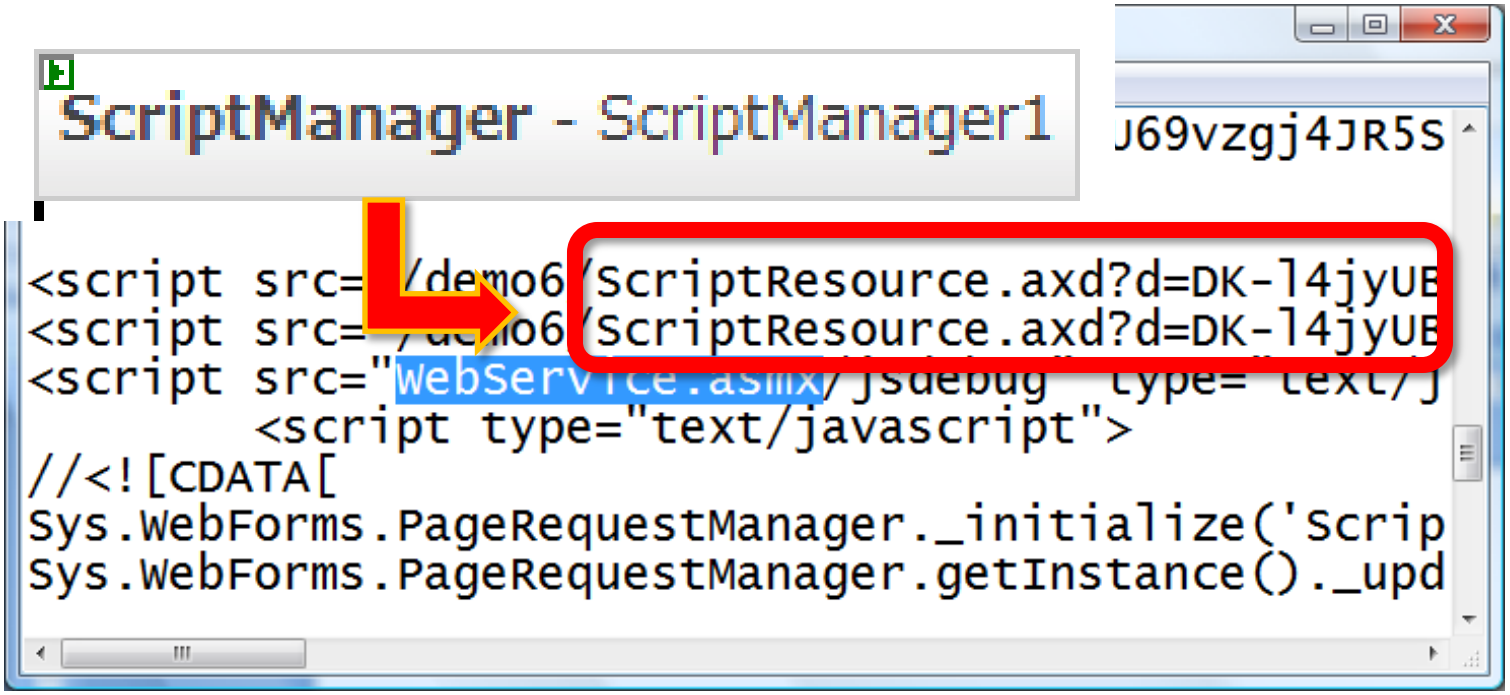

jsdebug[1] - メモ帳
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)
var WebService=function() {
this._timeout = 0;
this._userContext = null;
this._succeeded = null;
this._failed = null;
}
WebService.prototype={
HelloWorld:function(succeededCallback, failedCallback, userContext) {
return Sys.Net._WebMethod._invoke.apply(null, [this, 'HelloWorld','WebService.HelloWorld',false,{},succeededCallback, failedCallback, userContex
set_timeout: function(value) {
var e = Function._validateParams(arguments, [{name: 'timeout', type: Number}]);
if (e) throw e;
if (value < 0) {
throw Error.argumentOutOfRange('value', value, Sys.Res.invalidTimeout);
}
this._timeout = value;
},
get_timeout: function() {
return this._timeout;
},
set_defaultUserContext: function(value) {
this._userContext = value;
},
get_defaultUserContext: function() {
return this._userContext;
},
set_defaultSucceededCallback: function(value) {
var e = Function._validateParams(arguments, [{name: 'defaultSucceededCallback', type: Function}]);
if (e) throw e;
this._succeeded = value;
},
get_defaultSucceededCallback: function() {
return this._succeeded;
},
set_defaultFailedCallback: function(value) {
var e = Function._validateParams(arguments, [{name: 'set_defaultFailedCallback', type: Function}]);
if (e) throw e;
this._failed = value;
},
get_defaultFailedCallback: function() {
return this._failed;
}
}
WebService._staticInstance = new WebService();
WebService.set_path = function(value) {
var e = Function._validateParams(arguments, [{name: 'path', type: String}]); if (e) throw e; WebService._staticInstance._path = value; }
WebService.get_path = function() { return WebService._staticInstance._path; }
WebService.set_timeout = function(value) { var e = Function._validateParams(arguments, [{name: 'timeout', type: Number}]); if (e) throw e; if (va
WebService._staticInstance._timeout = value; }
WebService.get_timeout = function() {
return WebService._staticInstance._timeout; }
WebService.set_defaultUserContext = function(value) {
WebService._staticInstance._userContext = value; }
WebService.get_defaultUserContext = function() {
return WebService._staticInstance._userContext; }
WebService.set_defaultSucceededCallback = function(value) {
var e = Function._validateParams(arguments, [{name: 'defaultSucceededCallback', type: Function}]); if (e) throw e; WebService._staticInstance._su
WebService.get_defaultSucceededCallback = function() {

Demo6のポイント

- メリット
  - XML Webサービスを簡単にまともに利用できる局面ができた
- デメリット
  - jsメソッドは二つに分けなくてはいけない

Microsoft AJAX Library

正体は




```
<httpHandlers>
  <remove verb="*" path="*.asmx"/>
  <add verb="*" path="*.asmx" validate="false" type="Microso
  <add verb="GET" path="ScriptResource.axd" type="Microsoft
</httpHandlers>
```

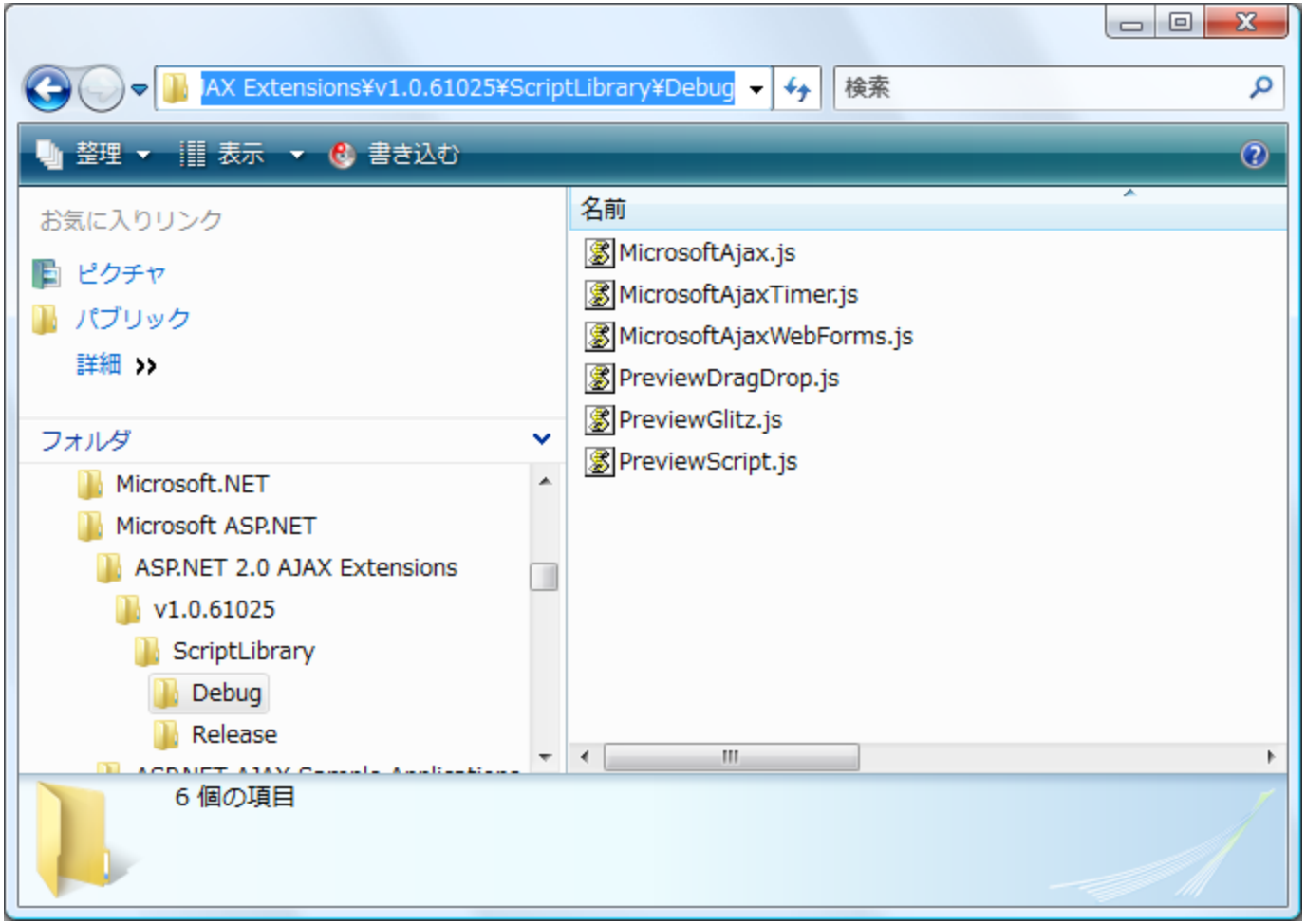
正体は

Microsoft.Web.Handlers.

ScriptResourceHandler

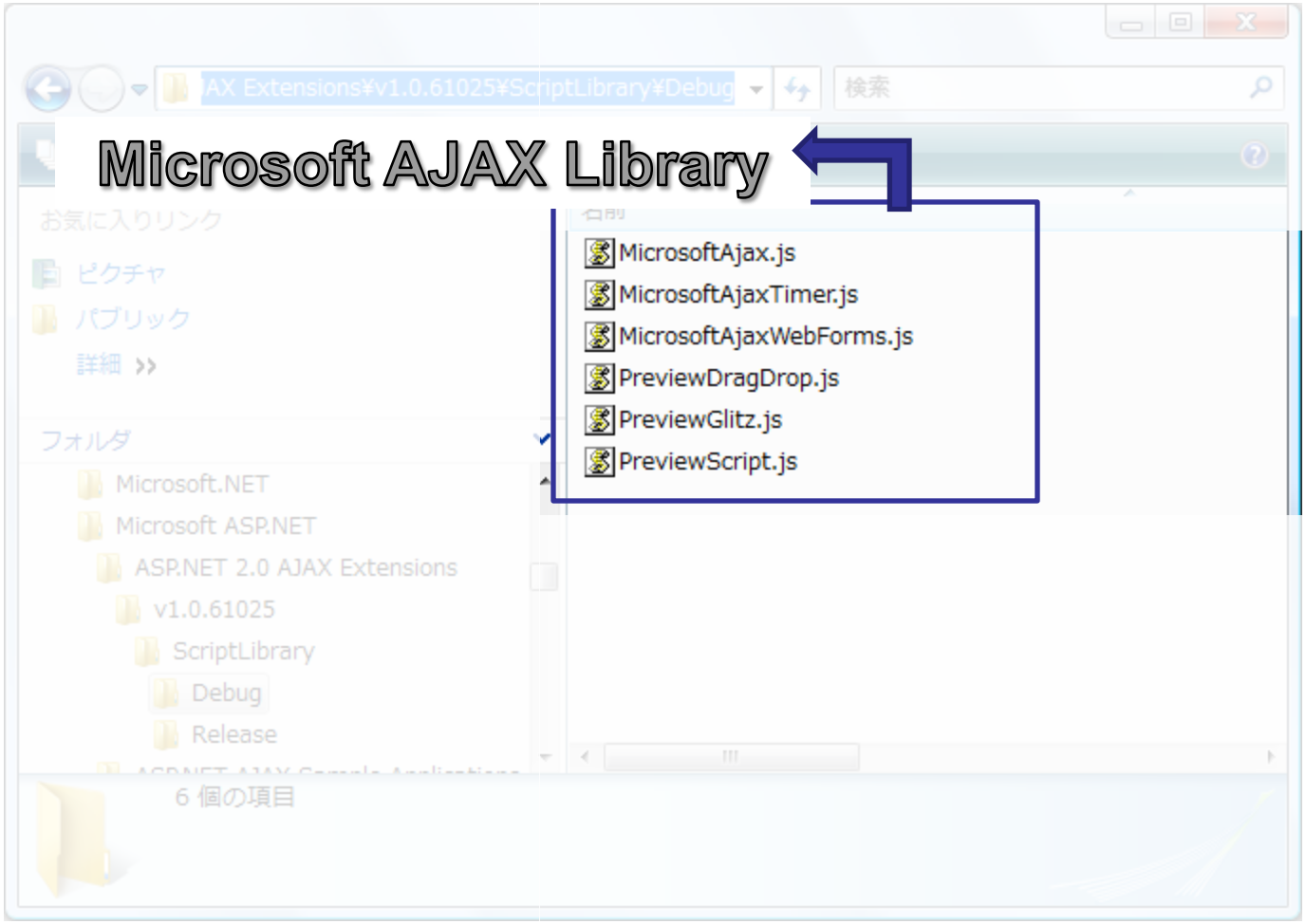
このHandlerが必要なJavaScriptのファイルをクライアントに送り込んでくれる。

実体は

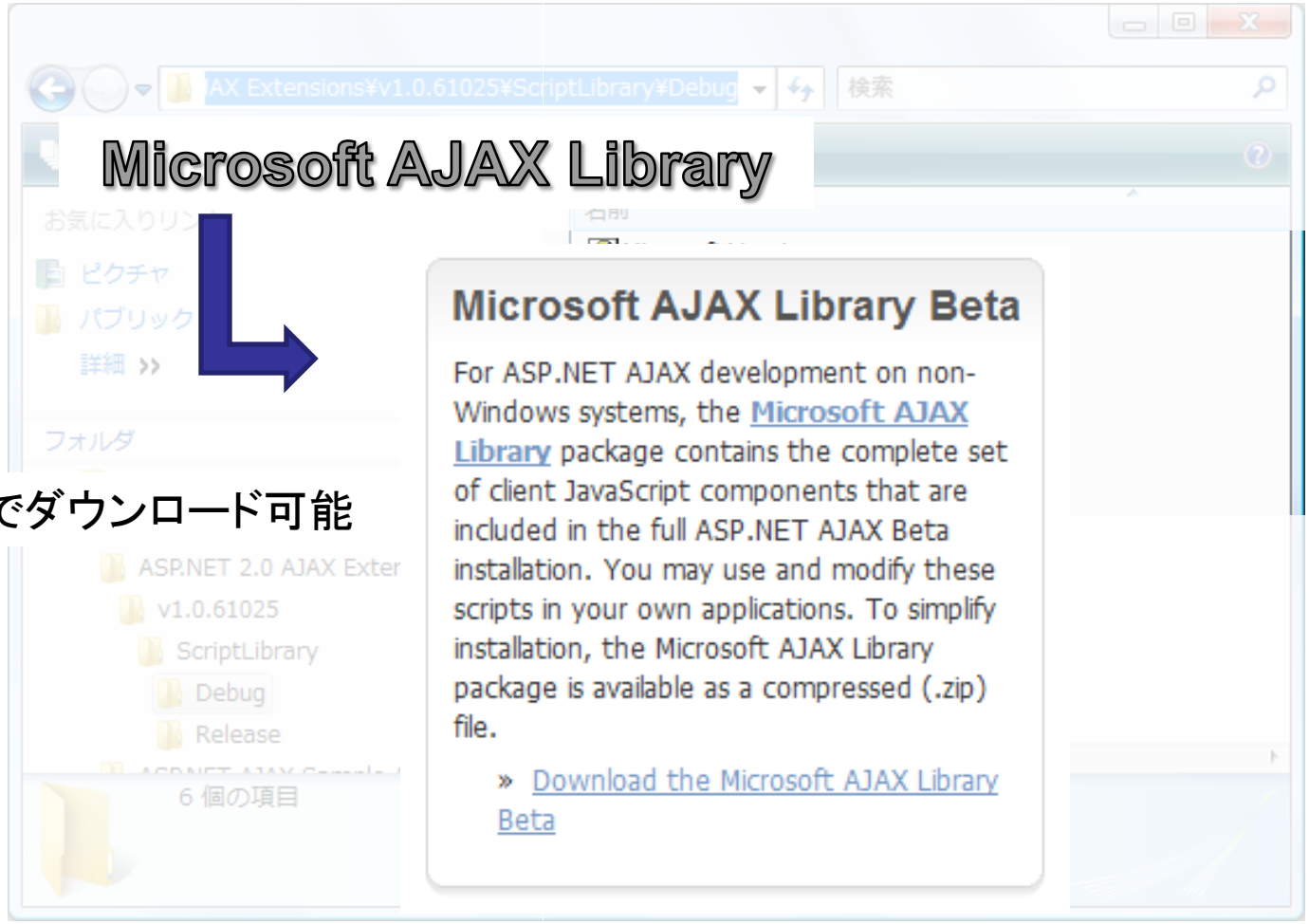


%ProgramFiles(x86)%¥Microsoft ASP.NET¥ASP.NET 2.0 AJAX Extensions¥v1.0.61025¥ScriptLibrary¥Debug

実体は


Microsoft AJAX Library
MicrosoftAjax.js
MicrosoftAjaxTimer.js
MicrosoftAjaxWebForms.js
PreviewDragDrop.js
PreviewGlitz.js
PreviewScript.js
Microsoft.NET
Microsoft ASP.NET
ASP.NET 2.0 AJAX Extensions
v1.0.61025
ScriptLibrary
Debug
Release
6 個の項目

実体は



単体でダウンロード可能

http://ajax.asp.net/default.aspx?tabid=47&subtabid=471

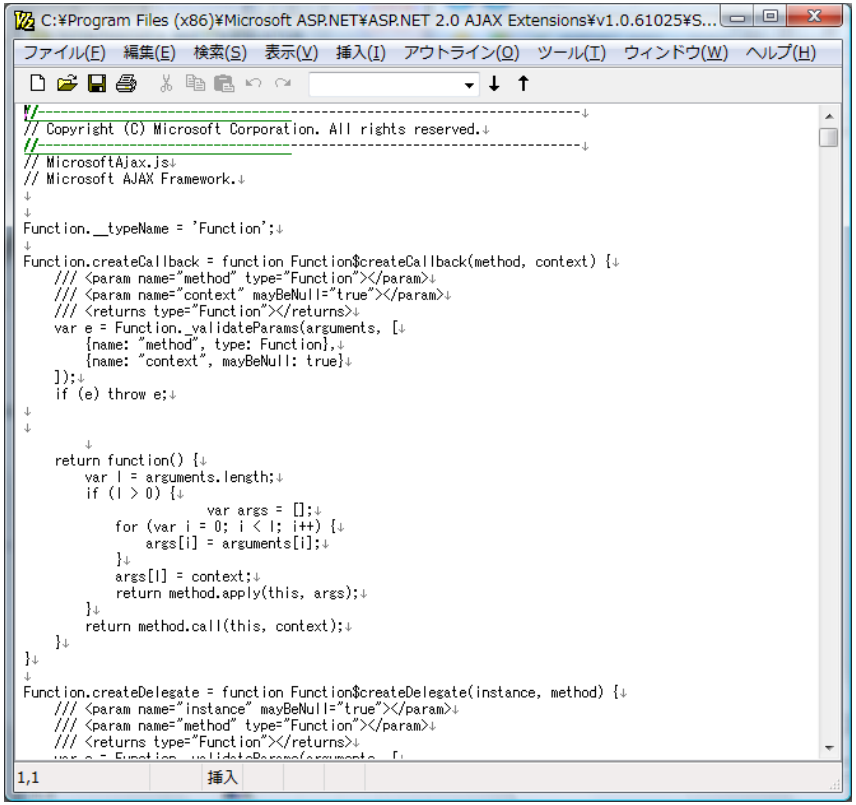
単体利用は

```
<head>
    <script type="text/javascript"
    src="Microsoft.Web.Resources.ScriptLibrary.MicrosoftAj
    ax.debug.js" />
    <script type="text/javascript"
    src="Microsoft.Web.Resources.ScriptLibrary.MicrosoftAj
    axWebForms.debug.js" />
    <script type="text/javascript">
        Sys.WebForms.PageRequestManager.getInstance()
    </script>
</head>
```

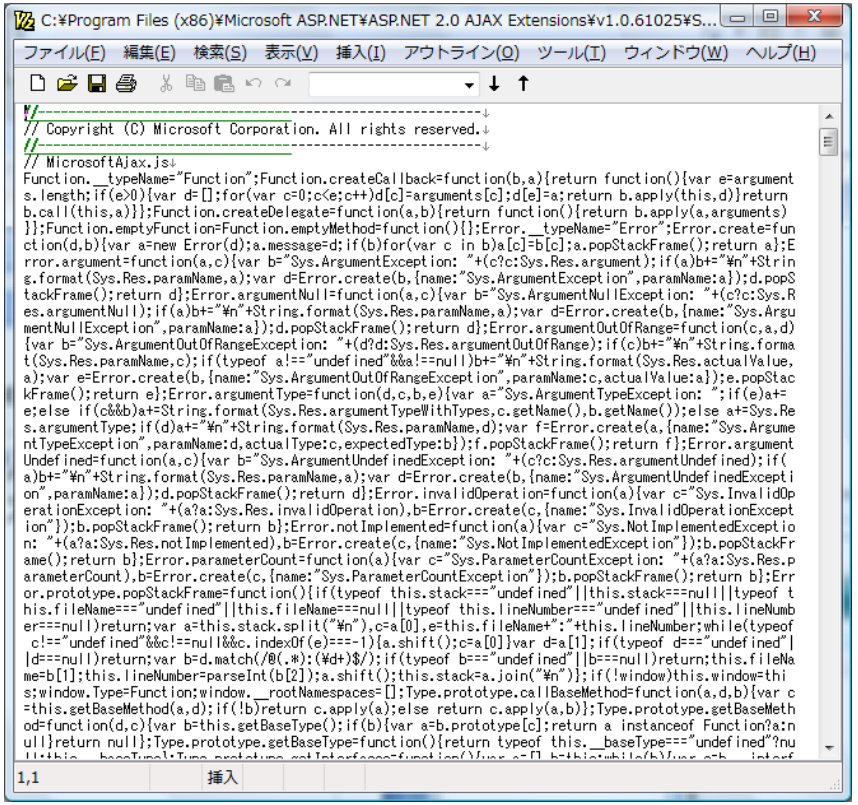
これだけでOK

Debug と Release

Debug



Release



どんなものが含まれているのか・・・1

- ベターJavaScript

Array	Date	Number
add	format	format
addRange	localeFormat Method	localeFormat
clear	Error	parse
clone	argument	Object
contains	argumentNull	getType
dequeue	argumentOutOfRange	getTypeName
exists	argumentType	String
forEach	argumentUndefined	endsWith
indexOf	create	format
insert	invalidOperation	localeFormat
parse Method	notImplemented	lTrim
queue	pageRequestManagerParserError	rTrim
remove	pageRequestManagerServerError	startsWith
removeAt	pageRequestManagerTimeout	trim
Boolean	parameterCount	trimEnd
parse	popstackFrame	trimStart
	scriptLoadFailed	

$getをつかえ

$getとは

→Sys.UI.DomElement.getElementById

の短い呼び方+α

いままでdocument.getElementById(id)を使っていたのを置換する。

→ $get(id)

あるelement配下からのみ取得する場合には

$get(“id”, $get(“parent”).childNodes)

String.Formatをつかえ

String.formatは.NETのSystem.String.Formatのようなもの。

```
String.format('NAME={0},Age={1}',
$get('TextBox1').value,
$get('TextBox3').value))
```

→

NAME=NAKA,Age=30

Sys.StringBuilderをつかえ

StringBuilderは.NETのSystem.String.Formatのようなもの。

```
var sb = new Sys.StringBuilder();
sb.append($get("TextBox1").value);
sb.append($get("TextBox2").value);
sb.append($get("TextBox3").value);
alert(sb.toString());
```

→

NAKAfalse30

まとめ

- AJAXは使うべきか？

使うべき!!

どのように

- ASP.NETのポストバックの局面
  - UpdateProgressで処理中を示せるだけでも User Experienceの向上になる
- ベターJavascript
  - $getでスクリプトの簡略化を
  - String.Formatや使えるメソッドが目白押し

Enjoy Programming!

Enjoy Community!

次は懇親会へ(^^