優れた AJAX 開発:
Windows® Internet Explorer® 8
Beta 1 for Developers

Web 作業の操作性を向上

2008 年 3 月

詳細の問い合わせ先 (報道関係者専用) :
Rapid Response Team
Waggener Edstrom Worldwide
(503) 443-7070
rrt@waggeneredstrom.com

このドキュメントに記載されている情報は、このドキュメントの発行時点におけるマイクロソフトの見解を反映したものです。マイクロソフトは市場の変化に対応する必要があるため、このドキュメントの内容に関する責任をマイクロソフトは問われないものとします。また、発行日以降に発表される情報の正確性を保証できません。

このドキュメントに記載された内容は情報の提供のみを目的としています。明示、黙示または法律の規定にかかわらず、これらの情報についてマイクロソフトはいかなる責任も負わないものとします。

お客様ご自身の責任において、適用されるすべての著作権関連法規に従ったご使用を願います。このドキュメントのいかなる部分も、米国 Microsoft Corporation の書面による許諾を受けることなく、その目的を問わず、どのような形態であっても、複製または譲渡することは禁じられています。ここでいう形態とは、複写や記録など、電子的な、または物理的なすべての手段を含みます。ただしこれは、著作権法上のお客様の権利を制限するものではありません。

マイクロソフトは、このドキュメントに記載されている内容に関し、特許、特許申請、商標、著作権、またはその他の無体財産権を有する場合があります。別途マイクロソフトのライセンス契約上に明示の規定のない限り、このドキュメントはこれらの特許、商標、著作権、またはその他の無体財産権に関する権利をお客様に許諾するものではありません。

別途記載されていない場合、このソフトウェアおよび関連するドキュメントで使用している会社、組織、製品、ドメイン名、電子メール アドレス、ロゴ、人物、場所、出来事などの名称は架空のものです。実在する商品名、団体名、個人名などとは一切関係ありません。

概要

開発者の生産性の向上は、Internet Explorer 8 Beta 1 for Developers の主要な目的の 1 つです。ブラウザ間の相互運用性、標準への準拠、およびさらに強化された API を提供することで、この目的を部分的に達成することを目指しています。

Internet Explorer 8 Beta 1 for Developers には、ブラウザ、Web ページ、およびサーバーでのやりとりを補う AJAX 開発用に簡易化されたさらに強力なプログラミング モデルが用意されています。結果的に、非常に優れたエンドユーザー エクスペリエンス、機能性やパフォーマンスの向上を実現する Web ページの構築がさらに簡単になります。API は、W3C HTML 5.0 または Web Applications Working Group の標準に基づいています。AJAX の強化機能や新たな知的所有権が、最終的な Internet Explorer 8 のリリース前の標準化に利用されます。

以下の AJAX の機能向上により、ブラウザと Web ページのギャップが埋められ、さらに優れたユーザー エクスペリエンスが提供されます。

1. **AJAX ナビゲーション**により、AJAX アプリケーションを使用したまま先に進んだり前に戻ったりすることが可能となり、従来のフル ナビゲーションを実行せずにページを移動することができます。これにより、window.location.hash 値の設定、ページ内のコンポーネントを警告するイベントの起動、さらにはトラベログのエントリの作成を行うことで、Web サイトがアドレス バーのようなブラウザ コンポーネントの更新をトリガできるようにします。

2. **DOM Storage** は、キー/値のペア データの文字列を保存および取得する、使いやすいメソッドです。データを、セッションのタブ インスタンスごとに保存するか、ローカル マシンに保持することができます。これにより、ページがマシンにテキストをキャッシュとして保存することができ、あらかじめキャッシュされたデータへの高速アクセスを提供することでネットワーク待ち時間の影響が軽減されます。斬新ないくつかの使用方法が可能となります。たとえば、この機能を新しいネットワーク接続イベントと組み合わせて使用すると、コンピュータがオフラインであることが検出された場合にページがデータをキャッシュできます。

3. **接続イベント**により、Web サイトは、いつユーザーがネットワークに接続したかを確認し、接続変更の通知を受け取ることができます。

4. ブロードバンド シナリオの 2 つではなく**ホストあたり 6 つの接続**とスクリプト可能なプロパティにより、Internet Explorer 8 Beta 1 for Developers でダウンロードの並列化が可能となり、パフォーマンスの向上を実現できます。また、これにより、2 つの接続が既に存在している場合に、ホストへの要求がブロックされないことで機能性が向上します。Web サイトでは、スクリプト可能なプロパティに基づいて、サイトのダウンロードを最適化できます。

5. **XMLHTTPRequest の強化**には、必要に応じて要求をキャンセルするように設定できるタイムアウト プロパティがあるため、開発者が要求を管理しやすくなります。

クロスドメイン通信は、AJAX 開発およびマッシュアップの Web アプリケーションに欠かせない部分です。Internet Explorer 8 Beta 1 for Developers には、安全かつ簡単に実装できるクロスドメイン通信の構築に役立つ次の 2 つの機能があります。

- **クロスドメイン要求 (XDR)** を使用して、開発者はクロスサイト データの集計シナリオを作成できます。XMLHttpRequest オブジェクトに類似していますが、より簡単なプログラミング モデルの使用により、この要求 XDomainRequest は、XDR をサポートしてドメイン間でデータを利用できるように選択されたサードパーティのサイトへの匿名の要求を作成する最も簡単な方法です。コードは 3 行で構成されており、基本的なサイト間の要求を作成できます。これにより、パブリック サイト (ブログなど) のデータ集計が、高速で簡単かつ安全に行われます。

- **クロスドキュメント メッセージング (XDM)** API により、簡単かつ安全で標準化された方法を用いて IFrame を介して異なるドメインのドキュメント間の通信を行うことができます。

互換性: Internet Explorer 7 からの動作変更

ホストあたりの接続の増加と互換性の問題
ブロードバンド接続を使用するユーザー数は増加しているため、クライアント側の帯域幅が必ずしもパフォーマンスのゲーティング ファクターであるとは限りません。通常、接続を設定して要求を送信するのに必要となる時間は、個別のオブジェクトの取得にかかる時間に大きく影響します。2 つの接続の制限は、HTTP 1.1 仕様 (RFC 2068) によって設定されました。同時接続数を増加させることで、Internet Explorer 8 Beta 1 for Developers では、Web サイトがそのコストを償却し保留中のオブジェクトのリストからより迅速に次々と乗り換えることができるようになるため、ユーザーが許容できるダウンロード時間の増加につながります。このため、Internet Explorer 8 Beta 1 for Developers は、接続がナローバンドまたはブロードバンドのどちらであるかを検出して、この接続が高速接続であればホストあたりの接続数を 6 まで増やすロジックを組み込んでいます。この最大接続数は、ダウンロードだけではなく、Web サーバーへのあらゆる接続に適用されます。

Web 開発者
互換性の問題がある場合には、Web 作成者は、自分のサイトで利用できる接続数に基づいてコンテンツ配信を最適化する必要があります。このため、Internet Explorer 8 Beta 1 for Developers には読み取り専用のスクリプト可能なプロパティも組み込まれています。これは window オブジェクトに新しく追加されており Internet Explorer 8 のホストあたりの接続制限を公表します。このプロパティは、MaxConnectionsPerServer と MaxConnectionsPer1_0Server の値をそれぞれ返します。

- **window. maxConnectionsPerServer** (HTTP 1.1 サーバー用)
- **window. maxConnectionsPer1_0Server** (HTTP 1.0 サーバー用)

エンド ユーザー
ページのロード時間に関する問題を抱えているエンド ユーザーは、機能コントロール キーを使用してホストあたりの接続数を変更できます。

この機能は 以下の 2 つの方法で無効にできます。
機能コントロール キー (Internet Explorer)
機能コントロール キー **FEATURE_AJAX_CONNECTIONSERVICES** を 0 に設定すると、この機能を無効にすることができます。これにより、Internet Explorer のホストあたり 6 つの接続は、Internet Explorer の HTTP 1.1 用のホスト プロセスあたり 2 つの接続、および HTTP 1.0 用のホストあたり 4 つの接続のデフォルト設定に戻されます。

1. Regedit を起動します。
2. レジストリ キー **HKLM¥Software¥Microsoft¥Internet Explorer¥Main¥FeatureControl¥ FEATURE_AJAX_CONNECTIONSERVICES** に移動します。
3. 新しい DWORD 値「iexplore.exe」を作成し、「0000000」に設定します。

レジストリ キー (Windows)
さらに、レジストリを使用すると、すべての設定を上書きしたり、Windows (Internet Explorer を含む) のサーバーごとの接続制限をユーザー指定の制限に変更したりすることができます。キーが存在しない場合は、このレジストリの場所にアクセスしてキーを作成できます。

1. Regedit を起動します。
2. レジストリ キー **HKCU¥Software¥Microsoft¥Windows¥CurrentVersion¥Internet Settings** に移動します。
3. HTTP 1.1 用に新しい DWORD "MaxConnectionsPerServer" を作成します。
4. この DWORD の値を「0000002」に設定します。
5. HTTP 1.0 用に別の DWORD "MaxConnectionsPer1_0Server" を新しく作成します。
6. この DWORD の値を「0000004」に設定します。

機能の詳細

A. AJAX ナビゲーション

背景
AJAX を使用する大きなメリットの 1 つは、従来のページの移動を行わずにページのコンテンツを更新できることです。更新発生時に状態の保存およびページのコンポーネントの警告が行えないと、ページの移動後に更新されるのはアドレス バーや戻る/進むボタンのようなコンポーネントのみであるため、一部のシナリオで問題が生じる可能性があります。この結果、ブラウザでは、トラベログの AJAX コンテンツの変更が保存されず、アドレス バーのようなコンテンツも更新されません。このため、ブラウザが古いコンテンツで固まっているように見えることに疑問を感じているエンド ユーザーは混乱します。

多くの場合、Web サイトでは、AJAX 経由でコンテンツを更新しながら非表示の IFrame に移動することで、この制限に対処しています。ただし、これによりパフォーマンスが低下します。

IE8 モードでは、Internet Explorer が、ナビゲーションのような window.location.hash 更新を処理し、以前のドキュメントの URL を保存します。この結果として、以下のアクションが発生します。

- 以前のハッシュ フラグメントからの可能性がある以前の URL が、アドレス バー、戻るボタン、およびその他のブラウザ コンポーネントで更新されます。
- "クリック" 音は、従来のナビゲーションが発生したときと同様に再生されます。
- 新しい hashChanged イベントが起動します。

Internet Explorer では、Web ページから移動する前にハッシュ URL フラグメントも保存されます。

> W3C 標準準拠: window.location.hash 変更時の URL の保存および hashChanged イベントの起動は、W3C HTML 5 作業用ドラフト (英語) に準拠している動作です。

B. DOM Storage

背景

現在、Web ページでは、ローカル マシン上のデータの保存に document.cookie プロパティを使用しています。Web サイトで保存できるのはドメインあたり 50 のキー/値のペアのみのため、Cookie の機能は限定されています。その上、Cookie のプログラミング モデルは扱いづらく、データの Cookie 文字列全体の解析が必要となります。

W3C の HTML 5 Document Object Model (DOM) Storage オブジェクトには、キー/値のペアの文字列データ用の非常にシンプルなグローバルおよびセッション ストレージ モデルがあります。サイトでは、タブの期間または Web サイトやユーザーがデータをクリアするまで、データを保存できます。

API の説明

Internet Explorer 8 Beta 1 for Developers は、ドメインごとにストアを作成し、以下のメソッドを含めて、エントリを設定、取得、および削除します。

以下は、ストアの作成、エントリの設定、取得、および削除を実行するコード サンプルです。

```
// 1. Create object to initialize store.Specify target domain
      var storage = window.globalStorage["http://www.contoso.com"];

// 1a.Similarly an object can be created for session storage.
      var storage = sessionStorage["http://www.contoso.com"];

// 2. Store a key-value pair
      storage.setItem("John","Public");

// 3. Retrieve value string for a given key
      var cptname = storage.getItem("John");

// 4. Remove item from store
      storage.removeItem("John")
```

メソッド

Internet Explorer 8 Beta 1 for Developers は、Web ページが継続できるように、ストアにアイテムを非同期的に書き込みます。結果的に、Internet Explorer 8 Beta 1 for Developers には、begin メソッドと commit メソッドが組み込まれるため、ストレージ記述の開始と終了をマークすることができます。これらのメソッドを使用すると、指定した開始と終了のシーケンスのすべてのアイテムをストアに書き込むかどうかを確認できます。つまり、ストレージ書き込み中にエラーが発生した場合に、ブラウザはローカル マシンのディスクにアイテムをコミットしません。

```
// 1. Begin committal process
      storage.begin

// 2. Assign one or more key and value pair strings as normal
      storage.setItem("John","Public");

// 3. Commit items to the disk
      storage.commit
```

上述の **setItem**、**getItem**、および **removeItem** に加えて、Internet Explorer には、各アイテムを反復せずにドメインごとのストア全体を削除する **clear** メソッドが組み込まれています。

プロパティ

Internet Explorer 8 では、以下のプロパティを使用してストレージ領域サイズを確認できます。

- **length** – **length** プロパティを使用すると、ドメインごとのストアにあるアイテム数を取得できます。
- **remainingSpace** – **remainingSpace** プロパティを使用すると、残りのストレージ領域が確認され、残りのバイト数が返されます。

イベント

Internet Explorer 8 Beta 1 for Developers は、ストレージ領域が変更されるとすぐにドキュメント エレメント上で **storage** イベントを起動します。Internet Explorer 8 はその後、アイテムがローカル マシンのディスクにコミットされると、ドキュメント オブジェクト上で **storagecommit** イベントを起動します。これらのイベントはキャンセルできないので、既定の操作はありません。

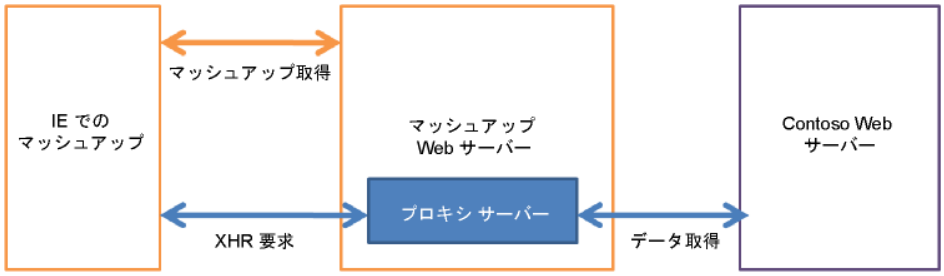
サブドメインを含めた、ドメインごとに 10 MB のローカル ストレージ領域があります。これにより、クロスドメイン攻撃の危険性が低くなります。同様に、ブラウザ タブごとに独自のセッション ストアがあります。

注: DOM Storage は、簡単にデータを保存するための Web アプリケーションの仕組みにすぎないため、データベースはありません。たとえば、値による検索のような複雑なクエリを実行することはできません。

W3C 標準準拠: globalStorage オブジェクト、sessionStorage オブジェクト、および関連メソッドは、W3C HTML 5 作業用ドラフト (英語)に準拠しています。

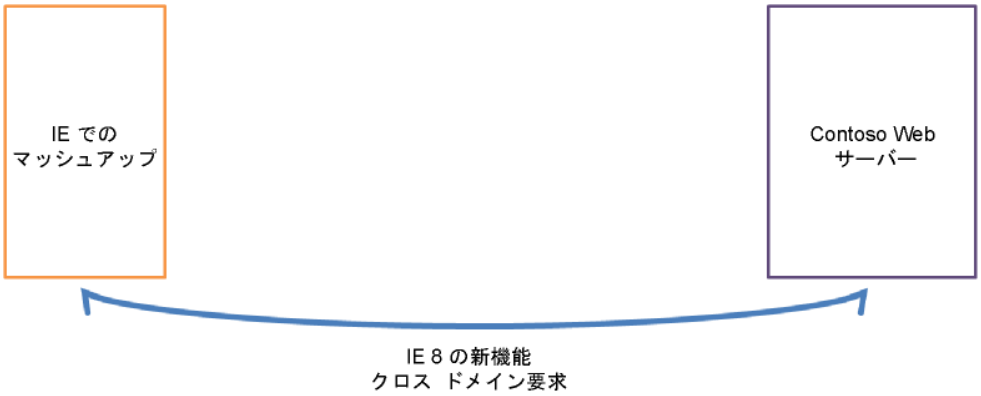
C. XDomainRequest (XDR)

背景
ブラウザには、同じサイトのオリジン ポリシーと呼ばれるセキュリティ ポリシーがあり、Web ページが他の ドメインからデータにアクセスできないようにします。多くの場合、Web サイトでは、バックエンドで他のサイトのサーバーからのサーバー要求コンテンツを受けてブラウザ内でチェックを回避することで、このポリシーに対処しています。



Internet Explorer 7 以前の場合、Web サーバーへプロキシする必要があるマッシュアップ サーバーに要求する必要があります。

Internet Explorer 8 では、サーバー間要求の代わりに新しい **XDomainRequest** オブジェクトを使用して、ブラウザ内で Web ページがクロスドメインのデータ要求を簡単に作成できます。



Internet Explorer 8 では、Web ページがサーバー間要求なしでクロスドメインのデータ要求を作成します。

クロスドメイン要求には、Web ページと Web サーバー間の相互の合意が必要です。window オブジェクトから **XDomainRequest** オブジェクトを作成して特定のドメインへの接続を開くことで、Web ページでクロスドメイン要求を開始できます。ブラウザは、XDomainRequest: 1 ヘッダーを送信することでドメインのサーバーからデータを要求します。値が 1 (true) の XDomainRequestAllowed ヘッダーを使用してサーバーが応答する場合、ブラウザは接続のみを完了します。

たとえば、サーバーの ASP ページに以下の応答ヘッダーがあります。

```
Response.AppendHeader("XDomainRequestAllowed","1");
```

> **セキュリティ上の注意** : クロスドメイン要求は、ユーザー データを保護するため、匿名です。このため、サーバーはデータの要求元を簡単に検出できません。結果として、要求と応答を実行できるのは、機密性が高くないか個人が特定されないクロスドメイン データでのみになります。

API の説明

メソッド

xdomainrequest オブジェクトを作成すると、**open** メソッドを使用して、ドメインのサーバーとの接続を開くことができます。このメソッドは、**GET** および **POST** HTTP メソッドをサポートしており、パラメータとして接続先となる URL を取得します。接続を開いた後に **send** メソッドを使用すると、必要に応じて処理のためにサーバーにデータ文字列を送信できます。以下に例を示します。

```
// 1. Create XDR object
xdr = new XDomainRequest();

// 2. Open connection with server using POST method.
xdr.open("POST", "http://www.contoso.com/xdr.txt")

// 3. Send string data to server.
xdr.send("data to be processed")
```

XDR には、アクティブな要求をキャンセルする **abort** メソッドもあります。パラメータは取得されません。データは **abort** では利用できません。

プロパティ

- **responseText** – サーバーの応答後、読み取り専用の **responseText** プロパティ経由でデータ文字列を取得できます。
- **timeout** – **timeout** プロパティを使用すると、ブラウザがサーバーの応答を待機するミリ秒数を設定または取得できます。このプロパティが明示的に設定されていない場合、Internet Explorer の既定の設定はタイムアウトなしです。要求がタイムアウトになると、データを使用できなくなります。
- **contentType** – サーバーにデータを投稿しているときに、**contentType** プロパティを使用すると、サーバーに送信されるコンテンツ タイプの文字列を定義できます。**GET** を使用している場合、このプロパティでは、コンテンツ タイプを読み取ることができます。

イベント

XDR には以下のイベントがあります。

- **onerror** – エラーの発生により要求を完了できないときに起動します。たとえば、ネットワークが利用できない場合です。
- **ontimeout** – 要求が上述の **timeOut** プロパティで定義されたタイムアウトに達したときに起動します。要求がタイムアウトになると、データを使用できなくなります。
- **onprogress** – データをブラウザに戻すことにより、サーバーが要求に応答する間に起動します。
- **onload** – クロスドメイン要求が完了してデータが使用できるときに起動します。

> **セキュリティ上の注意:** クロスドメイン要求は、以下の Internet Explorer ゾーン内の Web ページと URL の間で送受信のみ実行できます。イントラネットのデータが悪意のあるインターネット サイトに漏洩するのを防ぐために、イントラネット サイトでの XDR データの利用はお勧めしません。

		Web ページが以下のゾーンの URL からデータを要求					
		ローカル	イントラネット	信頼済み (イントラネット)	信頼済み (インターネット)	インターネット	制限付き
Web ページが以下のゾーンにある	ローカル	許可	許可	許可	許可	許可	ブロック
	イントラネット	ブロック	許可	許可	許可	許可	ブロック
	信頼済み (イントラネット)	ブロック	許可	許可	許可	許可	ブロック
	信頼済み (インターネット)	ブロック	ブロック	ブロック	許可	許可	ブロック
	インターネット	ブロック	ブロック	ブロック	許可	許可	ブロック
	制限付き	ブロック	ブロック	ブロック	ブロック	ブロック	ブロック

サーバー側

ブラウザは、XDomainRequest: 1 ヘッダーを送信することでドメインのサーバーからデータを要求します。値が 1 (true) の XDomainRequestAllowed ヘッダーを使用してサーバーが応答する場合、ブラウザは接続のみを完了します。

たとえば、サーバーの ASP ページに以下の応答ヘッダーがあります。

```
Response.AppendHeader("XDomainRequestAllowed","1");
```

これは、たとえば ASP.NET ページを使用することで、IIS で実行できます。以下のコード行を ASP ページに埋め込み、ヘッダーを返すことができます。

```
<<% Response.AddHeader "XDomainRequestAllowed","1" %>Data
```

D. クロスドメイン メッセージング (XDM)

背景
「クロスドメイン要求」のセクションで説明したように、ブラウザのサイトのオリジン ポリシーは、Web ページが他のドメインからデータを取得できないようにします。このため、単独の Web ページ上のさまざまなドメインがエクスペリエンス向上のために互いに通信できなくなります。

Web サイトでは、ネストされた IFrame を作成して URL 経由で送信されたデータを取得することで、このポリシーに対処しています。または、Web サイトでは、他のドメインからスクリプトおよび他のリソースファイルを直接ホストすることで、対処する方法もあります。この 2 番目の対処方法は、一方向の通信でのみ実行できます。また、埋め込みのスクリプトやリソースをホスト Web サイトと同じ権限で実行して、Cookies に保存されたデータなどのユーザー データへのアクセス権があるため、セキュリティ上のリスクがあります。

クロスドキュメント メッセージング (XDM) には、**window** オブジェクトの **postMessage** メソッドがあるため、さまざまなドメインで相互の同意のもとに互いに通信することができます。上述の対処方法に比べると、XDM は、双方向のクロスドメイン通信用の非常にシンプルでパフォーマンス主導のメカニズムを備えています。

API の説明

メソッド
postMessage メソッドを使用すると、他のドメインにメッセージを投稿できます。このメソッドは、必須パラメータとしてデータ文字列を、オプションのパラメータとして ターゲット URL を必要とします。ターゲット URL には、メッセージの投稿先となる URL のスキームおよびドメインが含まれます。**postMessage** にターゲット URL 文字列を含めた場合、Internet Explorer では、ターゲット URL によってホストされるウィンドウにのみメッセージを投稿します。これにより、ユーザーがウィンドウから移動する場合に、他のドメインがメッセージを受け取れないようにします。

イベント
XDM には以下のイベントがあります。

- **message** – このイベントは、他のドメインにより自分のコンテンツが取得するメッセージが投稿されたときに起動されます。

プロパティ

- **domain** – この読み取り専用のプロパティには、自分のコンテンツにメッセージを投稿するドメインの文字列が含まれています。IE では、**postMessage** を呼び出すドキュメントのドメインにこのプロパティが設定されます。
- **scheme** – この読み取り専用のプロパティには、HTTPS または HTTP のような、メッセージを投稿するウィンドウの URL スキームが含まれています。IE では、postMessage() を呼び出すドキュメントのスキームにこのプロパティが設定されます。
- **data** – このプロパティには、メッセージ イベントによって投稿されるデータ文字列が含まれます。
- **source** – このプロパティは、戻りメッセージを受信するために、window オブジェクトに設定されます。

  セキュリティ上の注意: postMessage() を送信するドメインをチェックして、このドメインがデータの投稿元のドメインであることを確認することをお勧めします。

以下のサンプルは、2 つのウィンドウの 2 つのドメイン間の通信の例です。

ページ A

```
// 1. Create event handler for message event.
<document.onmessage = HandleMsg()>;

// 2. Post message to a secure page B.
window.postMessage(“Hello world”, “https://lucernepublishing.com”)
```

ページ B

```
// 3. Create event handler for message event.
<document.onmessage = HandleMsg()>;

// 4. Create event object off window.
var e = window.event

// 5. Check domain on received event to ensure the message
// comes from the expected domain.
if (e.domain = “contoso.com”)
if (e.scheme = “HTTPS”)

// 6. Retrieve data from event.
var data = e.data

// 7. Send return message to Page A.
e.source.postMessage(“Hello”)
```

HTML 5 準拠: **postMessage** メソッドおよび関連するプロパティとイベントは、最新の W3C HTML 5 Working Draft に準拠しています。

postMessage のターゲット URL パラメータおよび **scheme** プロパティは、まだ HTML5 に準拠していません。

E. 接続イベント

背景

Internet Explorer 8 (Vista 用のみ) では、**window.navigator.onLine** プロパティおよび **online/offline** イベントによってブラウザがオンラインまたはオフラインであることを Web ページが検出できます。この情報を活用すると、DOM Storage オブジェクトを使用して優れたオフライン シナリオを実現することができます。たとえば、ユーザーが自分の Live メールのページにログオンした状態で接続が失われた場合に、ページでは、常に DOM ストアに下書きを保存するようにユーザーに通知できるため、ユーザーがメールの編集を継続することができます。接続性がリストアされると、スクリプトは、電子メールを取得してサーバーに送信することができます。

API の説明

```
// Add event handler for online and offline notification.
<body ononline="online()" onoffline="onoffline()">;

// Find out if browser is online.
online = window.navigator.onLine;
```

> W3C 標準準拠: オンラインおよびオフラインのイベントは、
> W3C HTML 5 作業用ドラフト (英語) に準拠しています。

F. ブロードバンドのホストあたり 6 つの接続のメリット

背景

ブロードバンド接続を使用するユーザー数は増加しているため、クライアント側の帯域幅が必ずしもパフォーマンスのゲーティング ファクターであるとは限りません。通常、接続を設定して要求を送信するのに必要となる時間は、個別のオブジェクトの取得にかかる時間に大きく影響します。同時接続数を増加させることで、Internet Explorer 8 では、Web サイトがそのコストを償却し保留中のオブジェクトのリストからより迅速に次々と乗り換えることができるようになるため、ユーザーが許容できるダウンロード時間の増加につながります。このため、Internet Explorer 8 は、接続がナローバンドまたはブロードバンドであるかを検出して、この接続が高速接続であればホストあたりの接続数を 6 まで増やすロジックを組み込んでいます。

Web 作成者は、自分のサイトで利用できる接続数に基づいてコンテンツ配信を最適化する必要があります。このため、Internet Explorer 8 には、ホストあたりの接続制限を公表するスクリプト可能なプロパティも組み込まれています。

API の説明

window オブジェクト

- **window** オブジェクトが新たに追加されました。詳細については、MSDN の このページ (英語) を参照してください。
- DOM は、以下の名前の読み取り専用のゲッターを介してこれらの値を取得します。

  window. maxConnectionsPerServer (HTTP 1.1 サーバー用.)
  window. maxConnectionsPer1_0Server (HTTP 1.0 サーバー用.)
  MaxConnectionsPerServer および MaxConnectionsPer1_0Server の値をそれぞれ返します。

G. XMLHttpRequest の強化

背景

高度な AJAX アプリケーションでは、AJAX の要求の詳細な制御が必要となる場合があります。**XMLHttpRequest** は、Outlook Web Access でページを更新する必要なく電子メールを表示可能にすることを主な目的として、約 10 年前に考案されました。

XMLHttpRequest の作成以降、XMLHttpRequest の導入は、機能上の設計は変更されないまま、本来の設計の範囲を超えて拡大されてきました。タイムアウト設定で **XMLHttpRequest** を更新すると、ページで定期的に XMLHttpRequest を使用して応答テキスト用のサーバーのポーリングを行い大きな遅延が発生した場合に、データが表示されなくなります。さらには、サイトでは、Internet Explorer 8 より前のバージョンではホストあたり 2 つの接続のみ許可されており、Internet Explorer 8 ではナローバンド シナリオ用に 2 つの接続をサポートし続けているため、新しい **XMLHttpRequest** オブジェクトを作成できません。

タイムアウトのサポートでは、Web 開発者が値をプリセットして、オブジェクトを中止して接続を解放することができます。

API の説明

```
// Add event handler for XHR Timeout.
xmlHttp.ontimeout=timeoutFired;
// Set the XHR Timeout value to 5 seconds.
xmlHttp.timeout = 5000;
```