

Adobe Edge Preview 6 が発表されたので、いろいろと試してみました。本レポートでは、Edge のコンテンツを JavaScript で制御する方法を中心にまとめています。

裏を取っておらず、「何故?」「Edge 爆発しろ」と思った動作やコード、そして資料を眺めて「こんなかなあ」という感覚でまとめた物ですので、このレポートの内容がきちんと Edge の仕様に沿っているかは分かりません。悪しからずご了承ください。

1. Edge の基礎知識

まずは「Edge」ってどんな物作るの? という所から確認してみましょう。

HTML5+CSS3+JavaScript なコンテンツを作成

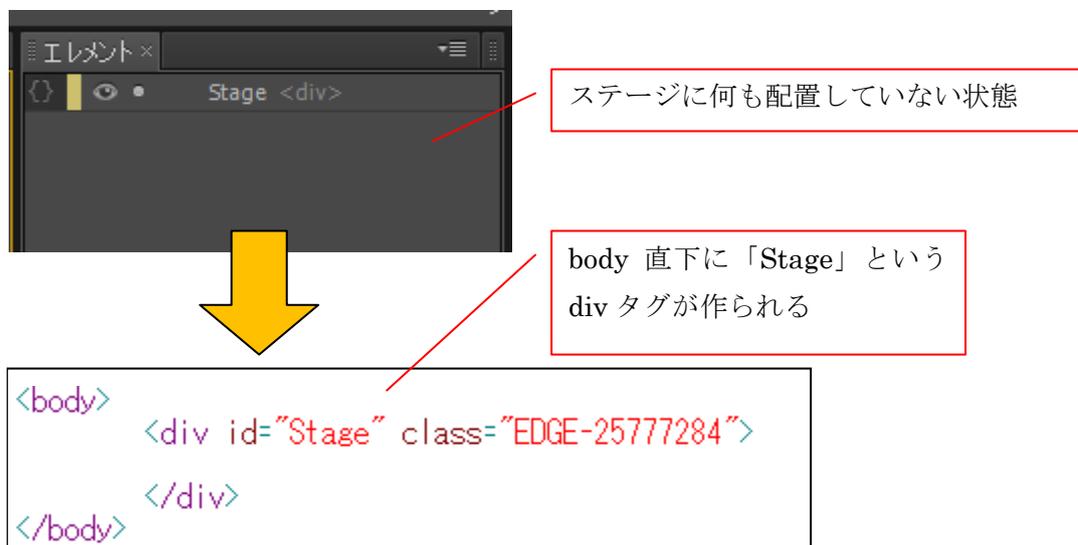
Edge は、Flash のような手順で作成したタイムラインベースのアニメーションを、HTML5+CSS3+JS を使ったコンテンツとして書き出す事ができます。つまり、iPad や iPhone でも見られる物が書き出せるというわけです。

アニメーション部分は、JavaScript で制御していますが、コアとなるのは毎度おなじみ jQuery (バージョンは、1.7.1) が使われています。この jQuery に、Edge 独自の拡張を行ってタイムラインベースのアニメーションを作成しています。

具体的にはどんな風の実現しているの

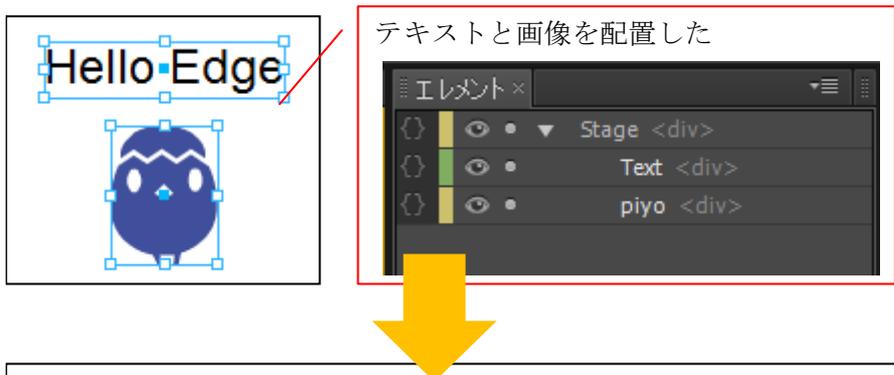
もう少し具体的に探ってみましょう。Edge では、アニメーションを行うために、HTML の DOM ツリー内の body 要素の子として、ひとつの div 要素を追加します。この要素が Flash で言う所の「ステージ」になります。

■図1. body の下にコンテンツを表示する div 要素を配置している



さらにステージ上に画像やテキストを配置すると、それらをステージ内の div 要素として配置し、配置した要素を CSS によってアニメーションさせています(要素の種類は変更もできます)。

■図2. Stage の div 要素の下にさらに div 要素を配置している



テキストと画像を配置した

```
<body>  
  <div id="Stage" class="EDGE-25777284">  
    <div id="Stage_Text" class="edgeLoad-EDGE-25777284">Hello Edge</div>  
    <div id="Stage_piyo" class="edgeLoad-EDGE-25777284"></div>  
  </div>  
</body>
```

Stage の下に、テキストと画像用の div 要素が追加される。
これを CSS によってアニメーションさせている。

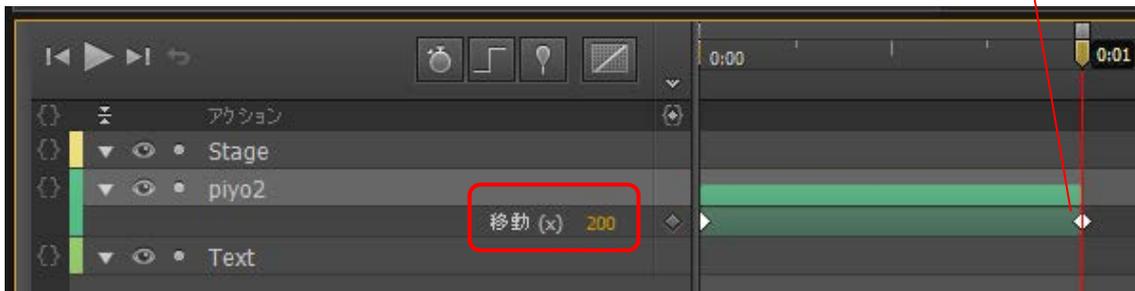
この Edge によって body 内に配置される div 要素の事を、「エレメント」と呼ぶことにしましょう。

Edge ではエレメントをどのようにアニメーションさせているかというと、CSS の transform の仕組みを使っているようです。top や left を動かさずに、transform ベースで動かしているわけですね。

※transform の資料 <https://developer.mozilla.org/ja/CSS/transform>

そのため、Edge での移動というのは、相対的な座標での移動がベースの考え方となります。「まず、エレメントの初期位置をきっちりと決め、そこから相対的にどう動かすのか」、といった形で管理しています。私は、Edge のタイムラインの数値が実値とズレてる？と感じていたのですが、この「相対的」な所を見落としていました。

■図3. 初期位置をベースに、相対的な座標等でアニメーションを管理している



例えば、最初に x 座標 100 に配置した画像を、x 座標 300 まで動かすアニメーションを作成すると、Edge 的には、「x 座標を基準点(100)から 200 だけ動かしたアニメーション」として記録されます。

このため、現バージョンでの Edge では、最初にライブラリから画像をステージに配置する操作が、大切な操作になっています。この時点で配置した場所が、その要素の「基準座標」となるからです（基準座標を手軽に再設定できる方法があるかもしれませんが、見つけれませんでした）。

ともあれ、このような形で Edge はタイムライン上のアニメーションを HTML5+CSS3+JavaScript で実現しています。

エレメントとシンボル

Edge には、エレメントの他に、「シンボル」という仕組みも用意されています。Edge のタイムラインには、レイヤー分けされた複数のエレメントを配置し、アニメーションを付加していきますが、この「タイムライン+複数のエレメント」をひとつの構成単位として捉えるのが「シンボル」です。

シンボルとエレメントの関係は、Flash でいうところのムービークリップとレイヤー毎に配置している絵やテキストの関係と同じですね。

シンボルを作成すると、[ライブラリ]内の「シンボル」欄に登録され、そこからドラッグ&ドロップすることで他のシンボルのタイムライン上へと入れ子状に配置できます。

■図4. シンボルの登録



Flash のムービークリップによく似た「シンボル」を登録し、他のシンボルのタイムライン上に配置して利用できます。

Edge では、シンボルのタイムライン上に、エレメント、もしくは入れ子となるシンボルを配置してアニメーションを付けていくわけですね。

ちなみに、シンボルもひとつのエレメントと対応しており、DOM 的な表現はこんな感じになります。

■図5. シンボルを配置した場合の DOM ツリー

```
<body style="margin:0;padding:0;">
  <div id="Stage" class="EDGE-13586389">
    <div id="Stage_kimonoGirl" class="edgeLoad-EDGE-13586389">
      <div id="Stage_kimonoGirl_hair"></div>
      <div id="Stage_kimonoGirl_neck"></div>
      <div id="Stage_kimonoGirl_kimono"></div>
      <div id="Stage_kimonoGirl_leftEar"></div>
      <div id="Stage_kimonoGirl_face"></div>
      <div id="Stage_kimonoGirl_rightEye">
        <div id="Stage_kimonoGirl_rightEye_leftEye"></div>
        <div id="Stage_kimonoGirl_rightEye_rightEye02"></div>
        <div id="Stage_kimonoGirl_rightEye_rightEye03"></div>
        <div id="Stage_kimonoGirl_rightEye_rightEye04"></div>
      </div>
      <div id="Stage_kimonoGirl_leftEye">
        <div id="Stage_kimonoGirl_leftEye_leftEye"></div>
        <div id="Stage_kimonoGirl_leftEye_rightEye02"></div>
        <div id="Stage_kimonoGirl_leftEye_rightEye03"></div>
        <div id="Stage_kimonoGirl_leftEye_rightEye04"></div>
      </div>
      <div id="Stage_kimonoGirl_mouth"></div>
    </div>
  </div>
</body>
```

シンボルとして div 要素が追加され、シンボル内のエレメントはシンボルの div 要素以下に配置されます

DOM ツリーの入れ子の親となっている部分が、シンボルに対応したエレメントです。

Edge に最初から配置されている、いわゆる「メインのタイムライン」も、ひとつのシンボルとなっています。

JavaScript で Edge のコンテンツを制御する際に押さえておきたいポイント

さて、以上の事を踏まえて、JavaScript を使って Edge のコンテンツをコントロールしておく際の2つのポイントを押さえておきましょう。

1. 各エレメントの DOM 上の表現は、body 直下に配置される div 要素によって管理されている
2. ひとつのタイムライン上の複数のエレメントは、ひとまとめのシンボルとして管理されている

そして最後にもうひとつ、

3. Edge は、jQuery を利用（拡張）したもの。

ここを押さえておきましょう。

2. Edge の構成物と基本的な操作方法

DOM 的な仕組みを押さえた所で、今度は JavaScript 的な構成との組み合わせを見ていきましょう。

Edge コンテンツを構成する4つのモノ

Edge のコンテンツは、JavaScript 的には3つのモノ、そして HTML の DOM 的には複数のエレメントで構成されています。

■図表1. Edge を構成している4つのモノ

本レポート内での名称	用途
Symbol クラス	JS 的にひとつのタイムラインを持つ「シンボル」を扱うクラス。 Flash でいうところの MovieClip。
Stage(ステージ)	JS 的にステージを扱う特殊なシンボルのインスタンス。Edge の最上位の Symbol クラスのインスタンス。 通常はここでキーボードイベント等の管理を行う。 Flash でいうところの「メインのタイムライン」。
エレメント	HTML5 の DOM 的に、body タグ内に追加された div 要素。画像やテキストの構成単位となる。「表示できるもの」は全部これ。
Composition クラス	JS 的に Edge のコンテンツ全体を統括するクラス。 内部的に Edge コンテンツの初期化や組み立てを行っているが、コンテンツ単体を操作する際には、そんなに利用する機会はない。 複数の Edge コンテンツや Edge 外のコンテンツと連携する場合は、窓口的な役割となる。

主に操作する事になるのは、「Symbol クラス(のインスタンス)」と「エレメント」です。

Symbol クラスは Edge の「ひとつのタイムラインと、その上にある入れ子のシンボルやエレメント」、つまりシンボルを管理するクラスです。Flash でいうと「MovieClip」にあたるクラスになります。

そして、「ちょっと特殊な Symbol のインスタンス」と言えるのが、Stage です。Stage もひとつのシンボル

なのですが、少し特殊な扱いとなっています。Edge を立ち上げて、最初に配置されているタイムラインが、この Stage にあたります(「Stage クラス」ってのは無いんですよ)。

私たちがコンテンツの構成を考える際に、「ステージ」という概念があった方がいろいろと連想しやすいだろうということで、特別枠で用意してくれているのしょうね。おそらくはマスク的というか、下敷きのというか、そんな役割もしていそうです。

エレメントはもう OK です。body タグ内に配置された div タグです。JavaScript からエレメントを操作するには、jQuery の力を借ります。

最後に、クラスにはもうひとつ、「Composition クラス」というものもあります。これは、Edge のコンテンツ全体を管理しているメインのクラスなのですが、私たちが Edge 内のコンテンツを操作する際にはあまり使用しませんので、今回はスルーします。

ひとつのシンボルの構成

それでは、基本となる Symbol クラスの仕組みを見ていきましょう。先ほど見たように、Symbol クラスは必ず一つのエレメントと関連付けられています。

Symbol クラスのインスタンスをひとつ生成するという事は、ひとつの対応する Edge エレメントも一緒に生成する事になります。メインのタイムラインで言えば @id の値に「Stage」を持つ div タグがこれに当たります。

■図6. Symbol クラスと Edge エレメントの関係

```
<body>  
  <div id="Stage" class="EDGE-25777284">  
    <div id="Stage_Text" class="edgeLoad-EDGE-25777284">Hello Edge</div>  
    <div id="Stage_pivo" class="edgeLoad-EDGE-25777284"></div>  
  </div>  
</body>
```

Symbol クラスはこのひとかたまりを扱います

Edge を開き、テキストと画像を一つずつタイムラインに配置したとします。この時、JavaScript 的な視点からこのコンテンツの構成をみると、「2つのエレメント(テキストと画像)を持つ、ひとつのシンボルインスタンス (Stage) が作成されている」というように見なされるわけですね。

タイムラインがシンボル、表示されている物がエレメントなので、Edge の JavaScript では、

- タイムラインを操作したい場合は Symbol クラスのインスタンスを操作する
- 表示されている物の位置などを操作したい場合には、エレメントを操作するという仕組みになっています。

Symbol クラスのメソッド

では、この関係を踏まえて、Symbol クラスに用意されているメソッドを見ていきましょう。

ほぼ一つだけのクラスということもあり、なおかつプレビュー版でもある事から、にぎやかで、後で絶対変更ありそうな雑多な構成になっています。

■図表2. Symbol クラスのメソッド

メソッド名	用途
jQuery 連携系	
\$()	おなじみ jQuery ハンドル 詳しくは後述
タイムライン操作系	
play(フレーム番号/ラベル)	指定のフレーム番号/ラベルから再生する ※もうひとつ引数があるのだけれども保留
stop(フレーム番号/ラベル)	指定のフレーム番号/ラベルに移動し、停止する
playReverse(フレーム番号/ラベル)	指定のフレーム番号/ラベルから逆再生する
getPosition()	現在の再生ヘッドのあるフレーム番号を取得
getDuration()	全体のフレーム数を取得
isPlaying()	再生中かどうかを判定(逆再生も含む)
isplayDirectionReverse()	再生方向が逆向きかどうかを返す
getLabelPosition(ラベル名)	ラベルの表すフレーム番号を取得
作成/削除系	
createChildSymbol(シンボル名, 親 Edge エLEMENTのセレクト, インデックス)	子となるシンボルを生成する 引数に癖がある。 この「シンボル名」は、Edge の[ライブラリ]内の「シンボル」欄に登録されているシンボルの名前
deleteSymbol()	自分を消す 関連付けられている Edge エLEMENTも消去される
シンボル選択系	
getParentSymbol()	親階層のシンボルを返す
getSymbol(シンボル名)	子の中から指定した名前を持つシンボルを返す
getChildSymbols()	子のシンボルの配列を返す
getComposition().getStage()	ステージへの参照を返す 実は 2 つのメソッドを結んだもの(getStage は Composition クラスのメソッド)。 Flash でいうところの stage プロパティや「_root」的な使い方をする
関連ELEMENT選択系	
getSymbolElement()	シンボルと関連付けられているELEMENTを返す
lookupSelector(ELEMENT名)	シンボル内に配置されている、引数と同じ名前のELEMENTの jQuery ハンドルを返す
プロパティ操作系	
getVariable(プロパティ名)	任意の名前のプロパティの値を取得する
setVariable(プロパティ名, 値)	任意の名前のプロパティに任意の値を設定する
既に廃止/置き換えされました系	
getSymbolElementNode()	getSymbolElement()になりました
getParameter()	getVariable()と setVariable()になりました

setParameter()	
show()	ありませんでした
hide()	ありませんでした

いろいろと揃っていますね。具体的な引数の設定方法などはともかく、これで **Symbol** クラスを使ってどんな事ができるのかは把握できるのではないのでしょうか。

タイムラインを操作したり、**Edge** 内の構成物の親子関係を辿ったり、実際表示されている **Edge** エレメントを取得したり、といった事ができるわけですね。

エレメントと jQuery

さて、表示されている物を扱うには、**Edge** エレメントをなんやかんや操作する事になるのですが、**Element** クラスという物は用意されていません。

エレメントに何かしたかったら **jQuery** の仕組みを使いな！というスタンスなのです。このため、エレメントに関する操作という物は、基本的には **jQuery** の文法なのです。ですので、**jQuery** に慣れている方であれば好き放題できるのですが、慣れていないと手も足も出なくなります（私もそうです）。しかも、**Edge** 独自の処理というわけでもないのに、マニュアル的な物にもあまり載らないかもしれません。

さらに、**Edge** のアニメーションのスタイル（**transform** 変化スタイル）を知っておかないと、**Edge** のやっている事と衝突してしまうかもしれません。

と、いうわけで、独断と偏見で「この辺知っておくといろいろ便利かなあ」という物をご紹介します。

■図表3. エレメント操作の際に便利そうな書き方

コード	用途
情報取得系	
attr(属性名)	指定した属性の値を取得
CSS(プロパティ名)	指定した CSS プロパティの値を取得
width()	幅を取得
height()	高さを取得
offset()	座標を取得 戻り値は、top と left を持つオブジェクト
html()	HTML を取得する。 Edge ではテキストに対して使って、記述してある文字列を取得できたりします。
CSS Transform 系	https://developer.mozilla.org/ja/CSS/transform あたりを参照
情報設定系	
attr(属性名, 値)	指定した属性の値を設定
CSS(プロパティ名, 値)	指定したプロパティに値を設定 引数はプロパティ名:値をまとめたオブジェクトでも OK
html(文字列)	HTML を設定する テキストに対して使って、表示する文字列を設定可能
append(文字列)	終端/先頭に HTML を挿入する
before(文字列)	テキストに対して使って、文字列を追加表示可能
CSS Transform 系	https://developer.mozilla.org/ja/CSS/transform あたりを参照
アニメーション系	
show(時間, コールバック)	徐々に表示する
hide(時間, コールバック)	徐々に非表示にする
fadeIn() fadeOut() などなど	フェードインしたりフェードアウトしたり いわゆる jQuery のアニメーション効果用命令群です
animate(プロパティ名, 時間, [イージング],[コールバック])	指定した CSS プロパティを徐々に変化させるアニメーションを作成。iPhone,iPad 等は下の transition を使用した方が良いでしょう
CSS Transitions 系 CSS Animations 系	https://developer.mozilla.org/ja/CSS/CSS_transitions https://developer.mozilla.org/ja/CSS/CSS_animations あたりを参照 モバイル系デバイスでは toransitions の方がスムーズに動くようです

エレメント(=表示されている物)に対しては。このあたりを使っていけば、いろいろな状態を取得したり、設定したり、アニメーションさせたりしていけます。

もちろん、この他にも HTML5+CSS3 そして jQuery の範疇でできる事であれば、すべてできますので、いろいろと試してみてください(その処理が Edge 内の処理と競合する可能性もありますが)。

3. Edge のイベント処理

Edge に用意されているイベント

Edge に用意されているイベント処理には、以下のようなものがあります。

■図表4. Edge のイベント処理

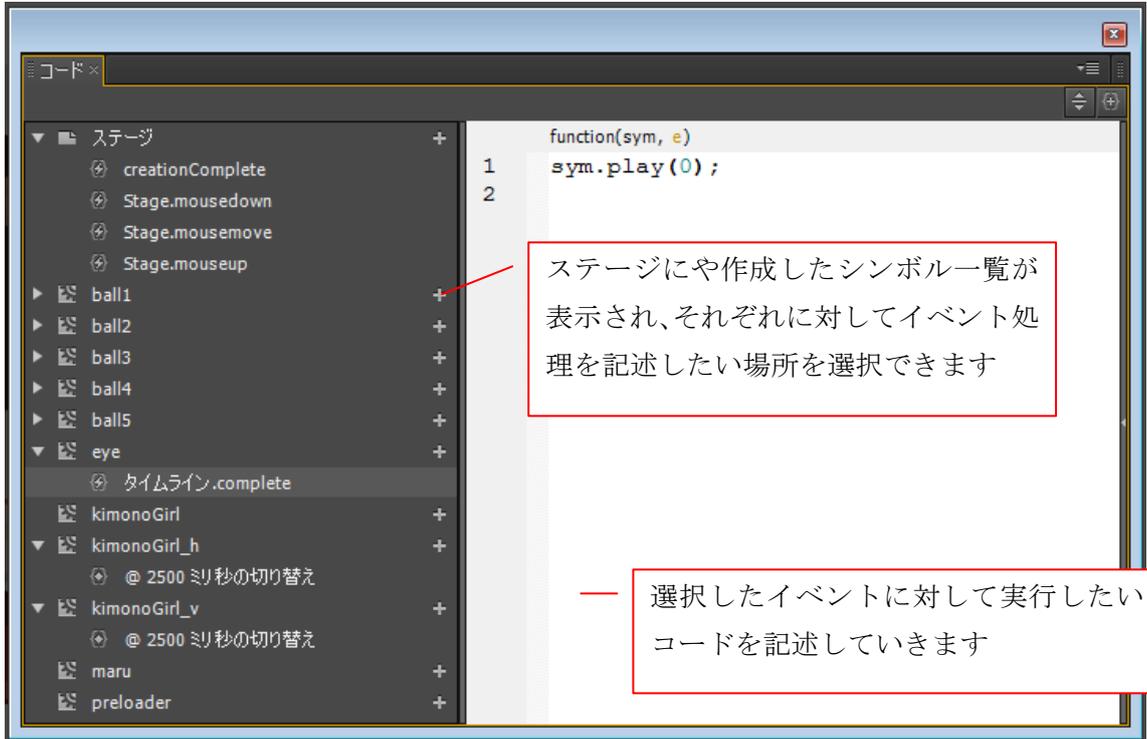
イベント	トリガー
ページレベルのイベント	
scroll	スクロール時
keydown	キーを押した時
keyup	キーを離した時
エレメントレベルのマウスイベント	
click	クリック時
dblclick	ダブルクリック時
mousedown	マウスボタンが押された時
mouseup	マウスボタンが離された時
mousuover	マウスカーソルが領域内に来た時
mousemove	マウスカーソルが領域内で移動した時
mouseout	マウスカーソルが領域内から出たとき
エレメントレベルのタッチイベント	
touchstart	タッチ操作開始時
touchmove	タッチしたまま移動した時
touchend	タッチ操作終了時
タイムラインに関するイベント	
play	再生開始時
stop	停止時
complete	タイムライン再生完了後
update	再生中毎フレームごと 「再生中だけ enterFrame」のような処理。 だいたい 60FPS で処理しようとしているっぽいです。
指定秒数に来たとき	指定秒数に来た事をトリガーにイベントを発生させることができます。
シンボル生成/消去に関するイベント	
creationComplete	シンボルが利用可能になった時。コンストラクティブに利用可能
beforeDeletion	シンボル削除時 デストラクティブに利用できます

イベントの利用方法と2つの引数

Edge のイベント処理は、Edge で編集している HTML ファイルと同階層に作成される「html 名 _edgeActions.js」という js ファイルに記録していきます。

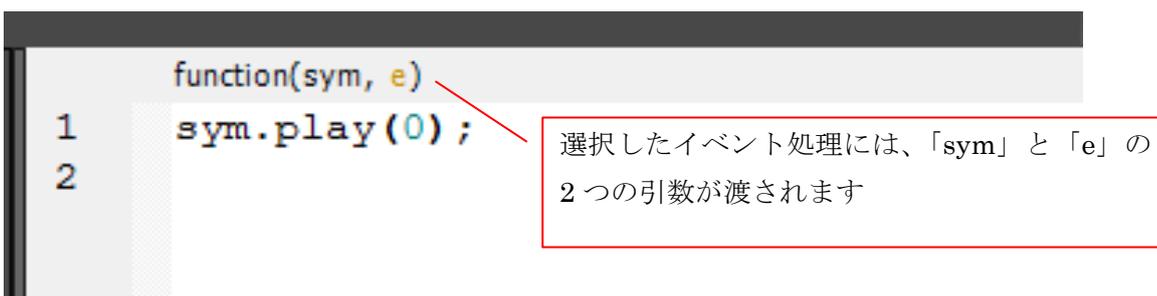
このファイルを直接編集してイベント処理を追加していくこともできますが、基本的には、Edge のタイムラインや、「コードウィンドウ」を利用して利用したいイベントを選択し、自動記録してもらうのが楽です。

■図7. タイムラインやコードウィンドウを使ってイベント処理を追加



自動的に追加されたイベント処理内では、2つの引数を受け取って利用できます。

■図8. 自動的に追加されたイベント処理と2つの引数



■図表5. 2つの引数

引数	用途
sym	イベントが発生したタイムラインを持つシンボルへの参照
e	jQuery から渡されるネイティブなイベントオブジェクトへの参照 イベントオブジェクトの内容は、イベントによって異なってくる

引数「sym」には、イベントが発生した「シンボル」への参照が格納されています。例えば、Stage 上に配置したエレメント「image1」に対してイベント処理を追加した場合、引数 sym には Stage への参照が格納されています。

また、このイベント処理内のスコープの「this」は、実は sym と同じシンボルとなります。エレメントで発

生じたイベントであれば、「this」はエレメントな気もしますが、Edge ではイベント処理内の this は、シンボルへの参照となっています。

それに対し、引数「e」には、jQuery から渡されたネイティブイベントオブジェクトが格納されています。この引数の target を調べれば、DOM 的なイベントの発生源であるエレメントも取得できるわけですね。

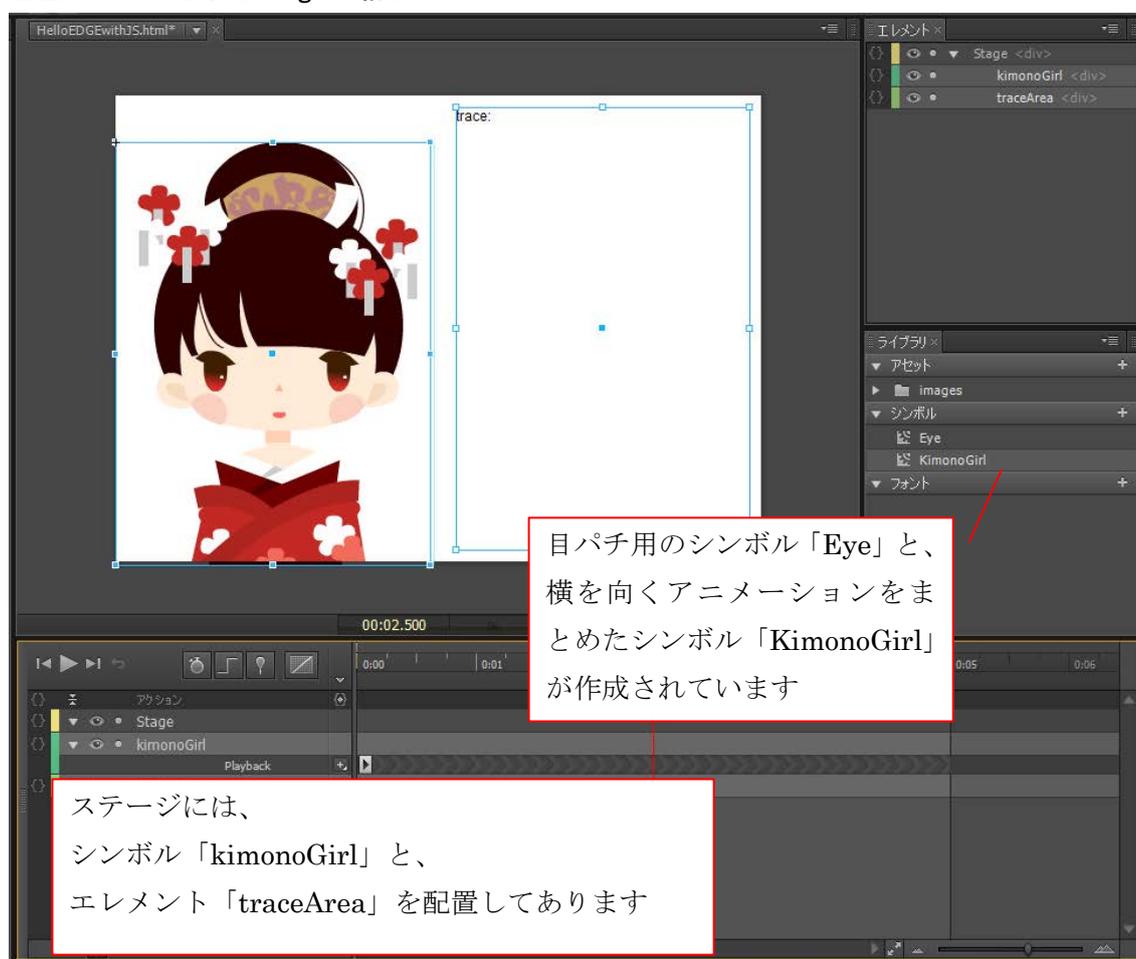
また、特に Android 端末のブラウザでは、エレメントをタップした際に「ここをタップしましたよマーク」としてオレンジ色の枠が表示されますが、これを表示させたくない場合には、ネイティブイベントオブジェクトである e を利用して、「e.preventDefault()」と記述して、既定のイベントをキャンセルしてしまう、といった用途にも利用できます。

4. 実際のコードを見て慣れてみよう

さあ、仕組みの解説はおしまいです。具体的なコードを見て、Edge のコンテンツのコントロール方法に慣れてみましょう。

今回は次のようなテスト用のエッジファイルを用意するものとします。

■図9. テストする Edge の構成



■図表6. Stage のエレメント構成

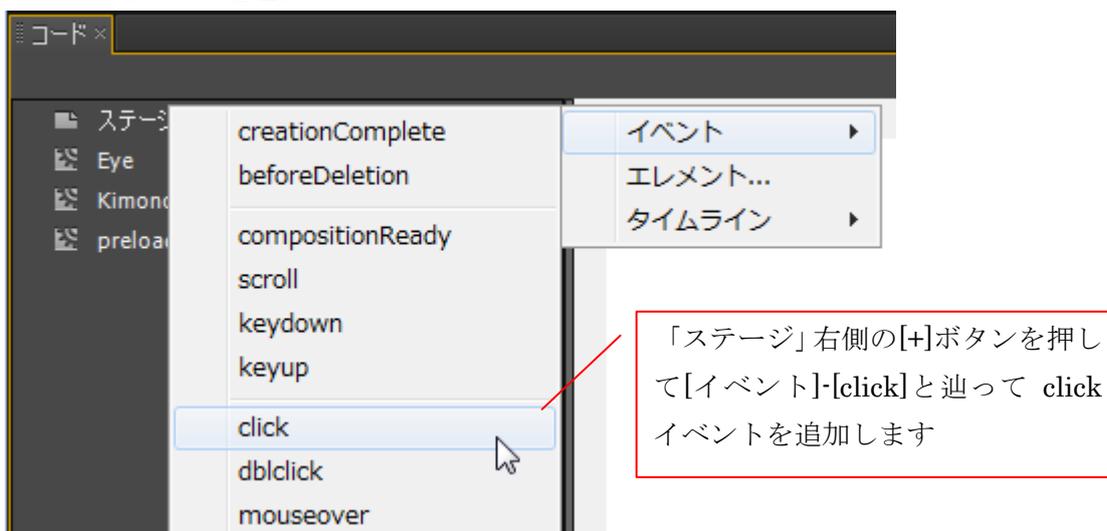
配置要素	説明
kimonoGirl	左から右へ向く 5000 フレームのアニメーションを持つシンボル「KimonoGirl」のインスタンス 中では各パーツ用の複数のエレメントを動かしています。 また、いわゆる「目パチ」用のシンボル「leftEye」と「rightEye」が入れ子として配置されています。leftEye と rightEye は、シンボル「Eye」のインスタンスです。
traceArea	いろいろ出力する用テキスト

いろいろ試す用のシンボルひとつと、コードの結果を出力する用のテキストひとつが配置してあるというわけですね。では、いってみましょう。

ステージの click イベントを利用する

まずはこれ。[ウィンドウ]-[コード]を選択してコードパネルを表示し、左側の「ステージ」の横の[+]ボタンを押し、ステージに対するイベント処理を追加します。

■図10. イベントを追加



追加するのは、[イベント]-[click]と辿って追加される「click イベント」です。
ここに次のようにコードを記述します。

■図11. コードを記述

```
function(sym, e)
1 // マウスクリックのコードをここに挿入します
2 sym.$("traceArea").html("Hello Edge from JS");
3
```

コメント部分は、Edge が自動で挿入してくれている物です。親切ですね。さて、このコードの実行結果を見たい。[ctrl]-[enter]でブラウザ上でプレビューを表示し、適当な場所をクリックすると、次のようになります。

■図12. 実行結果



ステージの click イベントを利用して、HTML 上のエレメントの内容を変更できましたね。

エレメントの選択方法あれこれ

先ほどのコードでは、イベント処理内の引数 `sym` に対して、「\$」を使用しました。`sym` はイベントの起きたシンボルですので、今回は click イベントを定義した「ステージ」への参照ですね。

次に「\$」ですが、これは「jQuery ハンドル」と呼ばれる仕組みです。「\$」を使用すると、jQuery のセレクタが使用できます(jQuery のセレクタについての説明は省きます。ざっくりと、「超簡単に DOM 内の特定の要素を取得できる jQuery の仕組み」くらいに考えておいてください)。

Edge のエレメントは、DOM 的には `body` 直下の「Stage エレメント(@id="Stage" の div 要素)」内に配置されたエレメントなので、エレメントを取得するなら jQuery の仕組みにお任せしているんでしょうね。

と、いうわけで、任意のエレメントを取得するには jQuery ハンドル経由でこんな感じで取得していきます。以下のコードは全て、ステージ上に配置されている「`traceArea`」の内容を変更します。

```
// Edge のタイムラインで指定した名前を選択
sym.$("traceArea").html("Hello Edge from JS");

// 同じ名前を lookupSelector メソッド経由で JQuery ハンドルに変換してから利用
$(sym.lookupSelector("traceArea")).html("Hello Edge from JS");

// ID 名がはっきりしている場合はこれでも OK
$("#Stage_traceArea").html("Hello Edge from JS");
```

このセレクタに関しては、正直私もまだあまり理解しきれていないのですが、こんな感じで操作したいエレメントを取得していきます。

タイムライン上へのトリガーの配置

続いて、シンボルインスタンス「kimonoGirl」の再生ヘッドが「2500」の位置に来たらメッセージを表示する処理を作成してみましょう。

シンボルインスタンス「kimonoGirl」をダブルクリックして、シンボル「KimonoGirl」のタイムラインを開き、2500 フレーム目を選択し、タイムライン上の「アクション」レイヤー上の「{O}」ボタンを押してトリガーを追加します。

■図13. 任意のフレームにトリガーを追加する



すると、コードウィンドウが開き、指定したフレーム再生時に実行したいコードを記述できるようになります。

シンボルの選択方法あれこれ

では、メッセージを表示するコードを記述していきましょう。今回コードを記述するのは、ステージに配置してあるシンボル「kimonoGirl」のタイムライン上です。このため、引数 `sym` には `kimonoGirl` への参照が格納されています。

メッセージを表示したい「`traceArea`」は、やはりステージ上に配置されていますので、`kimonoGirl` のタイムラインから見ると、親階層に配置されています。そこで、`getParentSymbol` メソッドを使ってシンボルツリーを辿り、`traceArea` への参照を取得して利用します。

```
// 親階層に配置してある「traceArea」へメッセージを表示  
sym.getParentSymbol().$("traceArea").html("2500 フレームを通過しました");
```

プレビューで結果を確認してみると、2500 フレーム通過時にメッセージが表示されることが確認できます。

■図14. 親階層にあるエレメントを操作できたところ



また、次のようにコードを記述しても、同じようにメッセージを表示できます。

```
//ステージから辿る
sym.getComposition().getStage().$("traceArea").html("2500 フレームを通過しました");

//いっそのこと id で指定して DOM 全体から探してしまう(シンボルを辿る気ゼロ)
$("#Stage_traceArea").html("2500 フレームを通過しました");
```

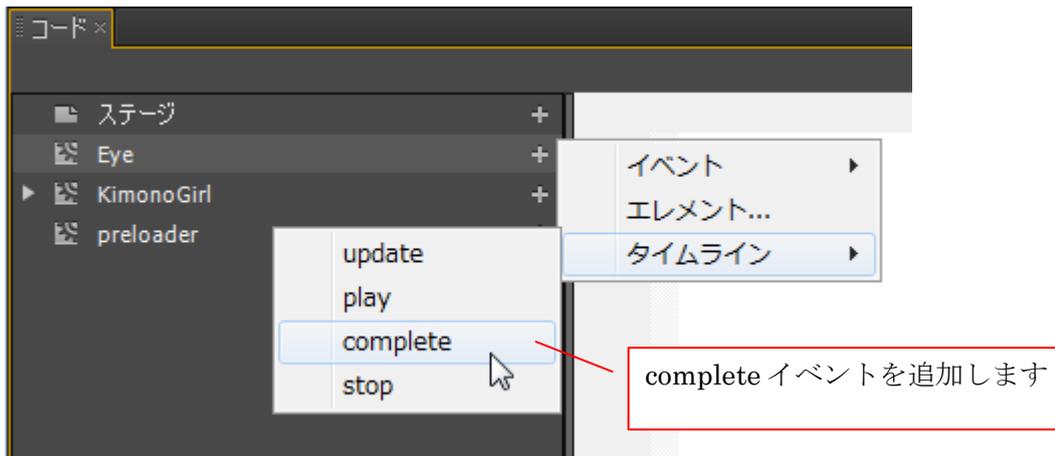
ループするアニメーションを作成する

続いて、ループするアニメーションを作成してみましょう。kimonoGirl の目の部分は、いわゆる「目パチ」アニメーションを作成してある「Eye」シンボルのインスタンスを 2 つ配置しています。Edge では、シンボルのタイムラインが最後まで再生されると、そこで停止する仕組みとなっていますので、このままでは瞬きを一回したらもう動かなくなります。

そこで、このアニメーションをループさせてみましょう。

コードウィンドウを開き、左側のペインから Eye シンボルを探して、[+]ボタンを押して[イベント]-[タイムライン]と辿って complete イベントを追加します。

■図15. タイムライン系のイベントを追加する



complete イベントを書く欄が追加されたら、次のようにコードを記述すれば OK です。

```
// 最初から再生  
sym.play(0);
```

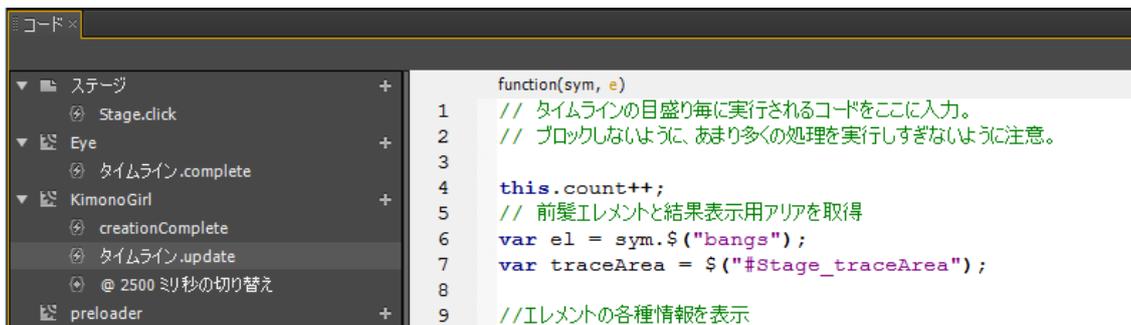
これで両目とも目パチアニメーションをループ再生してくれるようになります。

エレメントの状態を取得する

エレメントの状態を取得するには、jQuery の仕組みを使います。今回はアニメーション中にエレメントのパラメータがどうなっているのかを表示してみましょう。

まず、いわゆる「enterFrame」的な処理を行うために、KimonoGirl にタイムライン系イベントのひとつである「update イベント」を追加します。

■図16. update イベントを追加する



すると、図のような「わかってるよね？無茶すんなよ？すんなよ？」と注意書きが追加されます。親切ですね。では、無茶していきましょう。

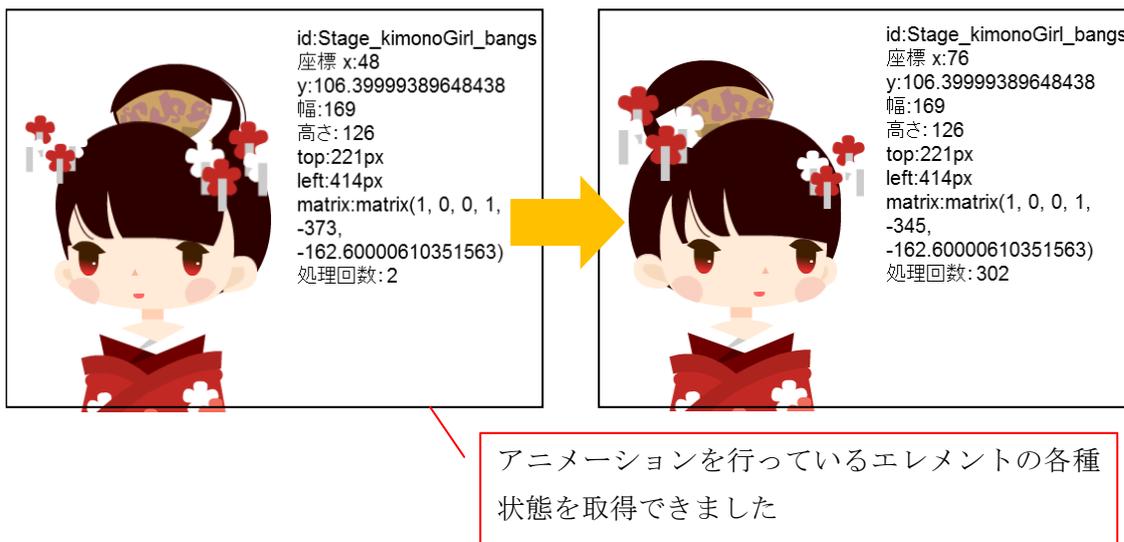
今回は、kimonoGirl 上のエレメントのうち、前髪部分のエレメントである「bangs」エレメントの値を追っていきたいと思います。

```
this.count++;  
  
// 前髪エレメントと結果表示用エリアを取得  
var el = sym.$("bangs");  
var traceArea = $("#Stage_traceArea");
```

```
//エレメントの各種情報を表示
```

```
var buf = "";  
buf+="id:" + el.attr("id")  
  +"座標 x:" + el.offset().left + " y:" + el.offset().top + "<br>"  
  +"幅:" + el.width() + "<br>"  
  +"高さ : " + el.height() + "<br>"  
  +"top:" + el.css("top") + "<br>"  
  +"left:" + el.css("left") + "<br>"  
  +"matrix:" + (  
      el.css("-webkit-transform")  
    | | el.css("-moz-transform")  
    | | el.css("-ms-transform")  
    | | el.css("-o-transform")  
  ) + "<br>"  
  +"処理回数 : " + this.count;  
traceArea.html(buf);
```

■図17. エレメントのいろいろな状態を取得していく



このように、エレメントの状態を取得するには、jQuery の仕組みで値を拾っていきます。

また、アニメーションの先頭フレームの結果と終端フレームの結果を見比べると、Edge でのアニメーションは、top や left といった基準座標を動かしているのではなく、matrix を変化させてアニメーションを行っている事が確認できます。

ついでに、5000 フレーム(5 秒)で update イベントの実行された回数を記録してみると、だいたい 300 回処理を行っていました。300/5=60 なので、Edge のアニメーションは、だいたい 60FPS くらいで動かしてそうです。

エレメントの状態を設定する

エレメントの状態を設定していくのにも、やはり jQuery の仕組みを使用します。例えば、角度を 45 度に変更するには、次のような形でコードを記述します。

```
// ベンダープレフィックスを全て記述するとわかりにくくなるので webkit だけで  
sym.$("traceArea").css("-webkit-transform","rotate(45deg)");
```

■図18. エレメントの状態を設定していく



これでいろいろな値をコントロールできますね。アニメーション用プロパティを利用すれば、動的にアニメーションを追加する事も可能です。

Edge のコンテンツは、こんな感じで JavaScript から制御していくことができます。

5. 好き放題してみよう

さて、何回か触れていますが、Edge のコンテンツは JavaScript でコントロールされており、なおかつ、そのベースのエンジンは jQuery です。ですので、JavaScript や jQuery のルールに則って、好きなように拡張してしまってもできます。

console 様をお願い

JavaScript という事は、ぼくらの味方である console 様を利用できるという事です。サンプルでは値の確認用にテキストを配置していましたが、おなじみ console.log を使えば、いろいろな情報をそのままコンソールへと表示する事もできます。

例えば、イベント内で以下のように記述しておけば、console 様がイベントオブジェクト「e」の詳細まで教えてくれます。

開発の際の強い味方になってくれることでしょう。

```
console.log(e);
```

■ 図19. コンソール様による情報確認



ただ、注意しておきたいのは、現在プレビュー版の Edge では、名前の解決ができないなどのエラー発生時には、無言で処理をストップします。そのため、原因を console 様で調べようと思っても、コード自体が走っていないというケースも出てきますのでご注意ください。

だって jQuery なんですよん

Edge のエンジンが jQuery という事は、jQuery のいろいろな仕組みをそのまま利用できるという事です。例えば、エレメントの matrix の値を得る時等、特定の CSS プロパティを利用したい際には、例のベンダープレフィックス地獄が待っているわけですが、このシンタックスシュガーをステージの creationComplete イベント内で jQuery のカスタム関数として登録してしまい、後のコード内で利用する、なんてことも可能です。

```
// matrix 取得用シンタックスシュガー
jQuery.fn.getTrans = function(){
  var el = jQuery(this);
  return   el.css("-webkit-transform")
          || el.css("-moz-transform")
          || el.css("-ms-transform")
          || el.css("-o-transform");
};

// matrix 設定用シンタックスシュガー
jQuery.fn.setTrans = function(trans){
  jQuery(this).css({
    "-webkit-transform":trans,
    "-moz-transform":trans,
    "-ms-transform":trans,
    "-o-transform":trans
  });
};
```

```
/* 他の箇所ではこんな感じで利用します */
// シンタックスシュエを使ってマトリクス取得
var mat = sym.$("traceArea").getTrans();
// ちょっと荒業ですけど、tx の値を 2 倍に
var arr = mat.split(",");
arr[4] = arr[4]*2;
mat = arr.join(",");
// シンタックスシュガーで設定
sym.$("traceArea").setTrans(mat);
```

お気に入りの jQuery プラグインがあれば、それもそのまま活用できますね。

だって JavaScript なんでもん

さらに JavaScript ベースという事は、Symbol クラスを独自に拡張して使う、なんて事もできちゃいます。イベント処理を定義している「edge_edgeAction.js」を見ると、Symbol クラスへの参照は、「Symbol」として用意されていることが分かります。

■図20. Style クラスなどはあらかじめ扱いやすいスコープに参照が用意されている

```

1  - /*****
2  * Adobe Edge Composition Actions
3  *
4  * Edit this file with caution, being careful to preserve
5  * function signatures and comments starting with 'Edge' to maintain the
6  * ability to interact with these actions from within Adobe Edge
7  *
8  *****/
9  - (function($, Edge, compId){
10 var Composition = Edge.Composition, Symbol = Edge.Symbol; // aliases for
11
12 //Edge symbol: 'stage'
13 - (function($, Edge, compId){
14

```

Symbol クラスへのショートカットがあらかじめ「Symbol」として登録されています

ですので、これを利用して、例によって Stage の creationComplete イベントで、prototype をいじるという暴挙に出てカスタムメソッドを準備してしまう事も可能といえ可能です。こんなメソッドを作成しておいて、

```
// シンボル自身に関連付けられているエレメントの ID を返すメソッド
Symbol.prototype.getID = function(){
  return this.getSymbolElement().attr("id");
}
```

他の場所でこんな風に使えるわけですね

```
console.log(sym.getID());
```

その他にも、お気に入りの JavaScript ライブラリをインポートして利用する、といった事も可能です。

さてさて、これで一通りのレビューはおしまいです。

プレビュー版と言うだけあって、まだまだ荒削りな所も多いのですが、「jQuery+CSS3+JavaScript でなんとかする」という大元の発想のおかげで、非常に自由度の高い操作も可能な作りになっていると思います。

皆さんもこの機会に Edge に触れてみてはいかがでしょうか。個人的には、Edge を通じて HTML5+CSS+JavaScript の枠組みの学習を進めるというのも、割とアリなんじゃないかな、と思っています。

ではでは。



よしおか

6. 参考資料

レポートを作成するにあたって目を通した資料はこちらになります。

- 上条さんの Edge 関連の記事
これが一番ありがたかったです。
<http://cuaoar.jp/html5/edge/>
- Adobe Edge Runtime API - Version 0.1.6 (英語)
API リファレンス
<http://www.adobe.com/devnet/apps/edge/api/current/EdgeAPI.0.1.6.html>
- Adobe® Edge preview (英語)
Edge のチュートリアル&サンプル集。ビデオ連携など、面白い物多数
<http://edge.adobe.com/showcase.html>
- Adobe Forums: スペース: Edge Preview (英語)
疑問だらけの Edge Preview コミュニティ。わかる…わかるよその気持ち。
<http://forums.adobe.com/community/labs/edge/>