CSSスプライトとBase64エンコード埋込

A Study of CSS Stripe and Base64 Encode Embedding

(2013年３月31日受理)

石原　信也　　橋本　和久
Nobuya Ishihara　Kazuhisa Hashimoto

Key words：CSSスプライト，Base64エンコード埋込，ウェブフォント，ウェブキャッシュ，manifest

概　　要

ホームページを制作する過程は，本文内容であるコンテンツの作成と見栄え部分のデザインの過程に大きく二分することができる[1]。特にデザインの過程に着目した場合,図画像を挿入するための無視できない技法がいくつかある。ページの読み込み時間を短縮することが主眼の技法だが，それを駆使するためのコードを保守性にも注意してみていく。

これらのコードが記載される場所はホームページのデザイン部門に相当する「スタイルシート」であるが，技術を駆使し高度な設計をすればするほどコードは煩雑なものになり，読み解くのが難しくなっている。本稿では画像利用で使われる技法を例示し，効果の反面読み解きやすさが犠牲になっていることを示す。

ホームページを記述するコード全体は，Web標準というルールを導入し混乱状態を脱し，更に新しいHTML5というスッテプへ移行しつつある。その中でスタイルシートの記述は整備が遅れているのだろうか。

1．は　じ　め　に

ウェブデザイナはホームページ上で画像を表示するために２通りの方法を使用してきた[2]。

一つ目は本文内容に付随した情報構造上必要な画像を表示するための方法で，もう一つは視覚的な演出効果であったり情報としての意味を持ち合わせていない画像[1]を配置するのに使用されるものである。

前者はHTMLのimg要素を使用する方法で

```
<img src="url of image" />
```

のように使用し[3]，幅や高さを指定することで拡大，縮小が可能である。

後者はスタイルシートのbackground-image属性が使用される。

```
background-image:url("url of image");
```

のように使用し，左上の位置指定や繰り返しの指定が可能である。

それぞれに特性があることから，画面の大きさに応じた背景画像の拡大縮小を行うためにbackground-image属性の代わりにimg要素を使用したり[4]，右クリックで不用意にコピーされないようにimg要素を使わず，逆にbackground-image属性で画像を表示するような例外もあるが，おおむねWeb標準の考えに沿っている。

視覚的な演出効果で画像を挿入する場合，どうしてもページが重くなりがちなため，高速に表示させる工夫が必要であった。そのためCSSストライプやBase64エンコード埋込と呼ばれる技法が使われている。これらの高速化はプログラムやコードを記述する際に重視される「可用性」の向上だろう。また別の視点として「保守性」があ

る。ホームページのコーディングでは,保守性は可読性,可用性は応答時間，言い換えるとページの重さと密接に結びつく。

一方，ウェブデザインを取り巻く環境は2012年12月にW3CはHTML5及びCanvas2Dに関する仕様策定を完了したところだ[5]。HTML5が「次世代Web標準」とか「次世代HTML標準」などと呼ばれることから判るように現段階迄はWeb標準が文字通りスタンダードとして認識されていた。Web標準ではCSS1,CSS2を導入して情報構造と表現を分離した。HTML5は「ドキュメントビューワからアプリケーションプラットフォームに」進化していくように感じられる。この流れの中でスタイルシートの技法はどのようなコードを見せているだろうか。

2．CSSスプライト

２Ｄゲームプログラミングで使用されていたスプライト表現［2]は，必要とされる画像リソースを予めメモリ上にロードし，必要な部分を描画する高速化の技法であった。又，キャラクターデザイナにとっては同一キャラクターのデザインを一覧できて管理の容易な製作方法でもあった。

ホームページ制作で使われる同様の技法を「CSSスプライト」と呼ぶ。

CSSスプライトを使った開発は，ロゴやアイコンなどを１枚にまとめ（本稿では１枚にまとめた画像をシートと略記する),CSSを読み込む時にシートを予め読み込み,画像が必要なHTML要素に割り振るという流れになる。CSSスプライトの技法はbackground-positionとbackground-imageの２つの属性が使用可能であれば実現できる。いずれも早くから実装されているため，この技法の利用の歴史も長く，支援ツールやウェブ上の支援サービスも多い[6]。

通常のホームページでは，図１に示すように，ブラウザからサーバにHTMLファイルがリクエストされ，サーバから読み込まれたHTMLを解析し必要な数だけ画像データのリクエストと受信が繰り返される。それに対してCSSスプライトでは図２のように画像データのリクエストが１度で行われる。

図１　通常の背景表示

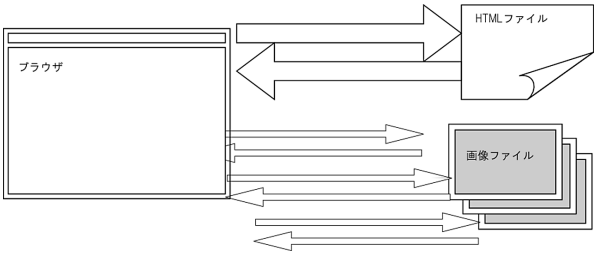
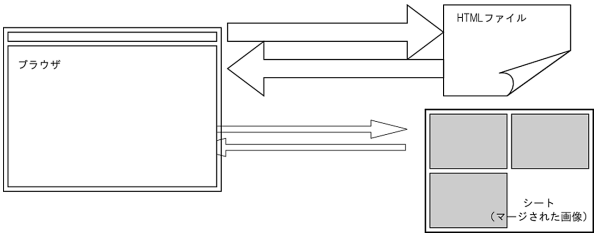


図２　CSSスプライトによる背景表示



CSSスプライトの第一の利点は高速化である。通常の場合,サーバに対して画像の要求をした後に発生する(リクエストからデータの受け取り開始までの）応答時間は要求した画像の枚数分だけ発生するCSSスプライトの技法を用いた場合はこれが１回ですむ。

図３　FireBug１

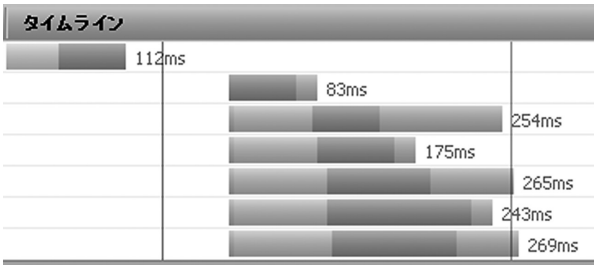


図４　FireBug２

図３，図４はFireFoxブラウザの拡張機能FireBugで表示したネットワークのトラフィックを拡大したものである。DNS問い合わせ時間などの待機時間が１度で済む。

更に利点としては，予め全ての画像が読み込まれているためマウスオーバーで画像が切り替わるといった短時間でレスポンスが必要なイベントに強い。タイムアウト

などでページ内の一部の画像が表示されないようなトラブルも回避できる。またシートによる画像の一元管理ができるためテーマ作成や適用も簡単に行えることなどが考えられる。

一方，シートは位置の指定を勘案しながら作る必要があり，CSSのコード上はHTML要素に対し位置指定のみが記述されるため，実際にどの画像が指定されているのか最終的なCSSの読み取りはシートに目盛りを当てて見る必要がある。これは相当に手間のかかる作業であるため開発用のツールはさまざま考案されている[7]。サンプルコードは以下のようになる。

```
div{
  height:94;
  width:94;
}
div:nth-child(1){background-image:url("1.png");}
div:nth-child(2){background-image:url("2.png");}
div:nth-child(3){background-image:url("3.png");}
div:nth-child(4){background-image:url("4.png");}
div:nth-child(5){background-image:url("5.png");}
div:nth-child(6){background-image:url("6.png");}
```

通常のコード。各エレメントにそれぞれ画像ファイルが指定されている。

```
div{
  height:94;
  width:94;
  background-image:url("a.png");
}
div:nth-child(1){background-position:-0px 0;}
div:nth-child(2){background-position:-94px 0;}
div:nth-child(3){background-position:-188px 0;}
div:nth-child(4){background-position:-282px 0;}
div:nth-child(5){background-position:-376px 0;}
div:nth-child(6){background-position:-470px 0;}
```

CSSストライプのサンプルコードは上のようになる。対象の要素全部に共通の画像ファイルが指定され，各要素には位置が指定されている。

CSSスプライトを使用しない場合，各エレメントに対しbackground-imageを指定することになる。ファイル名が十分に画像を表していれば可読性の確保を期待できるがCSSスプライトを使用した場合は各エレメントに対して指定されるのは位置情報のみのため可読性は劣るといえる。

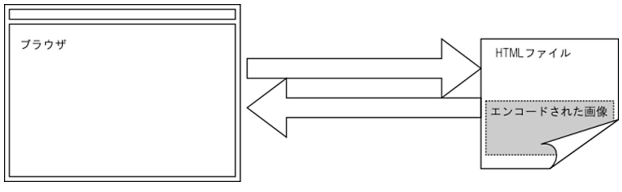
3．Base64エンコード

別の画像表示の技法として，Base64エンコード埋込がある。テキストファイルであるHTMLファイルやCSSファイルに，テキスト化したバイナリファイルを直接書き込む方式で，RFC4648等で規定されているBase64エンコードを用いる。

エンコードの仕組みはバイナリファイルを６ビットごとに取り出し該当コードの文字に置換するだけなので，簡単なスクリプトを用いても行える[8]。１文字を表現するには８ビット必要であるからデータ量は元データの30％以上の増加になる。

エンコードされたデータはHTMLあるいはスタイルシート内に直接記述することができる。

図５ Base64エンコード埋込



リソースの管理面からはすべてのデータを１つのテキストがファイルに収める事も可能で，別のファイルで画像を管理する必要がない，これは長所となる。ブラウザによってはページをローカルに保存した場合に画像データをBase64エンコードして保存するするものもある。

Base64エンコード埋込みを使用したスタイルシートのコードは以下のようになる。

```
div{
  height:94;
  width:94;
```

```
}
div:nth-child(1){background-position:-0px 0;}
div:nth-child(2){background-position:-94px 0;}
div:nth-child(3){background-position:-188px 0;}
div:nth-child(4){background-position:-282px 0;}
div:nth-child(5){background-position:-376px 0;}
div:nth-child(6){background-position:-470px 0;}
div{
background-image:url('
ORw0KGgoAAAANSUhEUgAAAgwAAABdCAIAAABGqakjAAAgAE
lEQVR4nO19TYhk1dn/mZd3YSI6I4ZRHBklDh0kCToGJiaBN
H5AMOG6SiATCNloKCELQVdSLkRxdQUhC4fSnWAJycZuopvc
kRG0kkDsapIQqmmDt9EJ1VGsEhQhi/ovfv/6+dRzznuubd
u94y+9Vtoz637cT6f7+c5R2azmVthhRVWWGGFEP7nUjdghR
VWWGGFyxcrJrHCCiussEIUKyaxwgorrLBCFCsmscIKK6ywQ
hQrJrHCCiu...
...
...1hhhRVWWCGKFZNYYYUVVlghiv8HS806c/8+x+8AAAAAS
UVORK5CYII=');
}
```

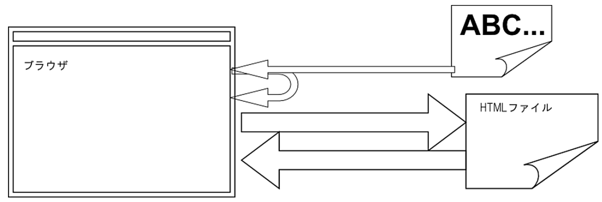
Base64エンコード埋込のサンプルコード（一部略）。コード内の「」で導かれている部分がエンコードされた画像データである。ここでは，Base64エンコード埋込に加えてCSSスプライトも併用している[9]。

バイナリファイルを強引にテキスト中に納めているためコード上に意味のとれない文字列がかなりの分量並ぶことになる。かなりの長さになるため場合によっては以降のコードのスクロールするのも一苦労になるほどで，可読性は期待すべくもない。

4．フォントアイコン

CSS3で導入された仕様であるWebフォントを使って単色のピクトグラム[10]を表示する方法である。従来はCSSスプライトで作られていたボタンの類がスマートフォンやタブレット[11]のピンチイン・ピンチアウト[12]で拡大された時に画質が心許なくなった。フォントアイコンはフォントとして給されるため，拡大してもなめらかな線を保つ。

図６　フォントアイコン



ブラウザに対して対応しているウェブフォントの形式が異なる[13]。図６のように，HTMLファイルで指定されるとフォントファイル全体が読み込まれるためサブセット化[14]しない限り，使用しない余分なデータもロードしなければならない[15]。

スタイルシートのコードは次のようになる。

```
@font{
  font-family:"myFont";
  src:url("url of webFont") type("woff");
}
.button:before{
  font-family:"myFont";
  font-size:32px;
  margin-right:32px;
}
.button{
  opacity:0.0;
}
.push:before{
  content:'a';
}
```

この設定を使って，HTML内で以下のようなコードが記述できる。

```
<span class="button push">push</span>
```

ウェブフォントをアイコンとして使用する場合，各キャラクターがどのコードに割り振られているかは，フォントによって異なる。このため，例えば文字コードコード’a’にプッシュボタンのアイコンが準備されて

いて使いたい場合，HTMLに

```
<span>a</span>
```

といったコードを書くことになる。「『HTML』には内容，『CSS』に表現」の原則から，HTML内にアイコンを示す文字を挿入するのはナンセンスでありCSSの擬似クラスのafter,beforeなどを援用するなどの工夫が必要になる[16]。

5．キャッシュ

上の３例ではスタイルシートのコードを見てきたが，一般のプログラムの場合と同様に，技法が工夫され積み重ねられる程に可読性は悪くなっている。また本稿の例では単純な画像の表示の技法を取り扱っているが，もっと複雑なケースで幾つもの効果や技法を織り込むとスタイルシートを読み解くのは煩雑になる。

本来スタイルシートの役割として高速化を担う必要があるのかも疑問ではある。

ここでは高速化をスクリプトに委ね，スタイルシートの変更は最小になるようなコードを試作してみた。

まずスタイルシート部分のコードは以下のようにした。

```
div{
  height:94;
  width:94;
}
div:nth-child(1){background-image:url("1.png+");}
div:nth-child(2){background-image:url("2.png+");}
div:nth-child(3){background-image:url("3.png+");}
div:nth-child(4){background-image:url("4.png+");}
div:nth-child(5){background-image:url("5.png+");}
div:nth-child(6){background-image:url("6.png+");}
```

何の技法も使わないレベルのコードに近い。キーワードurl()で導かれるファイル名に+マークを付加しスクリプトの対象要素であることを示した。

スクリプトではキャッシュとしてlocalStorageを使用して高速化を図った[17]。

```
$('div').each(function(){
  var fnm = $(this).css('background-image');
  fnm = fnm.replace(/^url\(\"|\"\)$/gi,"");
  fnm = fnm.replace('+',".txt");
  var st = localStorage.getItem(fnm);
  if(st != null){
     $(this).css('background-image',
     'url("' + st + '")');
  }else{
     var t = $(this);
     $.get(fnm, function(st){
     $(t).css('background-image',
     'url("' + st + '")' );
     localStorage.setItem(fnm, st);
     });
  }
});
```

スクリプトはライブラリとしてjQueryを使用している。画像ファイル名＋.txtにBase64エンコードされたテキストを準備しておく。ページの読み込みが終了したあと，すべての該当要素に対してlocalStorageをチェックして,キャッシュされていなければAjaxで読み込み,ファイル名をキーとしてキャッシュする。

6．マニフェストファイル

スクリプトでlocalStorageを使用してキャッシュを行うには，対象が文字列でなければならない。HTML5では，html要素のmanifest属性で，キャッシュコントロールに関するマニフェストファイルを指定できるようになっている。実際のマニフェストファイルは以下のようになる。

```
CACHE MANIFEST
#version 0.01
CACHE:
1.png
2.png
3.png
```

```
4.png
5.png
6.png
```

このファイルをHTMLでhtml要素に指定する。

```
<html manifest="manifest file">
```

マニフェストファイルの更新がなければ変更が反映しない[18]ので管理対象のファイルは増えるが，スタイルシートに関して言えば勿論なんの変更も必要ない。

7．終　わ　り　に

コードやプログラムを人にとって読み易い形にしていこうとする動きは普遍的で定期的に現れるようだ。アセンブラを作って機械語を置き換えることから始まって低級言語から高級言語へ移行し，またスパゲティプログラムからダイクストラらの提唱による構造化［3]プログラム，そしてオブジェクト指向も，だんだんと人にとって読みやすいコードが求められてきた。

ウェブデザインの分野で言えばWeb標準の考え方もそうした流れで捉える事ができる。「混乱したWebに，構造と表現を分離する［4]」構造を与え，見やすい形に整える方法でもあった。

2012年12月にW3CはHTML5及びCanvas2Dに関する仕様策定を完了した。ホームページはいよいよドキュメントビューワーからアプリケーションを提供するプラットフォームに生まれ変わった［5]。これはWeb標準を採用することで可読性を整え終えたホームページが新しいステップの試行を始めたと捉えられる。

その中でひとりスタイルシートのコーディングにおいては課題が残っているようだ。画像表示の高速化を模索するCSSストライプでは若干，Base64エンコードの技法では可読性を大きく犠牲にしている。CSSではコードの見易さに対して無頓着なのであろうか。

スタイルシートの可読性については実際，ガイドラインの作成が励行されたり，CSS拡張言語のSASSやLESSが開発されたりしている[19]。可読性を増すためにはネストやコメントも有効だが，時代を画すほどの改善は規則や構造の導入が必要であろう。逆にスタイルシートについてはまだまだ盛んに技法が開発されている過程であり混沌解消のための規則導入は時期尚早なのであろう。

注

[1]更に３つ目を挙げるとスクリプトによる動的な効果を作成する過程がある。

[2]現在ではcanvas要素を含めて３通り数えられるかもしれない。

[3]画像表示などに関してHTML5からキャプションとのグループ化が行える。figure要素内にfigcaption要素とimg要素を含ませて指定する。figure要素は図表，画像，プログラムなど文書から参照されていて独立したページにできるようなフロー・コンテンツをあらわすのに適している。

[4]背景画像の拡大縮小はCSS3からbackground-size属性で行えるようになった。

[5]ロードマップ（Plan 2014：http://dev.w3.org/html5/decision-policy/html5-2014-plan.html）によると,策定の完了した仕様は2014年に勧告になる。

[6]このテクニックは，米GoogleやYouTubeなどが採用していることでも有名だ［1]。

[7]DreamWeaverやHomePageBuilder等のいわゆる開発ツールにもあるし，WebサービスとしてもCSS Sprite Generator（http://ja.spritegen.website-performance.org/）　や CSS Sprites generator（http://csssprites.com/）などある。

[8]本稿で使用したWSHでのコードは以下のように画像ファイルのドラッグドロップでファイル名＋".txt"（～.png.txtといった名前になる）アスキーファイルを作る。

```
Const TransformTable =
    "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrs
    tuvwxyz0123456789+/"
Set objStream =
    WScript.CreateObject("ADODB.Stream")
Set objFSO =
    WScript.CreateObject("Scripting.FileSystemObject")
filename = WScript.Arguments(0)&".txt"
Set objFile = objFSO.OpenTextFile(filename, 8,
```

```
    True, vbUseDefault)
objStream.Open
objStream.Type = 1
objStream.LoadFromFile WScript.Arguments(0)
objStream.position = 0
vReadData = objStream.Read
objStream.Close

d = 0
For i = 1 To LenB(vReadData)
    b = AscB(MidB(vReadData, i, 1))
    Select Case (i-1) Mod 3
    Case 0
        c = b ¥ 4
        d = b Mod 4
    Case 1
        c = (b ¥ 16) + d*16
        d = b Mod 16
    Case 2
        c = d * 4 + (b ¥ 64)
        d = b Mod 64
        objFile.Write Mid(TransformTable, c + 1, 1)
        c = d
        d = 0
    End Select
    objFile.Write Mid(TransformTable, c + 1, 1)
    If  i Mod 57 = 0 Then
        objFile.Write vbCrLf
    End If
Next

Select Case (i-1) Mod 3
Case 0
Case 1
    objFile.Write Mid(TransformTable, d*16 + 1, 1)
    objFile.Write "=="
Case 2
    objFile.Write Mid(TransformTable, d*4 + 1, 1)
    objFile.Write "="
End Select
objFile.Close
```

[9]巨大で意味の取れない文字列をコードの最後に置くためにまとめて別に置いた。もしhtmlファイル内に記述するとしたらスタイルシートを<head>の中以外に文書末にも分けて記載する。

[10]「絵文字」のこと。一般的には単純な形で構成された視覚記号である。

[11]CSS3のMedia Queriesなどを使ってスマートフォンやタブレット，PCなどあらゆるデバイス端末表示に対応するデザイン手法をレスポンシブWebデザインと言う。

[12]（２本指で行う）画面の拡大・縮小操作のことをいう。

[13]ウェブフォントのフォーマットは WOFF・TTF・EOT・SVG の４種類で，現在はサポートしていないブラウザーも残っているが，今後はWOFF フォーマットが標準になる。

フォーマット	フォーマット指定
WOFF(Web Open Font Format)	"woff"
TrueType	"truetype"
OpenType	"opentype"
Embedded OpenType	"embedded-opentype"
SVGフォント	"svg"

[14]使われないフォントデータをファイルから省いて軽量化することをいう。

[15]フォントのサブセット化を行うWebサービスとしてhttp://fontello.com/を例示する。

[16]合字Ligatureを活用してカスタムフォントを表示させる。シンボルフォント（http://alistapar t.com/article/the-era-of- symbol-fonts）が注目されている。FireFoxでは@fontの指定だけで表示されたが，Chrome，Safariでは表示させるのにtext-rendering属性にoptimizeLegibilityも設定する。

[17]HTML5のWebストレージAPIでは，ローカルに保存できるデータ量の上限が特に設定されていない。大量のデータを保存させ，ハードディスクを埋めることもできる（http://feross.org/fill-disk/）。

[18]サーバのリソースを更新したい場合は，マニフェストファイルに何らかの変更を必ず行う必要があり，そのためコメント部分のバージョンやタイムスタンプをアップデートするたびに書き換える。

[19]CSSの文法はほぼそのままにCSSを拡張するメタ言語である。表現の分離という本来の目的を果たしつつ，変数の利用やルールセットのネスト(入れ子構造)，などが行える。RubyやJavaScriptで変換して使用する。

参　考　文　献

1．こもりまさあきほか.「高速化」.『HTML+CSSコーディング　ベストプラクティス』.（2010/8/25）．162-163頁.
2．大川善邦.「キャラクターのブリッティング」.『DirectXによるゲームプログラミング』.（1997/03）.166-167頁.
3．福崎俊博ほか.「保守性のために保守をする」.『C言語を256倍使うための本』.（1990/09）．194頁.
4．森部陽一郎.「Web標準に関する一考察」.『宮崎公立大学人文学部紀要』13(1).（2006-03-20）．293-302頁.
5．小松健作. Webアプリケーションの進展. 徹底解説HTML5 APIガイドブック コミュニケーション系API編. 出版地不明 : 秀和システム, 2010/12, ページ: 14.
6．小松健作, ほか.『JavaScriptコーディング ベストプラクティス』.（2011）.
7．小松健作.『徹底解説HTML5 APIガイドブック　オフライン系API編』.（2011）.