GENECODE

GC スクリプト利用ガイド

第 1 版

最終更新日 2016/03/31

目次

改訂履歴

版数	発行日	改訂内容
第 1 版	2016 年 3 月 31 日	・初版発行

第1章　　GC スクリプトとは？

GC スクリプトは **GENECODE 上(サーバ側)**で動作するスクリプト言語で、PHP や JSP のように HTML ソースコード中に、GC スクリプトタグを使って埋め込むことができます。

GC スクリプト内では、PC コンテンツの DOM を自由に編集して、テンプレート変換後の HTML として出力することができます。表の組み換え、画像の Alt 属性変更やリンク先の変更、要素の順序変更、削除や複製などを柔軟に記述することが可能です。

テンプレートファイルで GC スクリプトを使う場合、<gc-script></gc-script>タグの中に書きます。

GENECODE IDE の使い方については、ユーザーマニュアルをご参照ください。

- GeneCode Developer Connection IDE　ユーザーマニュアル
- http://developer.genecode.jp/manuals/ide/latest/gc_ide_user_manual.pdf

1. GENECODE IDE の Firefox で変換したいページを開く
2. GENECODE IDE を起動する
3. サーバー変換ステージへ切り替える
4. 新規テンプレートファイルを開く
5. エディターにコードを記述したりパーツを配置する
6. 保存して、右ペインのコンバージョンビューで確認する

上記マニュアルの「1.4.2. サーバー変換ステージ」以降に詳細な手順が説明されています。

この GC スクリプトリファレンスでは、上の図に従い、

1. 変換元 PC ページ
2. テンプレートファイル
3. 変換後スマートフォンページ

のステップで解説をしています。

それでは早速 テンプレートファイルを見てみましょう。

テンプレートファイルの例

```
<html>
      <head><title>DEMO</title></head>
       <body>
```

```
        <gc-script>
            // ここに GC スクリプトを書きます
            // スラッシュふたつの行はコメントとして扱われ、実行されません
            /*
              複数行のコメントはこのようにスラッシュとアスタリスクで
*/
            // 例）変換元のページから id が foo の要素を探し、変換結果に出力する例です

            gcutil.toHtml($('#foo'));
        </gc-script>
    </body>
</html>
```

第2章　　JavaScript と GC スクリプトの違いは？

JavaScript は、**ブラウザ上（クライアント側）**で動作するプログラミング言語のひとつです。
GC スクリプトは、GENECODE サーバ上で動作する V8 上で動作する JavaScript のことを指します。一般的な JavaScript 構文に加え、jQuery や GENECODE 独自のライブラリと合わせて利用します。

2-1　　jQuery とは？

jQuery は JavaScript をより簡単に使えるようにするためのライブラリです。GC スクリプトタグ内でも jQuery を記述して、簡単に変換元の DOM 構造に手を加えることができます。

2-2　　jQuery オブジェクトを作ってみよう

jQuery ではオブジェクトという HTML の要素を単位に処理を記述します。
ファーストステップとして、jQuery オブジェクトを生成してみましょう。

テンプレートファイル

```
<html>
      <head><title>DEMO</title></head>
       <body>
       <gc-script>
             $subtext = $( '<p>こんにちは！</p>' );
       </gc-script>
       </body>
</html>
```

これで ***$subtext*** と定義された jQuery オブジェクトに *** こんにちは！ *** という HTML 要素が格納されました。ただしこのままではオブジェクトが作られただけで、変換後のページには表示されません。次節で追加方法を説明します。

2-3　　jQuery オブジェクトを既存の要素に追加してみよう

前述の方法で作ったオブジェクトは、そのままではページに反映されません。

そこで既存の要素に対して追加する操作をしてみましょう。

変換元 PC ページ

```
<html>
      <head><title>DEMO</title></head>
      <body>
          <div id = “box1” >
              <h1>見出しです</h1>
          </div>
      </body>
  </html>
```

この DOM の見出し *div#box1* の次に前節で作成した *$subtext* オブジェクトを追加する GC スクリプト を書いてみましょう。

テンプレートファイル

```
<html>
      <head><title>DEMO</title></head>
       <body>
       <gc-script>
              $subtext = $( ‘<p>こんにちは！</p>’ );
                  $( ‘div#box1’ ).append($subtext);
       </gc-script>
       </body>
</html>
```

変換後スマートフォンページ

```
<html>
      <head><title>DEMO</title></head>
      <body>
          <div id = “box1” >
              <h1>見出しです</h1>
              <p>こんにちは！</p>
           </div>
      </body>
</html>
```

2-4　GC スクリプトで要素を出力してみよう

以下のページに対し、gcutil.toHtml($('#foo')) を実行してみましょう。

変換元 PC ページ

```
<html>
    <head><title>DEMO</title></head>
    <body>
        <p id = “foo” > この要素が出力されます </p>
        <p> この要素は出力されません </p>
    </body>
</html>
```

テンプレートファイル

```
<html>
    <head><title>DEMO</title></head>
     <body>
     <gc-script>
            // 例）変換元のページから id が foo の要素を探し、変換結果に出力する例です
            gcutil.toHtml($('#foo'));
        </gc-script>
   </body>
</html>
```

変換後スマートフォンページ

```
<html>
    <head><title>DEMO</title></head>
     <body>
     <p id = “foo” > この要素が出力されます </p>
     </body>
</html>
```

GENECODE で変換した結果は上記の通りとなります。

2-5　タイトルタグを連動させてみよう

変換元 PC ページ

```
<html>
    <head><title>サイトのタイトル - SYMMETRIC</title></head>
    <body>
        <p id = “foo”> 本文です。 </p>
    </body>
</html>
```

テンプレートファイル

```
<html>
    <head>
     <gc-script>
        //ページのタイトルを連動させます
            gcutil.toHtml($('title'));
     </gc-script>
  </head>
  <body>
     <gc-script>
            gcutil.toHtml($('#foo'));
     </gc-script>
  </body>
</html>
```

変換後スマートフォンページ

```
<html>
    <head>
     <title>サイトのタイトル - SYMMETRIC</title>
     </head>
     <body>
     <p id = “foo”> 本文です。 </p>
     </body>
</html>
```

GENECODE で変換した結果は上記の通りとなります。

第3章　　GC スクリプトタグ内で使える関数たち

GC スクリプトは基本的に通常の JavaScript と同じですが、入出力やロギングなどについて、独自の API を使用することができます。

その他の関数については、「JavaScript API リファレンス」をご参照ください。

JavaScript API リファレンス – GeneCode Developer Connection

http://developer.genecode.jp/manuals/gcruntime/latest/gc2_javascript_api.pdf

第4章　　代表的なjQuery セレクタ

4-1　　要素セレクタ

p や div など要素名を直接書くと、その要素すべてが対象となります。
また、セレクタを h1, h2, h3 のようにカンマ区切りで複数指定することができます。

```
$( “h1, h2, h3” ).remove();
```

4-2　　クラスセレクタ

クラス属性を持っている要素は、．(ドット)を付けて選択します。

```
$( “div.wrapper” ).remove();
```

4-3　　ID セレクタ

ID 属性を持っている要素は、# (シャープ)を付けて選択します。

```
$( “div#box1” ).remove();
```

4-4　　子孫セレクタ

要素の包含関係が親子に該当するときに使うセレクタです。
半角スペース記号を用いると孫以降の要素にもマッチし、＞ 記号を用いた場合は直接の子要素のみがマッチします。

$(‘div.mgn10 p ＞ span’) がマッチする要素の例です。

```
<div class =” mgn10” >
      <p> p タグの中身です。<span>この要素にマッチします</span></p>
      <span>この span タグにはマッチしません</span>
</div>
```

4-5　隣接セレクタ

兄弟セレクタとも呼ばれます。
ある要素と同じ階層にある、次の要素を指すときに使います。
$('h1 + span') というセレクタがマッチする要素の例です。

```
<h1>タイトルです</h1>
  <span>この要素にマッチします</span>
```

4-6　属性セレクタ

属性や属性値の有無を使用して要素を選択します。
input タグの形式によって処理対象を限定したい場合などによく用います。

$('input[type="text"]') というセレクタがマッチする要素の例です。
黄色のハイライトの行が該当します。

イコール記号以外に、前方一致や後方一致、部分一致などが使えます。

```
<form action="form.php" method="post">
    <input type=" text" />
    <input type=" submit"  value=" 送信する"  />
</form>
```

jQuery では基本的に CSS セレクタと同様のものが使用できます。
詳細は公式のリファレンスなどを参考にしてください。

セレクタ －jQuery 日本語リファレンス http://semooh.jp/jquery/api/selectors/

第5章　　要素の選択をしよう

5-1　　子孫要素を選択する関数

jQuery で取得したオブジェクトから、さらに子要素などを取得することができます。
children, find は代表的な関数です。

変換元 PC ページ

```
<html>
    <head><title>DEMO</title></head>
    <body>
        <div id = “box1” >
            <p>こんにちは。<br />
              <span class=” text” >シンメトリックです！</span>
            </p>
        </div>
    </body>
</html>
```

テンプレートファイル

```
<html>
    <head><title>DEMO</title></head>
     <body>
     <gc-script>
            var $target = $( ‘div#box1’ ).children();
            gcutil.toHtml($target);
     </gc-script>
    </body>
</html>
```

変換後スマートフォンページ

```
<html>
    <head><title>DEMO</title></head>
     <body>
```

```
                <p>こんにちは。<br /><span class=" text" >シンメトリックです！</span></p>
        </body>
</html>
```

となります。children 関数の場合は子孫要素すべてが対象です。
それに対し、find 関数は引数としてセレクタを与えることで、取得する対象を絞り込むことができます。
操作対象の要素を探す際に便利です。
前例と同じ変換元 PC サイトに以下テンプレートファイルを適用してみましょう。

テンプレートファイル

```
<html>
      <head><title>DEMO</title></head>
       <body>
       <gc-script>
              var $target = $( 'div#box1' ).find( 'span.text' );
                  gcutil.toHtml($target);
       </gc-script>
      </body>
</html>
```

変換後スマートフォンページ

```
<html>
      <head><title>DEMO</title></head>
   <body>
     <span class=" text" >シンメトリックです！</span>
   </body>
</html>
```

span.text 要素のみが出力されています。

find での検索では、子孫要素すべてが対象になります。

children – jQuery 日本語リファレンス
http://semooh.jp/jquery/api/traversing/children/%5Bexpr%5D/

find – jQuery 日本語リファレンス

http://semooh.jp/jquery/api/traversing/find/expr/

5-2　　兄弟要素を選択する関数

変換元 PC ページ

```
<html>
      <head><title>DEMO</title></head>
      <body>
          <div id = “box1” >
              <p>おはようございます。シンメトリックです！</p>
          </div>
          <div id = “box2” >
                  <p>こんにちは。シンメトリックです！</p>
          </div>
          <div id = “box3” >
              <p>こんばんは。シンメトリックです！</p>
          </div>
      </body>
</html>
```

div#box1 を起点として、ひとつ次の要素 *div#box2* を出力してみましょう。

テンプレートファイル

```
<html>
      <head><title>DEMO</title></head>
       <body>
       <gc-script>
              var $target = $( ‘div#box1’ ).next();
                  gcutil.toHtml($target);
        </gc-script>
       </body>
</html>
```

変換後スマートフォンページ

```
<html>
      <head>
```

```
        <title>サイトのタイトル - SYMMETRIC</title>
        </head>
        <body>
                <div id = “box2” >
                        <p>こんにちは。シンメトリックです！</p>
                </div>
        </body>
</html>
```

では次に、 *div#box3* を起点として、ひとつ前の要素 *div#box2* を出力してみましょう。

テンプレートファイル

```
<html>
      <head><title>DEMO</title></head>
       <body>
       <gc-script>
              var $target = $( ‘div#box3’ ).prev();
                  gcutil.toHtml($target);
        </gc-script>
       </body>
</html>
```

変換後スマートフォンページ

```
<html>
      <head><title>DEMO</title></head>
       <body>
                <div id = “box2” >
                        <p>こんにちは。シンメトリックです！</p>
                </div>
       </body>
</html>
```

詳細は jQuery の API リファレンスをご参照ください。

next – jQuery 日本語リファレンス

http://semooh.jp/jquery/api/traversing/next/%5Bexpr%5D/

prev – jQuery 日本語リファレンス

http://semooh.jp/jquery/api/traversing/prev/%5Bexpr%5D/

5-3　　親要素を選択する関数

ある要素を包含する親要素を選択したい場合、closest, parent, parents などの関数を用います。

変換元 PC ページ

```
<html>
      <head><title>DEMO</title></head>
      <body>
          <div class = “wrapper” >
              <div id = “box1” >
                  <p  id = “message1” >おはようございます。シンメトリックです！</p>
              </div>
              <div id = “box2” >
                  <p id = “message2” >こんにちは。シンメトリックです！</p>
              </div>
              <div id = “box3” >
                  <p id = “message3” >こんばんは。シンメトリックです！</p>
              </div>
          </div>
       </body>
</html>
```

まずは変換元 PC ページに対して *parent* 関数を実行してみましょう。

テンプレートファイル

```
<html>
      <head><title>DEMO</title></head>
       <body>
       <gc-script>
              $target = $( ‘#message1’ ).parent();
                  gcutil.toHtml($target);
       </gc-script>
```

```
  </body>
</html>
```

このコードでは直近の親要素である ***div#box1*** が選択・出力されます。

変換後スマートフォンページ

```
<html>
      <head><title>DEMO</title></head>
       <body>
           <div id = “box1” >
                  <p  id = “message1” >おはようございます。シンメトリックです！</p>
           </div>
       </body>
</html>
```

parent 関数では直接の親要素しか選択できませんが、parents 関数では親要素をさらにさかのぼって、親の親…のように DOM を選択することができます。

parent – jQuery 日本語リファレンス
http://semooh.jp/jquery/api/traversing/parent/%5Bexpr%5D/

parents – jQuery 日本語リファレンス
http://semooh.jp/jquery/api/traversing/parents/%5Bexpr%5D/
前例と同じ変換元 PC ページに対して *closest* 関数を実行してみましょう。

テンプレートファイル

```
<html>
      <head><title>DEMO</title></head>
   <body>
       <gc-script>
              $target = $( ‘#message1’ ).closest( ‘#wrapper’ );
                  gcutil.toHtml($target);
       </gc-script>
   </body>
</html>
```

このコードでは closest 関数にセレクタで wrapper という ID を指定しています。指定されたセレクタにマッチするまで親をさかのぼり、最初にマッチした要素が選択されます。div#wrapper は body 要素直下にありますので、上記テンプレートでは PC ページと同じタグ構造のまま出力されることになります。

変換後スマートフォンページ

```
<html>
      <head><title>DEMO</title></head>
       <body>
       <div class = “wrapper” >
              <div id = “box1” >
                  <p  id = “message1” >おはようございます。シンメトリックです！</p>
              </div>
              <div id = “box2” >
                  <p id = “message2” >こんにちは。シンメトリックです！</p>
              </div>
              <div id = “box3” >
                  <p id = “message3” >こんばんは。シンメトリックです！</p>
              </div>
        </div>
      </body>
</html>
```

用途に応じて使い分けましょう。

closest – jQuery 日本語リファレンス
http://semooh.jp/jquery/api/traversing/closest/%5Bexpr%5D/

第6章　　要素に属性を付加してみよう

6-1　　クラス属性を付け外ししてみよう

以下のソースコードにクラス属性を付加してみましょう。

変換元 PC ページ

```
<html>
      <head><title>DEMO</title></head>
      <body>
          <div id = “box1” >
              <p>こんにちは。<br />
                 <span class=” text” >シンメトリックです！</span>
                  </p>
          </div>
      </body>
</html>
```

テンプレートファイル

```
<html>
      <head><title>DEMO</title></head>
       <body>
       <gc-script>
              $target = $( ‘#box1’ ).addClass( ‘mgn10’ );
                  gcutil.toHtml($target);
       </gc-script>
       </body>
</html>
```

変換後スマートフォンページ

```
<html>
      <head><title>DEMO</title></head>
       <body>
```

```
        <div id = “box1” class=” mgn10” >
                <p>こんにちは。<br />
                 <span class=” text” >シンメトリックです！</span>
                    </p>
        </div>
        </body>
</html>
```

続いて、クラス属性を除去してみましょう。

テンプレートファイル

```
<html>
      <head><title>DEMO</title></head>
       <body>
       <gc-script>
              $target = $( ‘#box1 > p > span’ ).removeClass( ‘text’ );
                  gcutil.toHtml($target);
       </gc-script>
       </body>
</html>
```

変換後スマートフォンページ

```
<html>
      <head><title>DEMO</title></head>
   <body>
       <div id = “box1” class=” mgn10” >
              <p>こんにちは。<br />
               <span>シンメトリックです！</span>
                  </p>
       </div>
   </body>
</html>
```

span タグに設定されていた text クラス属性が取り除かれています。

6-2　属性の値を取得、変更してみよう

attr 関数に属性名を渡すと、要素に設定されたその属性値を取得することができます。

変換元 PC ページ

```
<html>
      <head><title>DEMO</title></head>
      <body>
          <div id = “box1” >
              <input type="number" name="phone" id=” tel”  />
          </div>
       </body>
</html>
```

に対して下記テンプレートファイルを適用してみましょう。

テンプレートファイル

```
<html>
      <head><title>DEMO</title></head>
       <body>
       <gc-script>
              var value = $( ‘#tel’ ).attr( ‘type’ );
       </gc-script>
   </body>
</html>
```

を実行すると、 ***value*** という変数には ***number*** という値が代入されます。また、attr 関数に属性名と値を渡すと、その属性に値をセットすることができます。前例と同じ変換元 PC ページに次のテンプレートを適用してみましょう。

テンプレートファイル

```
<html>
      <head><title>DEMO</title></head>
       <body>
       <gc-script>
              $( ‘#tel’ ).attr( ‘name’ ,  ‘mobile’ );
       </gc-script>
       </body>
```

```
</html>
```

を実行すると、 以下のように属性がセットされます。

変換後スマートフォンページ

```
<html>
      <head><title>DEMO</title></head>
       <body>
       <input type="number" name="mobile" id=" tel"  />
       </body>
</html>
```

attr の第 1 引数に属性名を、第 2 引数に値を渡すことで、さまざまな属性を設定することができます。

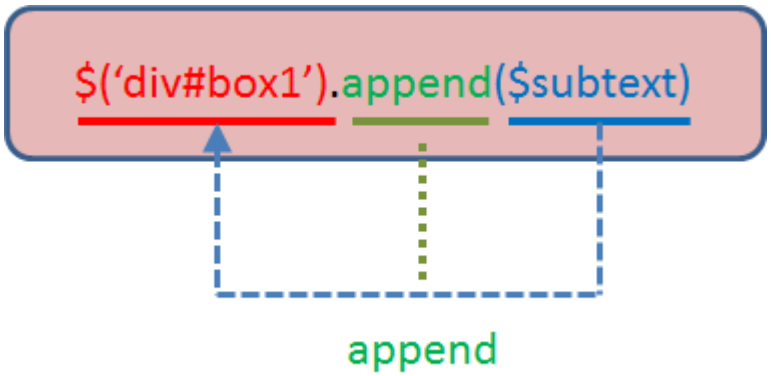
第7章　　要素を操作してみよう

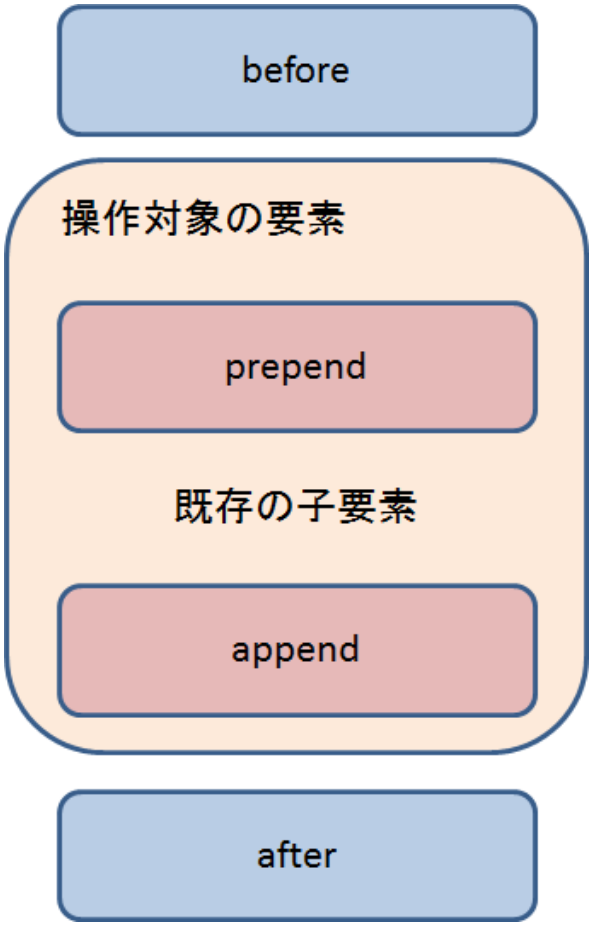
7-1　　要素にコンテンツを追加してみよう

既存 DOM に要素を追加する関数はいくつもありますが、代表的なものを挙げて紹介します。

基本となるのは append, prepend, after, before の 4 種類です。

box1 という ID を持つ div 要素に $subtext という jQuery オブジェクトに含まれる DOM 要素を追加する例です。





7-2　　要素を削除してみよう

DOM ツリーから要素を削除するときは remove 関数を使います。たとえば p タグの直下にある改行タグを削除する GC スクリプト は下記のようになります。

変換元 PC ページ

```
<html>
    <head><title>DEMO</title></head>
    <body>
        <div id = “box1” >
            <h1>見出しです</h1>
            <p>こんにちは！<br />シンメトリックです！<br />GC スクリプトです。</p>
        </div>
    </body>
</html>
```

テンプレートファイル

```
<html>
      <head><title>DEMO</title></head>
       <body>
       <gc-script>
             $( 'div#box1 > p > br' ).remove();
       </gc-script>
      </body>
</html>
```

ブラウザでの表示結果:

変換元 PC ページ

見出しです
こんにちは！
シンメトリックです！
GC スクリプトです。

変換後スマートフォンページ

見出しです
こんにちは！シンメトリックです！GC スクリプトです。

7-3　要素を複製してみよう

要素を DOM ごと複製したい場合は clone 関数を使います。たとえば p タグ以下を丸ごと複製し、div#box1 の末尾に追加する GC スクリプト は下記のようになります。

変換元 PC ページ

```
<html>
      <head><title>DEMO</title></head>
      <body>
          <div id = "box1" >
              <h1>見出しです</h1>
```

```
            <p>こんにちは！<br />シンメトリックです！<br />GC スクリプトです。</p>
        </div>
    </body>
</html>
```

テンプレートファイル

```
<html>
    <head><title>DEMO</title></head>
    <body>
        <gc-script>
            $( 'div#box1' ).append($( 'div#box1 > p' ).clone());
         gcutil.toHtml($('div#box1'));
        </gc-script>
    </body>
</html>
```

変換後スマートフォンページ

```
<html>
    <head><title>DEMO</title></head>
    <body>
        <div id = "box1" >
            <h1>見出しです</h1>
            <p>こんにちは！<br />シンメトリックです！<br />GC スクリプトです。</p>
            <p>こんにちは！<br />シンメトリックです！<br />GC スクリプトです。</p>
         </div>
    </body>
</html>
```

ID 属性を持つ要素を複製する場合は、複製後に重複しない ID に変更しないと意図せぬ動作を引き起こすことがあります。ご注意ください。

clone – jQuery 日本語リファレンス

http://semooh.jp/jquery/api/manipulation/clone/true/

7-4　要素を別のタグで囲ってみよう

要素を新しくタグで囲みたいときは wrap 関数を用います。
以下の要素を div タグで囲ってみましょう。

変換元 PC ページ

```
<html>
      <head><title>DEMO</title></head>
      <body>
          <span class = " text" >この要素がラップされます<span>
       </body>
</html>
```

テンプレートファイル

```
<html>
      <head>
       <gc-script>
           gcutil.toHtml($('title'));
       </gc-script>
       </head>
   <body>
       <gc-script>
             $( 'span.text' ).wrap( '<div class = "wrapper" />' );
       </gc-script>
   </body>
</html>
```

変換後スマートフォンページ

```
<html>
      <head><title>DEMO</title></head>
      <body>
          <div class = "wrapper" >
            <span class = " text" >この要素がラップされます<span>
          </div>
       </body>
</html>
```

となります。

wrap 関数にはいくつか種類があり、wrap, wrapAll, wrapInner など用途に応じて使い分けます。

詳しくは jQuery の API リファレンスをご参照ください。

wrap – jQuery 日本語リファレンス
http://semooh.jp/jquery/api/manipulation/wrap/html/

wrapInner – jQuery 日本語リファレンス
http://semooh.jp/jquery/api/manipulation/wrapInner/html/

wrapAll – jQuery 日本語リファレンス
http://semooh.jp/jquery/api/manipulation/wrapAll/html/

7-5　要素を囲んでいるタグを除去してみよう

要素をラップしている div タグが不要な場合を想定してみましょう。

変換元 PC ページ

```
<html>
      <head><title>DEMO</title></head>
      <body>
          <div class = “wrapper” >
           <span class = ” text” >この要素が unwrap されます<span>
         </div>
     </body>
</html>
```

に対し下記テンプレートファイルを適用します。

テンプレートファイル

```
<html>
      <head><title>DEMO</title></head>
       <body>
       <gc-script>
              $( ‘span.text’ ).unwrap();
```

```
        </gc-script>
      </body>
</html>
```

変換後スマートフォンページ

```
<html>
      <head><title>DEMO</title></head>
       <body>
       <span class = " text" >この要素が unwrap されます<span>
       </body>
</html>
```

span タグを囲っていた div タグが除去されました。

.unwrap | jQuery 1.9 日本語リファレンス | js STUDIO
http://js.studio-kingdom.com/jquery/manipulation/unwrap

7-6 DOM 要素の HTML コードを取得してみよう

html 関数は、任意の要素の HTML コードそのものを取得したり、変更したりすることができます。

変換元 PC ページ

```
<html>
      <head><title>DEMO</title></head>
      <body>
          <div id = "box1" >
              <h1>見出しです</h1>
              <p>こんにちは！</p>
           </div>
      </body>
</html>
```

テンプレートファイル

```
<html>
      <head><title>DEMO</title></head>
       <body>
```

```
        <gc-script>
              target = $( 'div#box1' ).html();
        </gc-script>
    </body>
</html>
```

を適用してみましょう。

target 変数には <h1>見出しです</h1> こんにちは！ という DOM 要素が格納されます。

7-7　DOM 要素の HTML を変更してみよう

続いて、取得した DOM をページに反映してみましょう。html 関数に引数を渡すことで、直接要素内の HTML コードを書き換えることができます。前述の変換元 PC ページに以下のテンプレートを適用してみます。

テンプレートファイル

```
<html>
      <head><title>DEMO</title></head>
       <body>
       <gc-script>
           $( 'div#box1' ).html( '<p>こんにちは。シンメトリックです！</p>' );
       </gc-script>
      </body>
</html>
```

変換後スマートフォンページ

```
<html>
      <head><title>DEMO</title></head>
      <body>
          <div id = "box1" >
              <p>こんにちは。シンメトリックです！</p>
           </div>
      </body>
</html>
```

h1 タグがなくなり、文言が変更されました。

7-8　任意の箇所に改行タグを挿入してみよう

では同じ変換元 PC ページ の DOM を操作し、「こんにちは。」のあとに改行タグを挿入してみましょう。

テンプレートファイル

```
<html>
      <head><title>DEMO</title></head>
       <body>
       <gc-script>
              $elem = $( '#box1 > p' );
              html = $elem.html() ||  '' ;
              $elem.html(html.replace(/(こんにちは。)/g,  '$1<br />' ));
       </gc-script>
       </body>
</html>
```

変換後スマートフォンページ

```
<html>
      <head><title>DEMO</title></head>
      <body>
          <div id = "box1" >
              <p>こんにちは。<br />シンメトリックです！</p>
           </div>
      </body>
</html>
```

前述の例では replace という関数を使って、html 関数で取得した文字列の中で「***こんにちは。***」を検索し、その後ろに ***
*** タグを挿入しています。

replace 関数は空のオブジェクトに対して実行するとエラーが発生します。
万が一 $('#box1 > p') で要素が取得できなかった場合、次に実行される html 変数への代入そのものが空になってしまうのを防ぐため、html = $elem.html() || ''; で空の文字列と or を取り、文字列のオブジェクトそのものが存在しないというエラーが発生しないようにしています。

/(こんにちは。)/g は正規表現です。$1 には正規表現にマッチした文字列が代入されます。

7-9　繰り返しの要素を出力してみよう

複数個の要素が繰り返し表示されているページを変換する際に、ひとつずつ要素を出力する GC スクリプトを書いていくのは大変です。また数が一定でない場合に対応できません。そこで GENECODE では、セレクタにマッチした要素の個数分ループして出力を行うためのタグがあります。それが ***gc-each*** タグです。

gc-each タグでは、終了タグまでに記述されているテンプレート・スクリプト・パーツをセレクタ属性で指定したセレクタで抽出された要素の数だけ繰り返します。

では実際に、下記ソースコードの div タグ内の h1, p タグのみを、要素の順番そのままに出力するテンプレートを作成してみましょう。

変換元 PC ページ

```
<html>
      <head><title>DEMO</title></head>
       <body>
       <div id = “box1” >
              <h1>タイトル 1</h1>
              <p>おはようございます。シンメトリックです！</p>
       </div>
          <div id = “box2” >
              <h1>タイトル 2</h1>
              <p>こんにちは。シンメトリックです！</p>
          </div>
          <div id = “box3” >
              <h1>タイトル 3</h1>
              <p>こんばんは。シンメトリックです！</p>
          </div>
   </body>
</html>
```

テンプレートファイル

```
<html>
      <head><title>DEMO</title></head>
       <body>
```

```
        <gc-each selector="div">
            <gc-script>
                    gcutil.toHtml($('h1').eq(GCIDX));
                    gcutil.toHtml($('p').eq(GCIDX));
            </gc-script>
        </gc-each>
        </body>
</html>
```

前述テンプレートファイルでは、*gc-each* タグ内で GC スクリプトを使用しています。
GCIDX 変数を使うことで、何番目の要素、といった指定の仕方ができます。

例では div タグの個数だけループしながら、h1 タグと p タグの *GCIDX* 番目を出力しています。

変換後スマートフォンページ

```
<html>
      <head><title>DEMO</title></head>
   <body>
       <h1>タイトル 1</h1>
          <p>おはようございます。シンメトリックです！</p>
          <h1>タイトル 2</h1>
          <p>こんにちは。シンメトリックです！</p>
          <h1>タイトル 3</h1>
          <p>こんばんは。シンメトリックです！</p>
    </body>
</html>
```

第8章　　参考文献、サイト

- jQuery 1.9 日本語リファレンス | js STUDIO http://js.studio-kingdom.com/jquery
- GeneCode Developer Connection テンプレート構文リファレンス

  http://developer.genecode.jp/manuals/gcruntime/latest/gc2_template_syntax.pdf

- GeneCode Developer Connection IDE ユーザーマニュアル

  http://developer.genecode.jp/manuals/ide/latest/gc_ide_user_manual.pdf

以下 URL にてサポートをしております。

お問い合わせ – GeneCode Developer Connection https://genecode.jp/developer/contact

または genecode_support@symmetric.co.jp までお気軽にお問い合わせください。