Word 文書の HTML による再利用法について

中田　聡美

目　次

――Word 文書の HTML による再利用法について――

1　はじめに

(1) 制作テーマ

福田ゼミでは「人の役に立つものを作る」を卒業論文のテーマとしている。そこで私は「Word 文書の HTML による再利用法について」卒業制作、また論文を作成した。これは Word 文書をインターネット上にあげる際、テキストを書き出し、１からタグを打たないといけない手間などをはぶこうというものだ。

(2) 制作目的

この論文中で書いている Word(1)は、Microsoft 社のワープロソフトである。これは Microsoft Office の一部として提供されているものだ。

Word を HTML で保存すると見た目には変わりがない。しかし、ソースを開くと Word 特有のタグが多く、短い文書内容でも長いソースになっている。それゆえに見づらく、容量も大きくなってしまっている。また Word 特有のタグがあることで Internet Explorer 以外のブラウザで見た場合にデザインがくずれることがある。

そこで私は、Word を HTML 形式で保存したときのソースにある、余計なタグを取り除き、足りないタグを加えるというソフトを作ることにした。デザインをなるべくくずさずに必要最低限のタグだけを残したソースを表示するようなソフトである。簡単にいえば、ソースを見やすくし、サイズを小さくすることが目的である。また、ソース内にある内部 CSS タグを消し、HTML のみのタグにすることでデザインを外部 CSS で指定できるようにする。簡単な操作で使うことを可能にし、ある程度の HTML や CSS の知識があれば使用できるようにすることを目指す。

また、HTML のみにタグを絞ったことで、後から CSS でデザインすることが可能になっている。デザインを変えるつもりなら、HTML 化した後の表示も考えた構成をしておいた方がよい。「編」「章」「節」「項」といったスタイルをはっきりし、全体を通じて見やすいドキュメントを作っておくことでソースもわかりやすく、修正しやすいものになる。なお、CSS は外部 CSS で指定することを推奨する。

制作中の勉強として利用したものは、「とほほの WWW 入門」(2)「正規表現メモ」(3)などのサイトが主である。HTML タグのチェックには「Another HTML-lint gateway」(4)サイトを使用した。制作したソフトは、ソフトを通して出来たソースがこのサイトで 100 点を出すことを目標にした。

(3) 内部 CSS と外部 CSS

CSS は HTML の見映えの部分のタグを統括しているものである。そして、統括したタグ類を、html 内の head 部分に置くことを内部 CSS といい、外部にリンクさせて置くことを外部 CSS という。CSS で見栄えを整える利点は、body 部分のタグを軽量化し、HTML 構文をシンプル化できるところである。

現在、HTML 文書の中に CSS が含まれているのはあまり推奨されていない。というのも、外部 CSS にすると複数のページを１枚のスタイルシートで管理統括できるという利点があるからである。ページが多い場合、見映えの部分に修正が生じた時、内部 html に CSS を埋め込んでいると全てのページをなおしていかなくてはならない。それが外部 CSS なら、１枚のスタイルシートをなおすだけですむからである。

以上のことからソフトを作成するうえで、Word 特有のタグだけでな

く CSS タグをも消し、HTML タグのみ残すということに決めた。

(4) 制作環境

ソフトを使用する前に Microsoft office Word が必要である。さらに Windows のサクラエディタを使用する。制作物は、コマンドプロンプトで実行、テストし、問題点が出れば、またサクラエディタを変更していくという作業を繰り返すことで完成を目指した。

ソフトを Windows で実行する場合、python をインストールしておく必要がある。制作物は python2.5 で作成している。

さらにソフトを実行する前に、Microsoft office Word で Word 文書を HTML として保存しておく。ここで注意する点がある。Word 文書を HTML で保存するのだが、拡張子を.doc から.htm に変更するのではなく、Word 内で [ファイル] → [Web ページとして保存] という手順を踏まないといけない。

次にコマンドプロンプトを開き、コマンドラインでソフト本体のファイル名 henkan.py のあとに半角スペースを空け、実行するファイル名を指定して python を実行する。(Z:\python henkan.py test.htm)

Enter キーを押してエラーがなければ変換できている。エラーが出た場合はファイル名の綴り確認、ファイルの指定場所が正しいか、機種依存文字[(5)]がないか確認してやりなおす。機種依存文字は特定の環境上でしか正しく表示されない文字のことである。同一の文字セット上で、特定の文字コードについて OS やコンピュータごとに違う文字が定義されている場合があり、この文字コードを含む文章を作成すると、環境によって異なる文字が表示され、意図したとおりの表示がされないばかりか、文字化けを起こすこともある。このため、多数の環境が混在してい

るインターネット上では機種依存文字を使うことはできない。丸付き数字や一部の記号、「昭和」「(株)」「メートル」などを一文字にまとめた文字などやローマ数は、機種依存文字としてよく知られている。これらが含まれているとエラーが出るので気をつけて欲しい。機種依存文字は、Shift_jis として python で認識できない。さらに、HTML で使うことは推奨されていないので、使わない方がいいだろう。

エラーがなければ正しく変換されているということなので、さらに違う名前で保存する。(Z:\python henkan.py test.htm > kansei.htm)

実行すると kansei.htm というファイルが自動的に表示される。test.htm は Word 文書を HTML で保存したままの形で、 kansei.htm は henkan.py を介して変換された余計なタグと、CSS タグを除いた HTML タグのみのファイルになる。

余計なタグを検索し、置き換える、また足りないタグを加えるという作業の繰り返しが主であることから、正規表現を使用した。

2　制作物に関する現状

(1) ソフト作成のための現状認識

ソフトを作成する前に現在、Word 文書を HTML に変換する方法にどういうものがあるかを調べる。調べた結果、主に５つの選択肢があることがわかった。PDF ファイルに変換する、専門会社へ依頼する、変換ソフトを使う、Word の機能だけで実行する、Google ドキュメントを使用するというものである。しかしその１つ１つには利点もあれば、問題もある。

■１つ目はよく使われている PDF に変換するという手段である。PDF ファイルは、作成したドキュメントを異なる環境のコンピュータ

で元のレイアウトどおりに表示・印刷できる点が利点である。さらに表示や印刷はアドビシステムズが無料配布している Adobe Reader をインストールすることで Web ブラウザ上で PDF ファイルを閲覧することができる。現在はパソコン購入時にインストール済のものも多く、いかに普及してるかがわかる。しかし、ファイルサイズが大きく重い。表示までに時間がかかるのが難点である。

■２つ目の専門会社に変換を依頼するというものは費用がかかる。料金は会社によって違うが、決して安価ではなく、量の多い文書を依頼する場合は高額になる。さらに、フロッピーや CD-R などの記憶媒体を会社に送らなければならない場合もあり、数日かかることが予想される。

■３つ目の変換ソフトを使うというものは、有名どころとして「軽々 HTML for Word」(6) などのソフトがある。しかしこれもシェアウェアなので費用がかかる。機能はタブや均等割付、テキストボックスなども置き換えてくれるので再現度は高いといわれている。ファイル容量も少なくなるうえに、「文書の幅を固定する」という機能があるので、Word で作成したものに近い HTML 文書をつくることができる。「軽々 HTML for Word」に関するホームページを読んだところ、テスト変換をしてくれるので試すこともできるところは良い。

■４つ目は Word の機能だけで実行するというものである。Word2000 を使っているなら、Microsoft Office HTML Filter というものがある。これは Microsoft 社で無料配布されているもので、インストールすると、ワードのツールバーに「コンパクトな HTML のエクスポート」というボタンが追加される。これをクリックすると、ワード独自のタグのない軽量な HTML 文書に変換することができる。2000 以降のものには元々この機能が追加されている。このことについては、後

述に記す「制作物と類似したソフト」の項目で詳しく説明している。

■５つ目の Google ドキュメント[7]の利用だが、これも後述に詳しく説明している。簡単にいえば、変換したい Word 文書を Gmail に送り、Edit HTML で表示する。すると、Word 文書を Google ドキュメントに変換したうえで HTML 化したソースが表示される。そのソースをコピーし、テキストエディタに貼り付ければいいというものである。もう一つの方法もあるが、後述で説明しているのでここでは省く。

(2) 制作ソフトの必要性

以上の点から、Word 文書を HTML 化するには少しなからず費用がかかることがわかる。かからない方法といえば、テキストに文章を打ち出し、タグを自ら書くことである。また、Google ドキュメントを利用するか、Word 文書を HTML 化しただけのソースから、自分で余計なタグを消すことである。しかしこの場合はお金がかからないかわりに時間がかかる。

一つの文書であったり、簡単なものなら、テキストエディタにコピー貼り付けし、自分で検索置換えするなどして余計なタグを消すのもいい。自らタグを書くのもいいだろう。しかし、膨大な量の Word 文書を処理したいとき、今回作成したソフトのようなものがあれば役に立つのではないだろうか。

(3) 類似したソフトの例

次に上で紹介した中で制作物と類似したソフトを２つ紹介する。

■ HTML Filter 2.1 for Office 2000

これは Microsoft 社が開発し、無料配布しているものだ。Office

HTML Filter は、HTML 形式で保存した Office 2000 文書に埋め込まれている Office 固有のマークアップ タグを削除するためのツールである。Office 2000 で HTML 文書を作成するときに、Office 固有のマークアップ タグが埋め込まれているとその文書を Word 2000 で再度開いたときに文書を作成したときと同じ書式設定、編集状態などが再現される。しかしこのままではブラウザによってはデザインが崩れる場合があるので推奨できない。

そこで Office HTML Filter を使用すると、Word 2000 または Excel 2000 で HTML 文書の編集を完了した後で、最終的な HTML 文書から Office 固有のマークアップタグを削除できる。タグを削除すると文書のサイズが小さくなるので、Web サーバー上の格納領域が少なくなり、ページのダウンロードにかかる時間も短くなる。

このプログラムは、使用したことがないのでわからないが、私の目指すものに近いように思う。しかし Word、Excel の 2000 シリーズにしか対応していないところが難点である。

Office XP などの Word2000 以降を使用している場合は、HTMLFilter はすでに組み込まれているので、[ファイル] メニューの [Web ページとして保存] を選択した時に、ファイルの種類を「Web ページ（フィルタ後）」に指定すればいい。しかしこの方法をとる場合は、Word 文書作成の際に、書式の中にはタブや均等割付などの失われるものがあることに気をつけて編集しなければならない。

■ Google ドキュメント

Google ドキュメントは、文書（ワードプロセッサ）、スプレッドシート（表計算ソフト）、プレゼンテーション（プレゼンテーションソフトウェア）の 3 つの機能が提供されている。ドキュメントはいずれも

Google のサーバ上に保存されるが、他形式とのインポート・エクスポートも可能である。他のユーザとのドキュメントの共有もサポートしている。一つの文書ファイルは 500KB まで、画像ファイルは 2MB まで保存することができる。

まずこのツールを使うためには Google のアカウントが必要である。アカウントは無料で簡単に登録できる。一度登録してしまえば様々な Google ツールを使用することが出来る。その中にある Google ドキュメントは、上で紹介したソフトとは違い、Google が無料提供するウェブ上で動くオフィスソフトである。ファイルは Microsoft Word 形式（.doc)、リッチテキスト（.rtf)、ODF（.odt)、PDF（.pdf)、テキストファイル（.txt)、HTML 形式 （.html）で保存できる。

まず、変換したい Word 文書を Gmail に送り、Google ドキュメントとして開く。そして Edit HTML で表示する。すると、Word 文書を Google ドキュメントに変換したのち、HTML 化したソースが表示される。その機能が制作したソフトと類似している。

しかし、このツールを使ってできたソースは、余計なタグが完全に取れているわけではない。表示されたソースをコピーし、テキストエディタに貼り付け、機能を使って検索、置換えをする必要はある。規則性が分かりやすい分、Word を HTML で保存した状態のソースよりは改変しやすい。

（4）制作物と類似ソフトの比較

ここで、上述で挙げた例の Google ドキュメントと制作したソフトとの違いを比べてみる。Web 上で今回の内容である、Word 文書を HTML 化したいときに使うツールとして一番多く推奨されていたのが、

Google ドキュメントだからである。

Google ドキュメントはあくまでオフィスソフトであり、Word に近い。違いは様々なところにあり、Google ドキュメントは Word 独特の形式がない。Word 独特の形式がないことで余計なソースがなく、Word を HTML 化したときよりもシンプルなソースができる。

Google ドキュメントを使ったソースの作り方は２つある。１つ目は Gmail に送った Word ドキュメントを HTML 形式で表示し、ソースをコピーするというもの。２つ目は Gmail に送ったものを Google ドキュメントとして開き、Edit HTML で表示するというものである。

(5) 例とする資料について

まず同じ文書を例として全ての資料に使用する。資料については資料編を参照してほしい。

- 例１＝ Word 文書を HTML 形式保存し、IE ブラウザで表示したもの
- 例２＝制作したソフトを使用し、IE ブラウザで表示したもの
- 資料 Word ＝例１のソース
- 資料 Gmail １＝ Gmail 内に送った Word ドキュメントを HTML 形式で表示し、ソースをコピーしたもの
- 資料 Gmail ２＝ Gmail に送ったものを Google ドキュメントとして開き、Edit HTML で表示したもの
- 資料 hikaku ＝制作したソフトを使って作られたソース

なお、資料 hikaku を基準として他の資料と比較していく。

(6) 各資料の特徴

■資料 Word の特徴

資料 Word は最初の宣言文で必要なタグがいくつか書かれていない。制作物についての項目で書いた、HTML に必要なタグのことである。タイトルの後には内部 CSS が組まれている。今回例に使用している文書は複雑ではないために比較的短いものだが、文書内容によっては長く書かれており、ファイル自体を重くする原因となっている。body で、lang 属性を指定しているが、これも meta タグ内にいれることを推奨されているものである。また、文書は div タグでセクションごとに分けられている。

タグ自体は改行されており見やすいが、内部 CSS を組んでいるにも関わらず、本文内で詳細な設定、class 指定や font-family などを書きすぎている。さらに、文と文との間に空行がある場合、 タグの中に を挟んで指定している。これはソース上不要なものである。

■資料 Gmail 1 の特徴

資料 Gmail 1 は最初の宣言文の後、特有の不要なタグが混ざっている。これは、HTML 形式で表示したもののソースをコピーしたことから発生している。また、改行がされていない、内部 CSS が組み込まれているなどの点も見られる。

本文のタグは ごとや
ごとに改行が見られる。 タグの塊を<ul>タグで括っているところもある。元は タグしか存在していないソースなので、Gmail もしくは、Google ドキュメントの何らかの法則性で<ul>タグに変更されたと推測できる。

さらに タグや<font>タグで文字位置や種類を一文一文指定している。これは CSS で指定できることなので省いた方が見やすいソースに

なる。

■資料 Gmail 2 の特徴

次に資料 Gmail 2 を見てみる。こちらはタグ名が大文字、小文字関係なく書かれている。さらに、最初の<html><title><body>などというHTML 文書としては必須のタグも欠けている。<p>タグで改行が行われているのは先ほどと同じである。

先ほどと同様に、<p>タグや<font>タグ、さらにタグが繰り返されており、ソースはさらに見づらいものとなっている。資料 Gmail 1 と比べるとより詳しい設定が書かれている。しかしこれらも CSS で指定できるので、必要最低限のタグのみを残して他は削除したほうがよい。また、先ほども述べたように本文のみを抜き取っているので、<html><title><body>などのタグは後付しなければならない。

(7) 資料 hikaku と各資料の比較

各資料の特徴を述べたところで、資料 hikaku との比較をする。資料 hikaku はこれらの良いところは参考にし、問題点だけを出来るだけ解消したものである。

まず資料 Word の問題点である、Word であることの宣言文と内部 CSS は全て抜き取っている。さらに改行しないスペースである を検索、置換えで 1 個のスペースにし、中身が空になった<p>などのタグを消している。なお、資料 Word のタグごとの改行は見やすいと感じたので取り入れている。宣言文などの HTML に必要なタグが足りないところは追加している。

また Google ドキュメントでソースをいじりたいと思うのなら、最初から Word ではなく、 Google ドキュメントを使って文書を作成するこ

とを推奨する。しかし、Google ドキュメントで表示されたソースを毎回コピーし、テキストエディタに貼り付けるのは手間がかかる。結局のところ Word を HTML 化するのも、Google ドキュメントを利用するのもあまり変わらないのかもしれない。

目標であった、ファイルを軽くするということは成功している。単に Word を HTML 化したソースと、制作したソフトで作られたソースを比べると、ファイルサイズが半分以下になっている。元々、ファイルサイズが小さいものだとあまり変化が見られないが、Word を HTML 化したファイルが 30KB 以上のものなら、制作したソフトを通したファイルは 10KB 程度のものになる。ちなみに Gmail を利用すると 20KB 程度になる。

3　制作物について

(1) ソフトの概要

プログラムのはじめに、python を使用するために宣言しておく。さらに、ファイルを UTF8 でエンコードしている。3 行目に書いてある import,re,sys のモジュールは、関数や正規表現の使用を可能にするために必要なものである。

また、入力したデータは unicode に変換し、書き出すときに Shift-JIS に戻す。データを unicode に変換したのは、日本語が入った正規表現を使用するために必要だからである。

(2) HTML に必要なタグの挿入

ソフトを制作するうえで、手順を考えた。まずは HTML に必要なタグが足りないことに気づいたので加えた。

1. 最初に HTML に必要とされている DOCTYPE 宣言を挿入する。<!DOCTYPE HTML PUBLIC ”-//W3C//DTD HTML 4.01 Transitional//EN”>
2. HTML には、LANG 属性を使ってその文書の言語を明示しておくことが WAI で薦められているので、<HTML lang=”ja”>を挿入する。
3. HTML がどのコードで書かれているのかを明示するために 、<meta http-equiv=Content-Type content=”text/html; charset=shift_jis”>を挿入する。
4. スタイルシートを使用する場合も、ベースとなるスタイル言語を明示しておくようにする。<META http-equiv=”Content-Style-Type” content=”text/css”>
5. さらに外部 CSS を呼び出せるように事前に<link rel=”stylesheet” type=”text/css” href=”style.css”>を挿入しておく。

以上が Word を HTML 化したときに足りないタグであり、HTML 文書に必要な最低限のものである。

(3) 削除したタグの概要

さらに body 内の本文の文書タグに目を通す。HTML タグの中にはいろいろな属性名や属性値を持ったタグが多い。それゆえに一つ一つのタグを検索し、置換えしているとファイル自体が重くなる。そこでグループ分けできるものは一まとめにして、コンパイルし置換えた。また、color:black を消しているのは、黒のフォントで指定されているが基本なので、わざわざ書くことはないからである。

次に span タグに番号を振る。span タグは、<span-1 ...> abcd <span-

2 ...> efg </span-2> hijk </span-1>というように、span 要素の中に別の span 要素が入る可能性がある。<span-1...>という開始タグの方が、属性がなくなったので削除しようとしたとき、それに対応する終了タグは、一番はじめの</span-2>ではなく、</span-1>の方を削除しなければならない。実際には 1 や 2 はもともと入っていないので、どのに対してどのを消せばいいのか分からなくなる。そこで、span が出てくる度に変数 num の番号を増加させてナンバーを振り、が出てくる度に変数 num の番号を減らしていく。こうすれば、どのとどのが対応しているかが分かるのである。

また注意してほしいのは、<p>や内にある style 属性や align 属性は推奨されていないことを知ったうえで残している。これは変換した後でもデザイン崩れを起こさないようにとの配慮からである。最後まで設定するつもりなら、外部 CSS に組み込むことを推奨する。

(4) ソフトの構成

資料 henkan.py の説明をする。

- python を使用するという宣言をする。(1 行目)
- utf8 でコーディング(ソースコードを作成)する。(2 行目)
- モジュール re と sys の使用宣言をする。(4 行目)
- ごとに番号を付ける関数 spannumber を定義する。(6 行目～16 行目)
- 変換するファイルを開き、読み込む。全て読み込み、処理を施した後終わる。(18 行目～21 行目)
- 文字コードを Shift-jis 形式から unicode に直す。(23 行目)
- 二個以上の改行を\kaigyou に置き換える。(25 行目～26 行目)

・一個の改行を空行にし、一行にする。（27 行目）
・二個以上の改行を\kaigyou を 2 個以上の改行に置き換える。（28 行目）
・name の meta タグと link から始まる文字列を空文字列に置き換える。（30 行目～31 行目）
・style タグと/style タグの代わりにスタイルシートを指定する meta タグに置き換える。（33 行目～35 行目）
・二個以上のスペースを一個にする。（37 行目～38 行目）
・コメントを空文字列に置き換える。（40 行目～41 行目）
・改行しないスペース（ ）を 1 個のスペースに置き換える。また、改行しないスペースが複数あった場合も同じようにする。（43 行目～44 行目）
・xmlns 以降のタグを>に置き換える。（46 行目～47 行目）
・改行があった方がいい開始タグに改行 2 個を入れる。（49 行目～50 行目）
・改行があった方がいい終了タグに改行 2 個、または 1 個入れる。（52 行目～56 行目）
・if から始まる文と endif から始まる文の の仮定部分を空文字列に置き換える。（58 行目～59 行目）
・div style から始まる文と layout-grid から始まる文を空文字列に置き換える。（61 行目～62 行目）
・各タグ内にある class 属性と lang 属性を空文字列に置き換える。（64 行目～65 行目）
・<html>を<html lang=”ja”>に変換する。（67 行目～68 行目）

・各タグ内にある text-、mso-、margin-、font family-、background-、tab-から始まる文を’ に置き換える。（70 行目～71 行目）
・style の中味が color:black のみなら、その style タグ全てを消す。そうでない場合は、style タグを残し、color:black のみを空文字列に置き換える。（73 行目～74 行目）
・<span lang> <span dir> から始まる文をのみに変換する。（76 行目～77 行目）
・エラーの出た記号を空文字列に置き換える。（79 行目～80 行目）
・!msorm を空文字列に置き換える。（82 行目～83 行目）
・<o:p>、</o:p>を空文字列に置き換える。（85 行目～86 行目）
・属性値がない場合、属性全体を空文字列に置き換える。（88 行目～89 行目）
・タグを全て消す。<p>タグの方で指定しているのでタグ内でフォントを指定する必要はない。（91 行目～92 行目）
・空白含む’ >’ を’>’ に変換する。（94 行目）
・6 行目～16 行目で指定した spannumber を実行する。（96 行目～97 行目）
・との間に何かある場合、のみを削除する。これを繰り返しているのは、プログラムを上から実行していくうえで、新たに同じような状況が発生するからである。（99 行目～105 行目）
・ 、 など div に関するタグ全てを空文字列に置き換える。（107 行目～108 行目）

・タグとタグの間に何も挟まれていない場合、全て削除する。これを繰り返しているのは、プログラムを上から実行していくうえで、新たに空タグが発生するからである。新たに発生した空タグを検索し、空文字列に置き換えるという作業を繰り返す。（110 行目～114 行目）
・プログラムの実行の家庭でできた余分な空行を、１個の空行に直す。（116 行目）
・<!DOCTYPE～>から始まるくだりをソースの頭にいれる。さらにソースを見やすくするために、\n で改行をいれる。（118 行目）
・文字コードをすべて Shift-jis 形式に unicode し直す。（120 行目）
・プログラムの結果を表示する。（121 行目）

4　最後に

(1) 苦労した点

Word を HTML 化したときのソースのタグの中で、一つ一つ何が必要で何が不必要か考えなければならない。そこで、20 個ほどのサンプルドキュメント文書で試し、どのタグが必要で、どのタグが必要でないかを調べていった。HTML だけでなく、CSS タグの理解が必要だったのだが、確証的ではない知識しか持ち合わせていなかったので、各タグを調べなければならなかった。また、グループ化できるタグをまとめることにも苦労した。

プログラムの内容は、上から実行されていくので、順序を間違えることでエラーが起きることもしばしばあり、気をつけなければならなかった。

具体的にあった例をあげてみる。 のように括弧内全てを消

すのではなく、括弧内の一部分のみを置換えたい場合は注意が必要だった。<span style=’font-size:24.0pt;font-family:” ＭＳ 明朝”’>の文から、font-family:” ＭＳ 明朝” のみを抜きたいとする。すると正規表現では、コンパイルしたい部分を font-family:.+?と書くのだが、これを空文字列に置き換えると、font-family:” ＭＳ 明朝”’ の部分が空文字列になる。しかし、最後の’ まで空文字列に置き換えてしまうと、タグの最初の’ に対応するものが無いことになってしまう。それゆえにコンパイルした後、’ に置き換える必要があった。

(2) 制作したソフトの改善点

制作は終了したが、まだ改善できる点は多々見られる。このソフトは、私とアンケートに答えてくれた数名のゼミ生のサンプルでしか試されていない。なるべくサンプルとして様々なものを揃えたつもりではあるが、違う形式のものがあっても不思議ではない。そこにもし、普段使われていないような記号が書かれていれば、制作したソフトを通して変換した時、ソースに何らかの影響を与えるだろう。現に、サンプルの中には、Word では表示されているが、HTML 化したときに違う文字、もしくは記号に変換されているものがあった。見つかったものはプログラム内で消している。このように、サンプルによっては上手く表示されないものもある。

このソフトが使用されていく中で、徐々にこういったものは見つかるだろう。そのたびにプログラムを書き換え変更する必要がある。

さらにソフトの概要の項目で述べた、「<p>や内にある style 属性や align 属性は推奨されていないが、変換した後でもデザイン崩れを起こさないようにとある程度残してある。」というものも改良の余地が

ある。画像などを使わないような簡単な文書であれば、font 指定タグを省いた方が見やすいソースになる。align にしても、大きくグループ化してから指定したほうが綺麗なソースになる。

以上の点をふまえて、このソフトを使用するのに適した Word 文書は、普段使われていない記号や機種依存文字を使っていない、複雑でない文書がいいことがわかる。文章が長いゆえにファイルが重くなっている文書などには適しているだろう。

一度は Word を HTML 化したときのソース内にある内部 CSS を外部 CSS として抽出できればいいのではないかと考えた。しかし、あくまで Word 特有の形で書かれた CSS であることに気付き、構想を実現するためには Word 特有の CSS を通常使われている形に検索置換えしたうえで、さらに外部 CSS として新しいファイルを作成しなければならないことがわかった。正規表現の知識が無い状態から始めたので、そこまではできないだろうと思い、断念したが、不可能なことではないように思う。

(3) アンケート結果

テーマである「人に役立つものを作る」ためには、ユーザ視点で考えることが大切である。使いやすく、わかりやすいものでないと使用してもらえない。そのためにアンケートを行い、使いやすさなどを調べた。

アンケートは、人文情報学科の同じゼミの３回生と４回生に行った。アンケート内容は、主に５つ。ホームページを作ったことがあるか、タグを理解しているか、CSS を使用したことがあるか、ソフトが使いやすいか、ソフトを通して出来た結果のソースはみやすいか、というものである。人文情報学科では、１、２年のうちにホームページの作成などを

経験しているはずなので、ホームページ作成経験、タグの理解、CSS の使用については経験あり、理解も少しはしているとの結果が出た。次にプログラムの使いやすさの結果だが、これも使いやすいとの声を多く聞くことが出来た。理由は難しい操作がなく少しの説明で出来るから、使い方の説明さえあれば使用できるとのことだった。ソフトを通して出来た結果のソースはみやすいかとのアンケートも、見やすいという結果が出たので、目標の一つは達成できた。アンケートの最後には、ソフトを使った感想を聞いてみた。そこでは、Word 文書を HTML として保存したソースはよくわからなかったので、ソフトを通して出来たソースはとても見やすいとの意見や、目に見えてソースが簡略化されていることがわかるとの意見を得ることが出来た。

しかし、ソースデザインが見づらいなどの声も少数ではあるが書かれていた。これは人それぞれの感性の問題なので仕方ないことだろう。さらにコマンドプロンプトでファイルを探す際、手間取ったという意見もあった。これは、アンケートをしてもらった時に一緒に付けた仕様書にコマンドプロンプトについて簡単にしか説明してなかったことが理由と考えられる。フォルダを置く場所をこちらで指定したのだが、それぞれのパソコンやコマンドプロンプトを置いている環境によって、エラーが出た場合があった。そう考えると多少のコマンドプロンプトの知識も必要だと言える。

これらアンケートの結果を元にソースデザインや仕様書の見直しをし、ソフトを仕上げた。

(4) まとめ

今回このソフトを制作したことで、Word 文書を HTML 化したときの粗や、HTML、CSS タグについて、正規表現についてもより深く学ぶことができた。完成と言えるまでにはいかなかったが、完成に近づけたように思う。さらにソフトを制作するには、日々の変化に対応していかなければならないと感じた。比較の例に出した Google のホームページは現在進行形で発展している。近年の内に今回制作したようなツールが組み込まれる可能性もあるだろう。このように、ソフトや Web サイトだけでなく、HTML や CSS などの形式も見直されてきている。XHTML のような新しい形式もこれから出てくる可能性は大いにある。ソフトを作る際は、現状を理解し、何が求められているかを見極めることが必要だと感じた。

注

（1）Microsoft office Word ホームページ http://www.office.microsoft.com/ja-jp/word/default.aspx

（2）とほほの WWW 入門 http://www.tohoho-web.com/

（3）正規表現メモ http://www.kt.rim.or.jp/~kbk/regex/regex.html#PYTHON

（4）Another HTML-lint gateway http://htmllint.itc.keio.ac.jp/htmllint/htmllintl.html

（5）IT 用語辞典 http://e-words.jp/

（6）「軽々 HTML for Word」http://www.ysdesign.bz/old/downloads/kghtml-Word.htm

（7）Google http://www.google.co.jp/

文献表

平田 豊

2001 『正規表現入門』工学社