HTML ファイルのオンライン変換ツール について

0448051　平　めぐみ

目　次

——HTML ファイルのオンライン変換ツールについて——

1　序論

(1) はじめに

私が所属する福田洋一ゼミのテーマは「人の役に立つデジタルツールの制作」である。その中で、私が制作するにあたっての主題は『HTML ファイルのオンライン変換ツールについて』である。このツールはオンラインにおいて、HTML ファイルに含まれたデザイン要素を持つタグをスタイルシートとし、HTML(1)部分とスタイルシート(2)部分に分離させるものである。このツールは HTML タグの中にデザインタグを含めて書いている人やスタイルシートの使い方に慣れていない人に向けて制作したものである。スタイルシート部分は CSS(3)ファイルを link 要素で読み込ませる外部スタイルシートではなく、HTML 文書の head 要素内に style 宣言を置き、その中にスタイルシート規則を直接含ませる設計にしている。

(2) 現状

(i) HTML とスタイルシートの現状

インターネット上にある Web ページは HTML というマークアップ言語(4)を使用している。これまでの HTML 言語は、タグ(5)を用いて文章の論理構造やページの修飾やレイアウトなどを施してきた。しかし、W3C(6)によって HTML4.0/4.01 が勧告された際に出された『HTML 4.01 仕様書』によると、「ページのスタイルやフォーマットを整えるためには、HTML のプレゼンテーション属性ではなく、スタイルシートを使用すべきであること、スタイルシートによる代替が可能な HTML のプレゼンテーション属性は推奨しないこと(7)」とされている。そのため、現在は文書構造は HTML で設計し、スタイルシートを用いて修飾

できるレイアウト部分はスタイルシートで設計することが好ましいのである。

(ii) HTML 文書に関するオンラインツールの現状

現在、HTML 文書に関したオンラインツールとして代表的なものに『Another HTML-lint gateway[8]』がある。このツールはブラウザを通じて Web 上の HTML 文書やローカルで作成した HTML 文書が文法的に間違っていないかどうかをチェックし、さらに文法上の間違いや表現の不適切部分もチェックし、各チェック項目の間違っている点について書かれた解説も共に提供している。例えば、スタイルシートに代替できる HTML タグが使われている Web ページのエラーチェックをすると、「あまり薦められないタグです。スタイルシートを使いましょう。」と指摘されている。しかし、ただ推奨することを指摘しているだけで、スタイルシートを知っていても、使うことにあまり慣れていない人には、何をしたらよいのかが分かりにくく、また使いにくい。

そこで、Another HTML-lint gateway のようにオンライン上で誰もが簡単にアクセスできることを前提に、デザインタグが含まれた HTML 文書でもスタイルシート推奨タグを自動的に検知し、自動的にスタイルシート化させるツールがあれば、スタイルシートに慣れていない人にとっても敬遠することのない便利なツールになるのではないかと考えた。

2 制作過程

(1) 制作背景

これまで私が個人的に Web サイトを作成する際、今までは HTML ファイル内にデザインタグを含めて書いていた。しかし、先述した通

り、HTML4.0/4.01 の勧告以降、文書構造とデザインやレイアウトは HTML とスタイルシートで別々に設計することが推奨された。しかし、この勧告に従って HTML4.0/4.01 以前に設計されたサイトからデザインタグだけを抜き出し、スタイルシートとして書き換えることは面倒であり、また時間のかかる作業である。そこで、HTML とスタイルシートの混在したファイルを瞬時に HTML とスタイルシートに分けて書き換えるものがあれば、作業時間の短縮になり、次にファイルを制作するときにもやり易くなるのではないかと思った。また、そのような作業をオンラインでできることになれば、より便利である。そして、過去に HTML と CSS を分離させるプログラムが制作してあることを知り、それを利用してオンラインツールに改良することができるのではないかと思い、制作することにした。

(2) 制作にいたるまで

昨年度の卒業論文の一つに『古い HTML の再利用について ―CSS 化ツールの制作―』[1] というものがある。この卒業論文において制作されたツールは、デザインタグが含まれた HTML タグをコマンドプロンプト(9)に変換コマンドを入力して変換し、HTML ファイルと CSS ファイルに分離させるというものである(10)。今回、HTML ファイルのオンライン変換ツールを制作するにあたって、このプログラムを検証した上であれば、制作に利用することができるのではないかと考えた。検証するにあたって、プログラムに不完全な部分があれば修正し、新たにできることは追加し、オンライン化に向けて改良することによって、より良い変換ツールを構築させることができるのではないかと推測した。

以下、先行プログラムを理解した上で、プログラム上に冗長性や機能

不足などの問題がないか、また改良する余地がどのくらいあるかを検証することにした。

(3) 先行プログラムの検証

(i) 検証 ―現状と問題点―

ここでは、先行プログラムに挙げられている改善点及び問題点とそれらに関する私自身の考えを述べていくことにする。以降、これらの考えを踏まえた上で、先行プログラムの改良に入ろうと思う。

■ オンライン化の問題

今後の改善点の一つに「'Another HTML-lint gateway' のように、Web 上で誰もがアクセスできるようにすること」と挙げられている。この「Web 上で誰もがアクセスできるようにすること」は私の今回の制作のメインテーマとしている。

先行プログラムの現状は、コマンドプロンプトに変換コマンドを入力し、ユーザーが変換された HTML ファイルと CSS ファイルを開くという半手動状態である。しかし、このままでは、ユーザーがプログラムファイルを手に入れてから作業することが前提であり、このプログラムツールだけでは汎用性が少ない。

このことから、この先行プログラムをオンライン化するために、Another HTML-lint gateway を参考に Web サイトを作成し、オンライン化を目指す。そのためには、Python で書かれている先行プログラムをインターネット上で起動させるために書き換えることが必要である。また、変換するための HTML 文書を Web サイトからプログラムへ読み込むために新しく cgi[(11)]でファイルを作成することも必要である。

■ table-border の問題

もう一つの改善点として、HTML 文書に表が挿入されている場合に組み込まれる table 要素の問題がある。その table 要素の持つ属性の一つである”border”の変換が不完全であることが改善点に挙げられている。詳細としては、`<table border="0">`と枠線を非表示に設定している場合、枠線が表示されてしまうことであり、改善後、border=”0”と指定されている場合は枠線を表示しないようにしなければならない。

この場合の改善策としては、table 要素に関する関数定義内で新しく条件分岐をさせることが考えられる。もし、属性名(12)に table-border があった場合、’border’ の属性値に分けて処理を行う。border=”0”の場合は、枠線を非表示にさせないため、スタイルシート部分には、border: 0px; と書き出させる。また、’border’ の属性値が”1”以上であれば、枠線を表示させるために現在設計されているように、スタイルシート部分に、border: 1px solid black; と書き出させるままにすれば、表示非表示と区別することができるのではないかと思う。

■ その他の問題点

この他に挙げられていない問題点として、table-border の問題と似ているが、border が指定された太さで変換されていない問題を指摘する。また、セルを連結させるための属性”colspan,rowspan”が使われている場合に、その役割に合った変換がされていないなど、table 要素の変換にはまだ若干の改良の余地が見られた。

そこで、修正にあたって、table 要素について詳しく理解する必要がある。そのため、先行プログラムが制作された時に作られている HTML と CSS の違いについてまとめた表を参考にしようと思ったのだが、table 要素に関しては大まかに違いがまとめられているだけで分かりづ

らいと思った。このことから、table 要素における違いをより分かりやすくなるように再度まとめた。（資料 テーブルタグにおける HTML とスタイルシートの違い.xls 参照）この新たな表は従来通り、table 要素における HTML の開始タグと終了タグ、スタイルシートの開始タグと終了タグとそれらの意味を記述している。また、分かりやすくしただけではなく、スタイルシートに使用される属性や単位を共に記述した。こうすることによって、後から表を確認する度に、そのタグに合った属性値を理解しやすいのではないかと思ったからである。

（ii）検証 ―関数定義―

先行段階で挙げられていた改善点や問題点を踏まえた上で、次はプログラムの詳細な中身を検証していく。このプログラムの構成を見ると、関数で細かく定義された部分が多い。また、この中に混沌としている部分やプログラム文の重複など冗長性がある部分が見つかった。そこで、そこから先述したような問題点が生まれているのではないかと思い、関数定義部分に着目することにした。そこで、このプログラムにおける関数の使われ方を理解するために、関数定義についてまとめた関数相関図を作成した。（資料 関数相関図.xls 参照）この相関図は、それぞれの関数または、正規表現がメインプログラムのどの部分に使われているか、また関数に冗長性がないかを確認するためのものである。

（4）制作にあたって

必要なプログラム言語として、ウェブサイトの制作には HTML を、プログラムの改良及び制作には Python[(13)]及び cgi を使用する。先行プログラムでは、タグや属性名を検索し、置換するという点から正規表現[(14)]を使用している。そこで、正規表現を理解しておくために『Web

プログラマのための正規表現実践のツボ』[2] を用いて、正規表現を学んだ。HTML とスタイルシートやそれらの表現の違いなどにおいては、『最新実用 HTML タグ辞典』[3] 及び、Web 上にあるホームページ作成関連総合サイトを参考にプログラムを構築することにした。

(5) 制作環境

プログラムの制作には KNOPPIX(15)の Emacs を使用した。また、Web サイトの制作には学内の Windows に標準インストールされているサクラエディタを使用した。制作したプログラムや Web ページの修正は、Emacs を使用して施していく。

動作確認は Web サイトを実際に作った上で KNOPPIX 上の Firefox で行う。または、KNOPPIX のシェルから ifconfig コマンドで IP アドレスを受け取り、Windows から接続して確認を行う。確認手順としては、まず Web サイトから変換させたい HTML 文書を読み込み、変換させるための cgi ファイルと py ファイルを介して、変換を行う。(資料 change.cgi、main.py 参照) 変換後は HTML とスタイルシートを分離させた文書で作られた Web ページが表示される。実際の HTML 文書を確認するには、ブラウザのメニューバーや右クリックでソースを確認して、HTML とスタイルシートに分割されているかどうかを確認することができる。

また、変換確認に入る前に使用する HTML ファイルの文法チェックを行う必要がある。なぜならば、ユーザーが作成した HTML ファイルの中身はすべて正しい文法で書かれているとは言えず、間違った文法があれば、プログラムエラーが起こる可能性も高くなるためである。そのため、エラーを起こさないためにも、ある程度正しい文法で書かれた

HTML ファイルを用いることが望ましいのである。

3　オンライン変換ツールの制作

（1）Web サイトの制作

（i）オンライン化するにあたって

先行プログラムでは、コマンドラインで変換させたい HTML ファイルを引数にして渡し、変換後、HTML ファイルと CSS ファイルをプログラムが置いてあるフォルダの内部に書き出すという形式であった。この先行プログラムをオンライン変換ツールにするにあたり、メインプログラム部分を関数”main”とした。（資料 main.py 215 行目参照）また、スタイルシートにおいては HTML 内に直接スタイルシートを記述させ、HTML とスタイルシートに分離させたものを一つのファイルとして表示させるようにした。（資料 main.py 243〜244 行目参照）これまでのように HTML ファイルと CSS ファイルに分けなかったのはページを表示させる上での問題が理由に挙げられる。

Web サイトから変換したファイルはブラウザ上で表示される。先行段階のように HTML ファイルと CSS ファイルを分けて表示するには、ページを分けて表示させなければならず、困難である。そこで、「HTML 4.01 仕様書[16]」でスタイルシートを HTML 内に付加して記述することができると表明されていることから、結果として、HTML とスタイルシートを一つのページにまとめることにし、変換後に変換プレビューの感覚で変換後のページ全体を確認ができるようにした。

HTML 文書の中にデフォルトスタイルシート言語を設定する場合には、スタイル情報を記すために meta 要素を記述する。その際には、`<meta http-equiv="Content-Style-Type"`

content="text/css">という宣言を HTML 文書の head 要素内に置く必要がある。その宣言に続いて<style type="text/css">と style 宣言をすることでスタイルシートを指定することができる。今回のプログラムでは__css__に代入してきたスタイルシート部分を css に置換し、最後に head として書き出すことで、HTML 部分と一緒に表示することにした。（資料 main.py 299〜303 行目参照）

（ii）HTML 文書の読み込み

オンラインツールを作成するにあたって、参考にする Web サイトは先述したように「Another HTML-lint gateway」である。この Web サイトにおける HTML 文書の読み込み方法には　チェックしたい HTML のある URL を指定する方法、　テキストエリアに直接 HTML 文を記述する方法、　ファイルをパソコン上からアップロードしてチェックする方法の三種類がある。このことから、HTML 若しくは XHTML ファイルであれば、自身で保存しているファイルでもオンライン上にあるファイルでもチェックが可能となる。また、出力する文字コードも選択することができる。これらの点をオンラインツールに取り入れ、制作していくことにした。

共通作業として、HTML には form 要素を用いる。（資料 index.html 98 行目参照）　の場合、URL を入力するためのテキストフィールドを<input type="text">で、　の場合は複数行の入力フィールドが必要となるため、textarea 要素を、　の場合は、アップロードするためのフィールドを設けるため、<input type="file">をそれぞれ用いる。

次に cgi スクリプトに関する作業である。　の場合は URL を指定させて読み出すために Python の関数のひとつに urllib.urlopen(url, プロキシ指定) という関数がある。この関数を用いることによって、イン

ターネット上のファイルをローカルにあるファイルと同じように読み込んでアクセスすることが可能である。　の場合はテキストエリア内に入力された HTML の文書を新しく作った cgi ファイルで変換させる。

の場合はファイルをアップロードさせるための Python での cgi スクリプトを記述する。これらの作業を cgi ファイル内で条件分岐を行いながら Web サイトで受けたそれぞれの場合に合わせて変換させる。そして、HTML とスタイルシートを分離させた Web ページへ表示されるという仕組みである。

しかし、URL を取得する方法とファイルをアップロードする方法の二つのパターンを試してみたところ、変換時にエラーが発生してしまった。前者に関しては、KNOPPIX 上から作業をしていたために、プロキシ指定をする上で問題があったと見られる。後者に関しては何パターンか cgi スクリプトを試してみたが、どれも上手く処理させることができなかった。そのため、変換するための HTML ファイルを読み込む方法として、テキストエリアに直接 HTML 文を記述する方法のみ取り入れることにした。

(2) プログラムの制作

(i) 関数の冗長性

関数定義の中に、属性名が一つの場合の関数”one_attr”と属性名が二つある場合の関数”each_property”がある。（資料 result.py 70 行目～84 行目、125 行目～135 行目参照）この二つの関数は HTML タグの属性が一つの場合と二つ以上ある場合の二つに分け、それぞれ定義させている。

先述した関数相関図と先行プログラムを見てみると、each_property 関数は、属性名が二つ以上あるタグの変換”replace_many_attr”という

関数の中で使われている。（資料 関数相関図.xls、result.py 185〜211 行目参照）そのことから、replace_many_attr 関数は属性名が複数の場合に関するタグの変換を行う根源の関数であるということが分かる。そこで、one_attr 関数と replace_many_attr 関数を比較して見ると同じ動きをするプログラム文が含まれていることが判明した。（資料 result.py 79〜81 行目、200〜202 行目参照）よって、one_attr 関数の働きを replace_many_attr 関数の中でも行うことができるのではないかと考え、one_attr 関数と関係する正規表現文を消去し、each_property に改変を加えた。

（ii）テーブルタグの見直し

先行プログラムの検証でも挙げたように、table 要素を変換させるプログラムには改良の余地が見られる。そのため、テーブルタグに関する関数定義を見直すことにした。

■ border の問題

一つ目の問題は、`<table border="0">`と枠線を非表示に設定している場合に枠線が表示されてしまうことである。本来、HTML タグにおける border 属性の役割は、枠線の幅を”1”以上の数値をピクセルで指定して表示させるものである。また、border=”0”と設定されているときにも枠線を表示させないことが可能である。

そこで、この問題の原因が「属性名が二つ以上あるタグの変換の関数 replace_many_attr」にあると考えた。この関数内には、「もし、`<table>`というタグがあれば、CSS に table, tr, th, td { border: 1px solid black;} と書き出す」となるように設定されている。このプログラム文がある限り、枠線の太さが指定されていても、1px で統一されてしまう。そのため、このプログラム文を削除することにした。（資料

result.py 194〜199 行目参照）

この削除によって、<table border="0">の場合、枠線は非表示になった。しかし、その代わりに<table border="1">と指定されている場合は表示されなくなった。<table>というタグがあれば、自動的に table, tr, th, td { border: 1px solid black;} と書き出されるようになっていたからである。そこで、「テーブルタグに属性がある場合の関数 table_each_property」に、”value = value + px solid #000000”と加えた。このプログラム文は、border 属性の値を value に置き換え、値にピクセル単位をあてて、スタイルシートに変換させるというものである。これによって、<table border="0">の場合には枠線を表示させず、また値が”1”以上に指定されている場合にもその値に合わせた枠線を表示させることができた。（資料 main.py 136 行目参照）

table 要素に関する関数に枠線の指定をしたため、table 要素の枠線は指定させることができたが、<tr>,<th>,<td>タグで設計されたセルにおいては、replace_many_attr 関数内のプログラム文を削除しているため、枠線が指定されても表示されなくなってしまった。この解決策として、もし、文書内に table 要素があった場合、div 属性で新しいスタイルシートを設計し、その中に tr, td, th {border:1px solid #000000;} と書き出させ、<tr>,<th>,<td>で設計されたセルにも枠線を表示させるように replace_many_attr 関数の中身を変更した。（資料 main.py 199〜204 行目参照）

また、テーブルの各セルの枠線の表示方法を指定する属性をスタイルシートで書き出すようにした。大半のブラウザでは、隣のセルとすこし離して表示させる separate 属性が規定値だが、CSS2 仕様書上の既定値は collapse 属性のため(17)、ここでは collapse 属性を書き出すことにし

た。（資料 main.py 199〜204 行目参照）

■ colspan, rowspan の問題

二つ目の問題は、セルを連結させる rowspan 属性と colspan 属性が指定されている場合、連結されずに崩れたレイアウトになってしまうことである。この二つの属性はセルの連結というテーブル構造を処理する属性であり、レイアウトや装飾を処理するためのスタイルシートに置き換えることはできない。

しかし、先行プログラムでは、HTML を CSS の属性名に変更させるためのテーブル”attr_table”に含まれていない属性も CSS に書き出されるように設計されていた。（資料 result.py 21〜46 行目, 159 行目参照）そこで、スタイルシートに書き換えることのできない属性はそのまま HTML として残すように設計し直した。（資料 main.py 149〜152 行目参照）こうすることにより、rowspan 属性や colspan 属性のようなテーブルの構造を処理する属性やこれまで CSS に書き出されていた attr_table に含まれていない属性も HTML として残すことが可能となった。

■ テーブルサイズの問題

テーブル全体の幅を指定する場合は、width 属性を用いる。この属性はピクセルで指定すると指定された大きさで固定され、パーセントで指定するとブラウザのウインドウサイズに対する割合となる。例えば、`<table width="300px">`とピクセル単位で指定したものと`<table width="300">`と単位をつけずに指定したものは共に 300px の幅になるが、`<table width="50%">`とパーセントで指定したものはブラウザに対して 50% の幅で表示される。

この HTML をスタイルシートに書き換える場合、ピクセル指定の場

合は table {width: 300px;}、パーセント指定の場合は table {width: 50%;} と記述させなければならない。先行プログラムでは、属性値に'px' の単位を付けて書き出す対応のため、変換させるファイルに最初から'px' 単位付きで指定されている場合やパーセント指定されている場合、table {width: 数値 px + px;} や table {width: 数値 % + px;} と正しい結果が反映されなかった。テーブル全体の高さを指定させる height 属性の場合は HTML においてピクセル指定のみしかできないため、今のままでよいのだが、width 属性においてはピクセル指定とパーセント指定を選択できるように修正を施す必要がある。

そこで、width 属性が使われている場合、"value" に'%' がないことを確認し、その後、数値に'px' を付ける。また、'%' が使われている場合には'px' は付けずにそのままの値を返すようにした。

(iii) 文字コードの問題

HTML ファイルは様々な文字コードで書かれている。主に宣言されている文字コードは Shift_JIS[18], euc-jp[19], iso-2022-jp[20], UTF-8[21]などである。しかし、文字コードを宣言するための'charset=' の部分が宣言されていないものがある。先行プログラムでは、まず文字コードは何で書かれているかを検索し、`<head>`要素内で宣言されていない場合、強制的に'charset=Shift_JIS' とし、宣言されていても識別できなければユーザーに文字コードを尋ねるという形式であった。

オンライン化にあたり、まず考えたのが変換前にユーザー自身に記述している文字コードを先に指定してもらうためにラジオボタン機能を用いることであった。しかし、このオンラインツール自身の文字コードが euc-jp で書かれているため、Shift_JIS や iso-2022-jp などで書かれているページは表示できないことに気がついた。そこで、文字コードは

unicode[22]で変換し代入していたものを euc-jp に固定することにした。（資料 main.py 308 行目参照）

文字コードは euc-jp で固定するのだが、変換時、ユーザーに注意してもらうことがある。一つ目はユーザーが Shift_JIS など euc-jp 以外の文字コードで HTML 文書を書いている場合である。この場合の変換後の対処方法として、ユーザー自身で宣言部分とファイルを保存する際の文字コードを変更をしてもらう必要がある。二つ目は、変換する前に行うことである。例えば、テストページに使用した Web サイトのひとつ「宮内庁ホームページ[23]」に「　仁親王」と表記されている箇所がある。この「　」という漢字は「寛」の旧字体であるため、euc-jp では認識されずにエラーが出てしまった。そのため、変換する前にはファイル内で使う文字に旧字体を控えるか文字コードのチェックを行ってからの方がエラーも起こらずに変換できるだろう。

（iv）画像非表示の問題

Web ページに画像を表示させるためには img 要素を用い、HTML では`<img src="…（画像ファイル名や URL）">`と表記する必要がある。例えば、`<img src="image/top.jpg">`と記述していた場合、今までコマンドプロンプトを介して変換をしていたため、変換するファイルと同じ階層に画像ファイルやフォルダがあれば、変換後、すぐに画像の含まれたファイルを確認することができた。

しかし、オンラインツールでは変換された Web ページが表示される。そのため、`<img src="image/top.jpg">`と書かれたファイルを変換させた場合、変換されたソースファイルの画像表示部分も同じく`<img src="image/top.jpg">`となるため非表示となってしまう。対策として考えたのが、画像を表示させるために`<img src="…">`内に絶対アド

レスとなる画像 URL を指定させることであるが、後々ユーザーが URL 部分を各自で変更することになる。しかし、この作業は、面倒な作業時間を短縮するという制作するにあたっての本来の意図に反してしまう。そのため、ユーザーには変換ファイルを各自保存してもらい、オフライン上で確認してもらうことになるが、変換後の Web ページには画像を非表示のまま処理することにした。

4　結論

2007 年 11 月末、人文情報学科の学生 11 人に制作物である HTML ファイルのオンライン変換ツールを試用してもらい、ツールの使用感や改善点、問題点の有無などに関するアンケート調査を行った。人文情報学科の学生をアンケートの対象に選んだ理由は、この学科では一回生の時に授業でホームページ制作の基礎を学んでおり、HTML とスタイルシートについてある程度は認識しているとの考えからである。このツールを使用するには最低でも HTML についての知識とスタイルシートがどういうものかをある程度知っている必要がある。アンケートでも 11 人全員が HTML とスタイルシートの存在を知っているとの結果が得られたため、アンケートは有効であったと言えるだろう。

良かった点として、変換後にきちんと HTML 部分とスタイルシート部分に分かれて表示されていたことやツール本体の Web ページが見やすいということ、また操作性が良く用途によっては便利に使えそうだという評価を受けた。

問題点としては、Internet Explorer（以下、IE）でソースを確認すると文字化けしてしまうこと、元のページでは左寄せとなっている部分が変換後のページでは中央寄せになっていることが挙げられた。

(1) 問題点と考察

(i) ソースが文字化けする問題

Windows 上の IE において、ソースを確認した場合、文字化けしてしまうという問題については、IE でソースを確認したことが原因と見られる。通常、Windows 上で IE を使用している場合、ソースを表示させると、テキストエディタのメモ帳が起動され、その Web ページの HTML データが表示される。しかし、メモ帳では文字コードとして Shift_JIS が使用されている Web ページでなければ内容を正しく表示することができない。この変換ツールは UNIX 環境の KNOPPIX 上で作成し、文字コードは euc-jp を使用しているため、変換後の Web ページの文字コードも euc-jp となる。そのため、メモ帳では正しく表示することができずに日本語が文字化けしてしまうのである。

この場合の解決策として、使用ブラウザを Firefox にするとソースも文字化けせずに表示することができる。しかし、Windows ユーザーの場合、IE を使うことが多いと思ったため、ツールの説明文に、IE では文字コードの関係でソースを確認することができないことを記述することにした。(資料 html.html 84 行目参照)

(ii) ブラウザによって表示が違う問題

コンテンツの中央寄せを指定するものとして、 タグがある。先行プログラムでは タグが使用されていた場合は、同様の効果が得られる に変換することにしていた。

しかし、ここで起きた問題は、 によるものである。例えば、HTML ファイル内に タグが使用されており、また、その中で table 要素が使用されているとする。従来のまま、変換すると、 は に変換される。その変換

された結果を確認すると、Firefox の場合は、文字は左寄せになる。しかし、IE の場合は、table 要素内で文字の指定をしていないと文字にも div 要素による効果が引き継がれ、中央寄せの表示になる。いくつかのページを変換して確認しているうちに、ブラウザによって表示が変わっていては分かりづらいのではないかと思った。そこで、 タグで指定されていても に変換させず、タグ自体を消去するようにした。（資料 main.py 283～284 行目参照）

こうすることで、中央寄せの表示は全て解除され、変換後のソースを確認すると タグは取り除かれている。ユーザーには後々、 タグで指定していた部分を各自で修正してもらうことになってしまったが、ブラウザでの見え方の差異を失くすための処置として タグを変換処理しない方法が最適だったのである。

（iii）HTML 文書の文法チェックの必要性

このオンライン変換ツールを使用する際にはあらかじめ、HTML 文法をチェックしておく必要がある。この作業は先行プログラムで確認を行う段階でも必要とされていたことである。

先述したとおり、これまでの動作確認において、変換する HTML ファイルの文法チェックをしてきたが、この作業は必須であることがわかった。なぜなら、ユーザーが作成した HTML がすべて正しいものとは言えないからである。例えば、HTML ファイルの文法が一つでも間違っていると、正しく変換したくてもエラーが起こってしまったり、スタイルシートの変換が上手くいかずに間違ったレイアウトになることが起きた。

本来、スタイルシートは、拡張デザインタグによって、HTML が複雑化してしまった問題を解決するために文書構造とデザインレイアウトを

切り離して設計するために提唱されたものである。そのため、HTML の文法がしっかりしていないと、スタイルシートを活かすことができず、おかしなレイアウトになることもある。このことから、今後、ツールとして使用する際、エラーを出さないため、また間違ったレイアウトに変換させないようにも事前の文法チェックは必然だと言える。

(2) 自己評価と今後の改善点

今回は『HTML ファイルのオンライン変換ツールについて』というテーマの元、制作を行った。私自身も今まで Web サイトを作ることがあり、そのときは HTML4.0/4.01 以前を使用することが多かった。そこで、HTML4.0/4.01 以前に作られた HTML ファイルを瞬時に HTML 文書とスタイルシート文書に分離させるという発想に惹かれた。またその先行プログラムに触れ、オンラインツールとして発展させることができれば、より便利で効率の良い HTML ファイルの作成支援ツールになるのではないかと思い、卒業制作として着手することにした。

このツールは変換したい HTML ファイルの文書を Web サイトのテキストエリアに貼り付けることで HTML とスタイルシートが分離され、変換されたページがプレビューとして見れる仕組みになっている。当初に挙げた問題点の一つであるコマンドラインで変換していた「プログラムツール」を「オンラインツール」化することは達成できた。また、そうしたことによって、誰もがアクセスできるようになって、汎用性は広がり、少しはオンライン変換ツールとして役に立てるのではないかと思う。

しかし、機能的にはまだ未熟なツールであるだろう。例えば、UNIX 環境で作成したことで文字コードは euc-jp に限定されてしまったり、

最初に挙げたもう一つの問題点であるテーブルの変換プログラムも不完全なまま終えてしまったこと、またそのことで一部変換後にユーザーに修正を施してもらう必要があるなど、半手動変換を完全に脱したとは言えない。

今後の改善点としては、 が使用されている場合の table 要素の文字位置を使用ブラウザによって差異が出ないようにすることである。IE で見た場合、文字位置の指定がされていないと中央に寄ってしまう。そのため、 が使用されている場合の table 要素において、文字位置の指定がされてない場合には text-align を用いて強制的に左寄せに指定させるなどの改善策が考えられる。

もう一つは、ファイルの読み込み方法の追加が挙げられる。現在はテキストエリアに直接 HTML 文書を貼り付けて読み込む方法だけである。そこで、今回はエラーが出て断念してしまったが、アップロードする方法と URL から読み込む方法を追加できれば、より汎用性が広がり、使いやすいツールとなるのではないだろうか。

また、ページ表示の都合上、今までのようにスタイルシートを CSS ファイルとせずに HTML 文書内に直接付加させているが、実際スタイルシートは CSS ファイルとして分離させる方が後で修正しやすいのではないかと思う。そこで、もし HTML ファイルと CSS ファイルに分けるとするならば、一旦プログラムで変換させた二つのファイルを、Web サイトにボタン機能を追加し、HTML と CSS、二つのファイルを保存させることができるようにしたい。

(3) 今後の展望

昨今、Weblog[24]や SNS[25]が普及し、HTML で作られた個人のホームページは年々数少なくなっているのではないだろうか。また、スタイルシートの概念が広まってきたことで、昔ほど HTML とスタイルシートが混沌としたサイトも少ないのではないかと思う。また、スタイルシートの導入に関するアンケートの回答では「スタイルシートを使っている」と答えた学生が半数であった。

しかし、HTML4.0/4.01 以前から Web サイトを作成場合や自分で Web ページを初めて作ってみたのは良いが HTML とスタイルシートの使い方が曖昧な場合など、このオンライン変換ツールを使うことによって、少しでも効率の良い Web サイト作成に役立てることができるのではないかと思う。

(4) 最後に

今回、このオンライン変換ツールを制作するにあたって、ツールを開発していくことの難しさを実感した。今までは新しくプログラムを開発することの方がプログラムを修正することよりも難しいことのように思っていた。しかし、今回は先行プログラムがあり、まずはそのプログラムの見直しから入った。最初にプログラムを制作するということは如何様にも融通を効かせられるということである。しかし、そのプログラムを最初とは別の人間が修正するということは、当然簡単な部分は既に終えられており、難しい部分が課題として残されているということである。そのとき、先人がどのような考えで制作に至ったか、どのようにプログラムを組んでいたかなどを汲む必要がある。この度、修正・改良を進めていくうちに、プログラムを修正していくことは開発するよりも難

しいということに気づかされた。

また、「人の役に立つものを作る」ということについても、ただ対象となるユーザーに合わせて作ればいいというわけではないことが分かった。大切なことはユーザーにとって使いやすく分かりやすいことであり、また希望やニーズを詰め込んでこそ「人の役に立つもの」と言えるのだろう。

今回は先行プログラムを修正し、新たなものを作り上げるということを卒業制作として進めてきた。今までの良いところは継承し、問題があるところはできる限り直してきた。使い方をオフラインからオンラインに変えたことで新たな問題も発生した。しかし、改善することにおいて新しい問題が発生することは避けられないことだと思う。私はこの制作を通じて、少しでもより良いものを制作したいという意識を強く持つことが大切なのだということを実感することができた。

注

（1）HTML = Hyper Text Markup Language
Web ページを記述するためのマークアップ言語で W3C が作成している規格。

（2）Web ページのレイアウト（見栄え）を定義するもの。

（3）CSS = Cascading Style Sheets
Web ページのレイアウトを定義する規格。

（4）文書の一部を「タグ」と呼ばれる特別な文字列で囲うことにより、文章の構造や修飾情報を文章中に記述していく記述言語。

（5）「<」と「>」で囲まれた標識。

（6）W3C = World Wide Web Consortium
WWW で利用される技術の標準化をすすめる団体。

（7）HTML 4.01 仕様書邦訳— 4 適合条件:必須事項と推奨事項
”http://www.asahi-net.or.jp/ SD5A-UCD/rec-html401j/conform.html”

（8）”http://openlab.ring.gr.jp/k16/htmllint/htmllint.html”

（9）MS-DOS や UNIX 系 OS など、キーボードから文字で命令を入力して操作を行う CUI において、システムが命令入力を受け付けられる状態にあることを示すために表示される記号。ユーザはこれに続けてコンピュータへの指示を入力し、改行キーなどで決定する。

（10）コマンドプロンプト上で「python result.py 変換させたい HTML ファイル名」と入力する。

（11）cgi = Common Gateway Interface
Web サーバが、Web ブラウザからの要求に応じて、プログラムを起動するための仕組み。

（12）HTML のタグに指定することができる属性（attribute）で、例えば、border=1px の場合、border の部分を属性名、1px の部分を属性値と呼ぶ。

（13）プログラミング言語のひとつで、コンパイルを必要としないスクリプト言語。

（14）文字列のパターンを表現する表記法。

（15）ハードディスクにインストールすることなく、CD1 枚で利用可能な Linux ディストリビューションの一つ。

（16）HTML 4.01 仕様書邦訳― 14.2 HTML へのスタイル付加 ”http://www.asahi-net.or.jp/ SD5A-UCD/rec-html401j/cover.html”

（17）”http://www.tohoho-web.com/css/reference.htm#border-collapse” より

（18）日本語文字コードの一つ。

（19）日本語だけでなく複数バイト言語の各国の文字コードが規定されている。

（20）JIS 規格によって規定されている日本語の文字コードの一つ。Shift JIS コード、日本語 EUC と並んでインターネット上でよく使われる。

（21）1 文字を 1〜6 バイトの可変長の数値に変換する。

（22）すべての文字を 16 ビット (2 バイト) で表現し、1 つの文字コード体系で多国語処理を可能にしようとするもの。

（23）宮内庁ホームページ ”http://www.kunaicho.go.jp/” より

（24）個人や数人のグループで運営され、日々更新される日記的な Web サイトの総称。

（25）SNS = Social Networking Service

人と人とのつながりを促進・サポートする、コミュニティ型の Web サイト。

参考文献

[1] 成田有希『古い HTML の再利用について —CSS 化ツールの制作—』. 大谷大学. 2006 年度卒業論文.

[2] 高橋良明『Web プログラマのための正規表現実践のツボ』. 九天社. 2005

[3] 株式会社アンク『ホームページ辞典 第 3 版』. 翔泳社. 2004

参考 Web サイト

Python Japan User's Group　オンラインドキュメント
（http://www.python.jp/Zope/links/python_documents）

内田明　HTML 4.01 仕様書邦訳
（http://www.asahi-net.or.jp/ SD5A-UCD/rec-html401j/cover.html）

Yoichi Fukuda　大谷大学人文情報学科　講義ノート
（http://tibet.que.ne.jp/otani/lecture.html）

k16@chiba.email.ne.jp　Another HTML-lint gateway
（http://htmllint.itc.keio.ac.jp/htmllint/htmllint.html）

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

高橋永治　TAG index
（http://www.tagindex.com/）

Incept Inc.　IT 用語辞典 e-Words
（http://e-words.jp/）