第8章　Web パブリッシング入門

この章について

この章では Web パブリッシング、特にそこで用いられる出版用言語である XHTML について解説します。ここで「言語」が意味しているのは、コンピューターが理解することのできる形で記述されているということですが、心配する必要はありません。XHTML は、人間にとっても可読な形で記述されている出版用言語です。慣れは必要ですが、初心者でも概念さえ理解できれば 1 時間ほどで簡単な Web ページを制作することができるようになります。

といっても、凝ったページを制作できるようになるまでの知識をここで取り上げるわけではありません。配色やデザインなど、Web パブリッシングは多分にセンスの問題でもあります[1]。ここでは、誰もがルールに従って記述していけば作成することのできる、シンプルで効果的な Web ページの作成を目指します。

ここで重要なのは「構造化文書」との関係です。XHTML は Web パブリッシングのためのコンピューター言語ですが、この言語が直接記述するのは、文書構造です。逆に言えば、文書構造がしっかり組み立てられている文書であれば、これを Web ページとして記述し直すのは非常に簡単なことなのです。本章は XHTML の解説が目的ですが、同時に構造化文書について学習してもらうことも目的にしています。

XHTML は Extensible HyperText Markup Language(拡張可能なハイパーテキストマークアップ言語) の略であり、後述するように従来から Web パブリッシングに用いられてきた HTML の後継として、XML という言語体系を基に記述されている出版用言語です。HTML は、Web ブラウザ[2]開発を巡る競争や対立、プログラム上の欠陥や仕様からの逸脱、その他諸々のために、誤った解釈をされたり不正な書き方が許容されたりしてきました。XHTML ではそのような曖昧さはなく、ある程度厳密に記述することが求められます。これは一面では面倒に思えるかもしれませんが、誤りが検出されやすく、修正が容易であると考えることもできます。

XHTML にもいくつかの種類がありますが、HTML 4.01 との互換性を持った最も基本的な XHTML である、XHTML 1.0 を中心に解説します[3]。

注意しなければならないのはコンピューター言語として XHTML が正しく記述されているかどうかというのも重要ですが、記述されている内容もしっかりとしていなければならないということです。正しい XHTML によって、内容の正しくない文書を記述しても意味がないのです。正しい XHTML によって記述する意義は、記述されている内容をより多くの読者へと効率的・効果的に伝達するところにあるのです。

皆さんの頭の中にある情報は、誰かに何かの形で伝えてこそ価値を持ちます。大学で学んだことや新たに生み出された知識の伝達は、多くの場合口頭発表や論文という形をとりますが、今では Web パブリッシングも重要な柱となっています。皆さんが正しい知識を身につけ、より多くの人と知識を分かち持つことを期待しています。

---

[1] とはいえ、配色やデザインにも理論はあります。後述するように、アクセシビリティやユーザビリティという観点からも重要なポイントではあるのですが、ここでは一部配色について述べるのみに留めます。詳しくは参考文献を参照してください。

[2] Netscape 社の Netscape Navigator や Microsoft 社の Internet Explorer など。

[3] 現在の最新版は XHTML 1.1 であり、XHTML 2.0 も草案が公開されている状態ですが、XHTML 1.0 をしっかり理解すれば十分これらにも対応することができます。

8.1　Web パブリッシングの全体像

Web パブリッシングを始める前に、その全体像がどのようなものであるかということを解説します。とりあえずすぐに始めてみたいという方は 8.3「最初の XHTML」から読み始めても構いませんが、Web パブリッシングの基本的な原理を理解しておくのは、実際の作業を進める際にも重要なポイントです。

ここでは、(1) クライアント・サーバーモデル、(2) ファイル形式の 2 つについて述べます。

8.1.1　クライアント・サーバーモデル

Web は、ネットワークを通じた情報の取得です。その通信としての性質を十分に理解しておくのは、Web パブリッシングだけでなく、単に Web を利用するというだけであっても全体像の理解には必要なことです。ここでは、Web ブラウザが誰とどのように通信を行っているのかということについて、その概要をおおまかに解説します[4]

Web という通信の基本的枠組みは、「クライアント・サーバーモデル」です。これは、Web における登場人物がサービスを提供する「サーバー」と、サービスを受ける「クライアント」の 2 者であることを意味しています[5]。ここで「クライアント」とは、皆さんの利用している PC に入っている Web ブラウザのことです。Microsoft Internet Explorer や、Netscape Navigator、Mozilla Firefox などがその代表です。サーバーは、ここでは Web のサービスを提供するサーバーですので「Web サーバー」と呼びますが、これはネットワーク上のコンピューターで動作しているプログラムです[6]。

自分が Web ページを閲覧する際の流れを考えてみると理解しやすいと思いますが、Web サーバーは、頼みもしないのにクライアントに情報を送りつけてくるようなことはしません。クライアントからの要求 (リクエスト) を待ち、これに応じて情報を受け渡すのです。つまり、Web の閲覧を始めるのは、クライアントからのリクエストということになります。Web ブラウザ、例えば Internet Explorer の「アドレス」[7]欄に後述する URL を入力するとか、「お気に入り」のどれかを選択するといった操作によって、このリクエストがクライアントから Web サーバーへと送信されます (図 8.1・図 8.2 参照)。URL を入力するにしてもお気に入りを選択するにしても、実際に行っているのは、ユーザによる Web ブラウザに対する「URL」の指示です。

図 8.1: Internet Explorer の「アドレス」

図 8.2: Internet Explorer の「お気に入り」

ここで URL とは Uniform Resource Locator の略です[8]。Resource とは直訳すれば「資源」ですが, これはインターネット上に散在する様々な情報（データ）であると考えてください。具体的には「ファイル」という形式をとる情報であることがほとんどですが、必ずしもそうではないことがあり

[4] ここで「おおまか」と書いているのは、ここで示す単純なモデルは Web 創生期の頃から利用されているものであって、これに当てはまらないようなネットワーク構成がよく利用されており、また日々新たなモデルが開発されているからです。もっとも、どのような場合でも基本形は変わりません。

[5] 早稲田大学の場合はこれに「プロキシ」が加わりますが、詳細は省略します。

[6] Web サーバーとしては Apache Software Foundation の Apache、Microsoft 社の IIS(Internet Information Services) などが代表です。

[7] この、Internet Explorer における「アドレス」には後述する URL を入力しますので、この表記は誤用であると思われます。「アドレス」や「お気に入り」は、あくまでも Microsoft のソフトウェアにおける呼称であることに注意しましょう。

[8] URI(Uniform Resource Identifier) という、より広い概念を指す語が利用されることもあります。

ます。Locator は「指し示すもの」という意味になりますので、URL の意味するところは「統一的な方法でインターネット上の情報のありかを表記したもの」ということになります。

図 8.3 に URL の例と意味を示します。URL の基本構成は「スキーム」「オーソリティ」「パス」の 3 つです。

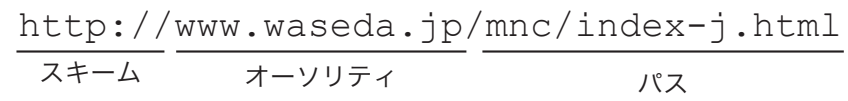

図 8.3: Uniform Resource Locator,URL

スキームには「http」や「ftp」などが入ります。スキームはリソースにアクセスするための枠組みであると考えてください。Web の場合は、HTTP(Hyper Text Transfer Protocol) という「プロトコル」[9]を利用して通信を行うため、URL のスキームは「http」となっています。

「オーソリティ」は、そのリソースを管轄しているコンピューターであると考えると良いでしょう。インターネット上のすべてのコンピューターや通信機器には、それぞれユニークな管理上の数字が割り振られています。これを IP アドレスといい、0 から 255 までの数字を 4 つ、ピリオドで区切って並べたものです。例えば、133.9.1.3 のようなものです。この IP アドレスを利用してインターネット上のコンピューターを一意に識別することが可能です。このような IP アドレスの欠点は覚えづらいということで、これを解決するために人間が記憶しやすい名前 (ドメイン名) と IP アドレスの変換データベースを利用することもできます。例えば、「ns.cfi.waseda.ac.jp 」は「133.9.1.3 」と変換されます[10]。

「パス」は、そのオーソリティ内のどこにリソースがあるかということを示しています。パスは、スラッシュ「/」で区切られた階層的な表記が行われるのが普通です。つまり、フォルダ (ディレクトリ) の概念がここにも適用されています。パスで表記されるのは、具体的なリソース (ファイル) に至るまでのディレクトリと、具体的なファイル名です。ただし、ファイル名が省略される場合もあります。例えば、初めて訪問するサイトがあったとして、スキームとオーソリティ (または一部のパス) だけ知っている (例えば http://www.yahoo.co.jp/ など) という場合も多いはずです。このような場合でもアクセスすることができる、というわけです。

ファイル名が省略された場合は、そのディレクトリのインデックスファイルが指定されたものとみなされます。インデックスファイルとは、そのディレクトリの玄関となるべきページです。通常、インデックスファイルは index.html とか index.htm といったファイル名ですが、これはサーバーの設定によって異なります。試しにブラウザで、http://www.yahoo.co.jp/ と http://www.yahoo.co.jp/index.html の両方にアクセスして同じ結果が得られることを確認してみてください。

ここでおさえておかなければならない重要なポイントは、Web はこのようにファイルをコンピューターからコンピューターへと転送するためのシステムである、ということです。HTTP の後半二文字が「Transfer Protocol」の略であることからも分かるように、本質的に Web はファイル転送のためのシステムなのです。

[9]複数コンピューターの間で通信を行う場合、あらかじめどのような手順で通信を行うのかを取り決めておく必要があります。そのような通信方式のことをプロトコルといいます。コンピューター間の通信は電気や光、電波などを利用して行われますが、どのように電気を流すかというような物理的な通信方式と、電気が流れるとして具体的にどのようにしてデータを送受信するかという論理的な方式の 2 つがあります。URL における「スキーム」は、多くの場合論理的な通信方式を指定しています。

[10]このような変換システムを DNS(Domain Name System) といいます。

8.1.2　ファイル形式

Web パブリッシングは、大きく分けて (1) 文字で記述された文書とデータ (2) それ以外の (多くの場合マルチメディア) ファイルという、二種類のファイル形式を 1 つのページに混在させることで成り立っています。簡単に言えば、Web ページの中に文字や絵などが混じって存在している、ということです。

文字による文章と後述する文書構造は、XHTML では「テキスト」という形式で、同時に表現されます。テキストという形式で文章と文書構造を同じファイルに記述します。この、文書をその構造とともに記述するという点が、XHTML がその他のデータ形式とは異なる特徴的な点です。

「ファイル」の概念については既に学習していますが、注意しなければならないのはファイル名です。通常、テキストファイルはその拡張子を「txt」としますが、XHTML の場合は「html」とします[11]。ファイルを XHTML で記述したら、そのファイル名の拡張子部分を「html」とするのを忘れないようにしましょう。

一方、様々な Web ページを見渡せば文字だけで制作されている Web ページはむしろ少数派であることは明らかで、多くのページは図、絵、写真、動画、音声といった「マルチメディア」を駆使して作成されています。こうした情報はテキストファイルとして共存させることは難しいので、本文や体裁とは別のファイルとして保持しておき、必要なときに呼び出すという形をとります。

つまり、写真などは別ファイルにしておいて、必要に応じて「ここに写真を埋め込む」という情報を、体裁情報と同じような形で XHTML 中に記述する、ということです。これは、写真、絵、動画などテキスト以外のファイルすべてについて同じことが言えます。

ただし、Web ページ中に埋め込むことのできるファイル形式にどのようなものがあるかということは、Web ブラウザによって異なるということに注意が必要です。例えば、Adobe Flash という著名な形式のファイルがありますが、これは多くの場合「プラグイン」という形のソフトウェアを別途インストールしなければ通常の Web ブラウザでは参照できないことが多いはずです。

絵や写真ならほとんどの場合 Web ブラウザが標準で対応しているのであまり問題にはなりにくいかもしれませんが、自分の利用しているコンピューターで参照できるからといって、他の人も同じように参照できていると考えるべきではないということを理解しておきましょう。この章で学習しているのは広く言えば情報発信の方法論ということになりますが、受信する側のことをよく考えて情報発信しなければなりません。

XHTML は、拡張子が html で内容はテキストであるとして、Web パブリッシングでよく利用されているファイル形式を表 8.1 にまとめておきます。

8.1.3　この節のまとめ

Web は、サーバーとクライアントの二者間における通信です。サーバーもクライアントもコンピューターであり、クライアントがサーバーにリクエストを行い、サーバーがこれに応えることで Web の閲覧が行われます。リクエストは、クライアントからサーバーへと URL を送信することを通じて行われ、これはリソース、つまりファイルの送信要求に他なりません。Web サーバーはクライアントに対して、Web ページの本体である HTML ファイルやそこで読み込まれることになっている画像などのマルチメディアファイルを送信し、クライアントはこれを表示します。

サーバーとクライアント間で送受信されるファイルには様々な種類があります。大きく分けて Web ページ内に表示されるもの (XHTML 本体、画像など) と他のプログラムで処理される文書 (Word 書

[11]「htm」でもよい場合があります。これは後述する web サーバーの設定に依存します。従来は「html」という拡張子しか用いられていませんでしたが、MS-DOS や Windows 3.1 において拡張子が 3 文字固定であったことなどから、「htm」という拡張子が生み出され、いまだに利用されています。多くの場合「html」という拡張子が利用できないことは、ほぼ無いと思われます。逆に言えば、「html」という拡張子で不都合がある場合は、利用しているサーバーの管理者に質問とお願いをしてみると良いでしょう。

表 8.1: Web パブリッシングで一般的なファイル形式

形式	拡張子	備考
JPEG	jpg	Joint Photographic Experts Group の略。画像形式を開発した団体名称がそのままファイル形式名となったもの。高い圧縮率が特徴で、デジタルカメラを始めとしたフルカラー画像によく利用される。
GIF	gif	米国パソコン通信の大手であった CompuServe 社によって開発された画像フォーマット。256 色を表示することが可能で、複数の画像を格納してのアニメーションなども可能である。
PNG	png	Portable Network Graphics の略。GIF の後発として開発されたため、あらゆる点で GIF より優れており、フルカラー画像を扱うこともできる。ただし、可逆圧縮であるため、多少の劣化が気にならない場合は JPEG を、劣化が許されなかったり図表の場合は PNG を使うとよい。
PDF	pdf	Portable Document Format の略。Adobe 社による電子文書の規格。やはり Adobe 社が無償配布している Adobe Reader を利用すれば、閲覧だけは自由に行うことができる。変更されたくない文書を配布する際によく用いられている。
Word	doc	Microsoft 社の Word 文書。VBA(Visual Basic for Applications) というマクロ言語を含んでいる場合もあるので、内容が分からない場合は開かない方がよい。また、開かなければならない場合は Word で VBA を無効にするべきである。
Excel	xls	Microsoft 社の Excel 文書。VBA(Visual Basic for Applications) というマクロ言語を含んでいる場合もあるので、内容が分からない場合は開かない方がよい。また、開かなければならない場合は Excel で VBA を無効にするべきである。
CSS	css	Cascading Stylesheet の略。Web パブリッシングにおける体裁情報である。
VBscript	vbs	Visual Basic Script の略。BASIC を基礎とする Microsoft 社による言語である。このファイルは、内容が分からない場合は決してダウンロードしたり開いたりしてはならない。コンピューターウィルス等、悪意のあるプログラムである場合も多い。内容はテキストファイルである。
JavaScript	js	Netscape 社が開発した言語。Web ブラウザ上で動作する言語であり、ほぼすべての Web ブラウザがサポートしている。ただし、サポートの程度や言語仕様は Web ブラウザによって異なる。Web ブラウザのセキュリティ設定で無効にすることも可能だが、昨今これを利用した Web アプリケーションが流行しつつあるので、必要に応じて有効にすると良い。内容はテキストファイルである。

類や PDF)、プログラム (JavaScript や VBS) があります。ファイルの種類は拡張子で判断することが可能ですが、前述したように Web はファイル転送のためのシステムです。信頼できないサイトにアクセスしない、内容の分からないファイルはダウンロードしないのが重要です。これを Web パブリッシングを行う側の観点から考え直せば、Web ページ制作を行う前の心がけが自ずとできるものと思います。

8.2　XHTML 制作のための環境整備

ここでは、XHTML の制作に必要な PC の環境を整え、また必要な知識を確認します。「テキストファイル」と「テキストエディター」、「拡張子」というキーワードが何を意味するか理解できており、自分の好みのエディターおよび FTP ソフトウェアを利用できている人は、この章をスキップしても構いません。

8.2.1　エディター

XHTML は、人間がデータを直接参照して読み書きすることができる、「テキスト」という形式のデータです。この種のデータを作成するのに必要なのは、「テキストエディター」という種類のアプリケーションです。テキストエディターとしては、Windows なら「メモ帳」(スタート→プログラム→アクセサリ→メモ帳)、Macintosh なら「テキストエディット」を利用することができます。どちらでも同じようなデータを作成することが可能です。

これらの標準添付のテキストエディターを使っても構いませんが、フリーソフトウェアないしシェアウェアという形で配布されているエディターを利用することもできます。エディターを別途用意することの利点は、高機能であるということに尽きます。例えば、XHTML では前述のように文章の本文と構造情報を同じファイルの中に記述します。この構造情報は「タグ」と呼ばれることは前述の通りですが、エディターによってはこのタグを本文の文章とは違う色で表示してくれたり、タグの入力支援機能を持っています。

またテキストエディターはテキストの入力に特化したソフトウェアですので、ワードプロセッサのように動作が緩慢ということもなく、軽快に動作します。

高機能で軽快に動作する、かつ無料であれば、利用しない手はありません。ここでは Windows 用と MacOS 用それぞれ 1 つずつ紹介しておきます。

- Windows
  TeraPad (`http://www5f.biglobe.ne.jp/%7Et-susumu/`)
- MacOS
  mi(`http://www.mimikaki.net/`)

これらはいずれも無償で利用することができますが、決して自動的に生み出されている訳ではなく、作者の方々が貴重な時間を割いて作成し、公開してくれているソフトウェアです。感謝の気持ちをもって利用すると同時に、本当に感謝したらお礼のメールを書くこともできますし、mi の場合は任意で使用料金を受け付けてもいますので、支払うと良いでしょう。

これらのソフトウェアの利用方法は、それぞれのソフトウェアに添付されているマニュアルや Web ページを参照してください。テキストエディターはキーボードを打鍵すれば文字が入力されるというプログラムです。重要なのはファイルを作成できること、既存のファイルを開くことができること、プログラムを安全に終了させることができること、そしてこれはエディターとは無関係ですが作成したファイルをコンピューター内で正しく管理することができるということです。

なお、早稲田大学のコンピューター教室における標準環境では「秀丸」というテキストエディターがインストールされています。

8.2.2　FTP ソフトウェア

FTP は、File Transfer Protocol の略です。「ファイル転送プロトコル」と訳されますが、文字通りコンピューター間でファイルを転送するための通信方式です。HTTP に引き続き、別のファイル転

送プロトコルが登場したことになります。なぜ、わざわざ別のファイル転送用ソフトウェアが必要なのでしょうか。

HTTP は Web サーバーと Web クライアントの間で行われる通信でした。クライアントはサーバーから文書を受け取ります。ここで、皆さんは単に Web を閲覧するということでなく、Web ページを制作する側です。とすると、皆さんがこれから作成する文書 (XHTML や画像など) を Web サーバーに設置しなければなりません。これは、どのようにして設置するのでしょうか。

Web を閲覧する際には、多くの場合 ID やパスワードによる認証は必要ありません[12]。もともと自由に参照してもらうことを前提にして出版を行うのが原則だからです。

しかし、自由に行ってもらうのはあくまでも閲覧であって、その Web ページを修正したり新しいページを作成することまで自由に行ってもらうことは意図されていません。他者の作成した Web ページを勝手に書き換えるのは明白な犯罪行為です。Web ページについては作成や編集と閲覧では完全に非対称な作業であるということです。従って、FTP という枠組みの中で ID とパスワードによる認証を利用して、Web ページをサーバーに設置したりこれを書き換えたりするということがしばしば行われます[13]。

FTP もクライアント・サーバーモデルに基づいています。FTP サーバーがあり、FTP クライアントがあるということです。この場合、FTP サーバーはファイルの設置や修正の対象となるコンピューターであり、ここでは Web サーバーと同一です。FTP クライアントは、読者の皆さんが操作するコンピューターです。

FTP は Windows にも標準で付属している機能ですが、「コマンドプロンプト」からコマンドライン入力を行って利用しなければならず、とても使いやすいものとは言えません。そこで、よりユーザフレンドリーな FTP クライアントを利用しましょう。

Windows にも Macintosh にも無償で利用することのできる FTP クライアントがあります。

- Windows: WinSCP
  `http://winscp.net/eng/docs/lang:jp`
- MacOS: Fetch [14]
  `http://fetchsoftworks.com/`

8.2.3 その他のソフトウェア

XHTML の制作を行い、これをサーバーに転送するだけならテキストエディターと FTP ソフトウェアがあれば十分です。しかし、その他にもいくつかソフトウェアがあると便利です。

まず最初に、作画や画像処理のためのソフトウェアです。「百聞は一見にしかず」というように、言葉を尽くしても伝わりづらいことが一枚の写真や絵で伝えられることもあります。それほど大げさでなくても、自分で絵を描くにせよデジタルカメラによる写真を加工するにせよ、ちょっとした画像を Web ページに取り込みたいことはよくあることです。

一方で画像処理は高度なソフトウェアであり、販売されているものは高価なものが多いようです。そこで、無償で利用できるソフトウェアを紹介しておきます。GNU Image Manipulation Program、GIMP です。

もともと英語のソフトウェアですが日本語版もあり、マニュアルも翻訳されています。

---

[12]必要な場合もありますし、サーバーの機能によっては認証を求めることも可能ですが、ここでは不要な場合に限って解説しています。

[13]FTP 以外にも色々な方法があります。組織（大学・会社）やプロバイダによって採用している方式が異なります。

[14]シェアウェアであり 25 ドルの支払いが必要ですが、教育および寄付行為が税控除の対象となる慈善事業（アメリカでのことと思われますので、日本での扱いは不明です）については無償のライセンスを申請することができます。Fetch Softworks のページ（上記 URL）の左段「Licensing」に「Educational/Charitable」というリンクがありますので、こちらから申請してください。Fetch には日本語版もありますが、日本語版については教育ライセンスの適用は不明です。

- プログラム本体
  http://www.geocities.jp/gimproject/gimp2.0.html
- マニュアル
  http://www.geocities.jp/gimpfile/gum_jp/index.html

GIMP は、市販のソフトウェアと比較して見劣りしない機能を備えています。多少使いづらいところもあるようですが、無償でここまでのことができるのは特筆すべきことです。

次に、ソフトウェアを PC にインストールする必要のない、Web アプリケーションを 2 つ紹介します。どちらも作成した XHTML が文法に沿って正しく作成されている（妥当）かどうかを確認してくれるものです。

- W3C Markup Validation Service
  http://validator.w3.org/
- Another HTML-lint gateway
  http://openlab.ring.gr.jp/k16/htmllint/htmllint.html

最初のものは、XHTML の規格を定めている W3C による検証サービスです。後者は日本語で間違っている部分とその理由を指摘してくれ、採点もしてくれます。自分で XHTML を作成したら、これらのサービスで妥当性を確認してみましょう。

8.2.4　この節のまとめ

この節では、Web パブリッシングに便利なツールについて解説しました。Web パブリッシングに最低限必要なツールはテキストエディターと FTP ソフトウェアですが、どちらも Windows や MacOS には付属しています。ただし、より便利で簡単に利用できるものが無償で提供されていますので、それらを使いましょう。

最低限必要というわけではありませんが、画像処理プログラムがあると表現の幅が広がります。GIMP というプログラムを無償で利用することができます。また、これから作成する XHTML の妥当性検証に利用できる Web アプリケーションを積極的に利用して、正しい XHTML の作成を心がけましょう。

8.3　最初の XHTML

8.3.1　マークアップとタグ

ここでは、まず最小限の XHTML を作成してみます。正確には XHTML として妥当（valid）ではありませんが、とりあえず Web ブラウザで表示して見ることのできるコンパクトな XHTML を作って、これを徐々に妥当なものへと成長させていくことにしましょう。

前述のように、XHTML は文書構造を記述するための言語です。ここではまず、「見出し」と「段落」からなる文書を作ってみます。テキストエディターを開いて、次のように入力してみてください。ここで、記号とアルファベット類は、すべて半角[15]で入力します。

[15] ここでは 1 バイトの英数文字を半角と呼んでいます。この「半角」とか「1 バイトの英数文字」という呼称はいずれも正確ではないのですが、このように呼んでおきます。日本語キーボードでは、キーボード左上にある「半角/ 全角」というキーを押すと半角と全角が切り替わります。また、ここでは本文中のアルファベット「XHTML」は半角で入力する必要はありません。

```
<html>
<h1>
はじめての XHTML
</h1>

<p>
これははじめての XHTML です。
</p>
</html>
```

ここで、上の内容のファイルを保存しましょう。保存先はファイルシステムのどこであっても構いませんが、見つけやすい場所 (例えばデスクトップなど) にしておきましょう。ファイル名は、すべて半角で「index.html」としてください。ファイルを保存したら、エディターは終了してください。

次に、作成したファイルをダブルクリックして開いてみましょう。ダブルクリックしてブラウザが開かない場合は、ファイル名、特に拡張子を確認してみてください。また、Web ブラウザを起動し、「ファイル」メニューの「開く」を選択してファイルを開いても構いません。Web ブラウザには図 8.4 のように表示されているはずです。

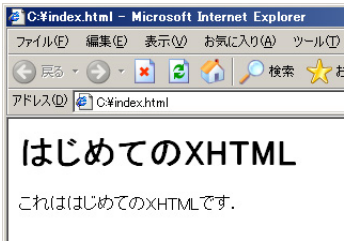

図 8.4: はじめての XHTML をブラウザで表示したところ

これで見出しと本文を含むページを作成することができましたので、極端な話をすれば皆さんが情報発信をしたいと考えたときに利用しなければならない最低限のことは勉強できたことになります。

図 8.4 のように表示されていない場合は、テキストエディターを起動し、ファイルメニューから「開く」を選択して先ほど保存した index.html というファイルを開き、先ほど入力した内容が間違っていないかどうか、もう一度よく確認してみてください。

では、入力した内容を検討しましょう。XHTML では、上の例で取り上げた「見出し」や「段落」のように利用することのできる文書の要素はあらかじめ定義されています。このように、あらかじめ決められた要素に対応する「タグ」と呼ばれる記号で実際の文書の各要素を「マークアップ」していくことで XHTML を作成していきます。

ここでタグとは「 」のように「< 」および「> 」で囲まれています。「 」のようにスラッシュが入ると要素の終了を意味し、スラッシュがなければ要素の開始を意味します。上の例を見てみると、「これははじめての XHTML です。」という行が開始と終了のタグによって囲まれています (図 8.5 参照)。この「 」というタグは「段落 (paragraph)」を表すタグです。つまり、このタグで囲まれた部分は、1 つの段落であるという意味になります。一方で<h1> は「レベル 1（最も高次）の見出し」という意味です。画面上では大きく表示されていますが、これは「文字を大きくしろ」と指定しているわけではなく、レベルの高い見出しの文字であるため大きく表示されていることに注意してください。この点は後に詳述します。

このように、タグなどで「意味付け」することを「マークアップ」と呼んでいます。XHTML は

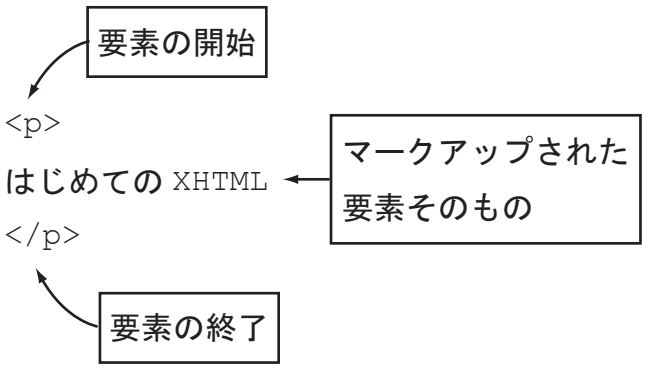

図 8.5: マークアップ

Extensible Hypertext Markup Language の略ですが、「Markup Language」と呼ばれるのはこのためです。ここで、XHTML のことを「タグで囲む言語」とは覚えないでください。タグは非常に重要ですが、タグそのものが重要なのではなくそれによって要素をマークアップしているのだ、ということが重要なのです。

また、先の例では文書全体が<html> と/<html> でマークアップされていることに注意してください。この文書全体が html 要素であるということです（この要素の正式な記述方法は後に詳述します)。

さて、前述のように XHTML で利用することのできる「文書要素」はあらかじめ定義されています。「定義されていない文書要素は使えないのですか？」という疑問を持つ方もいるかもしれません。XHTML の名前が示すようにある意味では拡張可能です。これに関する解説は後述（158 ページ）しますが、拡張可能だからといって自分で勝手に文書要素を定義してタグが作れるわけではありません。

8.3.2　タグと体裁、文書構造の関係

Web ブラウザは Web サーバーからこのようにタグでマークアップされたデータを受け取り、これを解釈し、「レンダリング」します。レンダリングとは、文書をコンピューターのモニタ上に表示するにあたって文書構造に合わせた体裁を付けるということです。

文書構造と体裁は無関係とは言い切れません。例えば、意味上のまとまりとして章は節を含み、節は項を含み、項には段落が含まれます（図 8.6 参照)。章や節、項にはそれぞれ見出しが付き、段落には本文が入りますが、そこで利用される文字の大きさは、もし可能であれば 章 > 節 > 項 > 段落という関係にするべきです。そのようにすることで、我々は文章の包含関係を文字の大きさで直感的に把握することができるのです。体裁と文書構造は本質的には関係ないとしても、体裁上のメリハリを付けることは「人に優しいデザイン」であると言えます[16]。本書もそのような体裁付けが行われていますので、参考にすると良いでしょう。

そこで、多くの Web ブラウザでは XHTML をサーバーから受信するとこれを解釈し、レンダリングして見出しを大きい文字として表示するなどします[17]。また、後述するように XHTML から呼び出される写真や絵などを配置して表示するなどします。

[16]ただし、決して良いことではありませんが、このような文書構造と体裁の関係を無視して自分の強調したい部分の文字を大きくするといったことも可能で、またよく行われています。

[17]レンダリングは、完全に Web ブラウザの機能に依存します。Web ブラウザの中には、同じ大きさのテキストしか表示できないというものも存在します。画像その他も当然表示することができません。Lynx や w3m などが代表です。このようなブラウザを利用する意義がどのあたりにあるのかという考え方は人それぞれですので調べてみると良いでしょう。

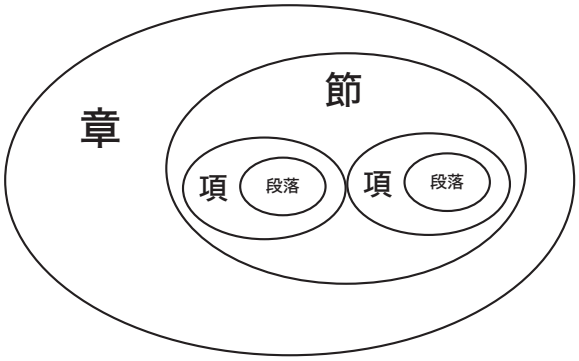

図 8.6: 章、節、項、段落の包含関係

このように、XHTML がタグという形での解釈やレンダリングについてクライアント（Web ブラウザ）に任せているのは、1 つには多様なクライアントが Web に参加できるように配慮している、ということなのです。昨今は、携帯電話から Web サイトを閲覧することができるようになっています。PC と比較して、画面の大きさ、色数、CPU、メモリ、文字入力方法など様々な面で制限のある Web ブラウザは数多くあります。点字ディスプレイや音声朗読ソフトウェアで Web を参照している人にも的確に情報を伝えることができるかもしれません。

また、例えば XHTML には「見出し」を定義するタグがありますが、このような見出しとして定義された要素だけ抜き出せば、目次を作成することができそうです。しかも、人間が目で見ながら拾い集めなくとも、コンピューターが自動処理するのが得意そうな作業です。コンピューターで情報を作成することの大きな利点の 1 つが「再利用性」にあります。これを世界的な規模に拡大するのが Web の目標であると言っても良いのですが、これもタグを利用して要素が表現されているために実現することができる特徴なのです。

XHTML は、基本的にこのような文書の内容そのものと文書構造から成り立っており、これらはすべてテキストとして表現されています。後述するように、このようなテキストという形式で表現されるファイルを作成・編集するには前述のテキストエディターを利用します。

8.4 この章で解説する文書要素

見通しを良くするために、本書で取り上げる XHTML 1.0 の要素一覧とそれを表現するためのタグの一覧を、それぞれ表 8.2 および表 8.3 にまとめておきます。中には Web やコンピューターに特有のものもありますが、要素の多くは紙で行われている組版で長い間利用されているものであり、それらの語彙に馴染みがなくても、実際に組版されたものを一見すれば馴染みの深いものばかりです。

XHTML 1.0 で利用することのできる文書要素は、HTML 4.01 のものとほぼ同じです。XHTML 1.0 では多数の要素が定義されているのですが、そのすべてを学習する必要はありません (また、ここで紹介し切れるものでもありません)。ここでは、XHTML Basic [18] という仕様に基づいて要素を分類した上で、そのうち重要であると思われるものに限って解説します。

XHTML では要素の他に属性（attribute）を指定することが可能で、また属性を伴って初めて意味を持つ要素もあります。どの要素に対してどの属性を指定することができるのか、ということもやはり決まっています。以下で要素を紹介する際に、必要に応じて属性も紹介していきます。

[18]携帯電話や携帯情報端末、カーナビゲーションシステム、各種情報家電などのように何かしらの Web ページを表示する機器は急速に増加しています。XHTML Basic はこのようにプロセッサやメモリ、表示機能が限定されている機器向けの、最低限の XHTML 規格です。このような XHTML を学習する意義は、単に携帯電話向けの Web ページを制作できるようになるというだけではありません。これは、様々な機器を念頭に置いたベースとなる規格ですので、XHTML 学習の参考にするのに最適です。

表 8.2: 本書で取り上げる文書要素一覧

文書構造		html、ヘッダ、表題、本文
テキスト	ブロック	見出し (レベル 16)、段落、引用、整形済みテキスト、問い合わせ先
	インライン	改行、他文書の参照、引用、強調、より強い強調、省略語
ハイパーテキスト		アンカー
箇条書き		並列、順序付き、定義型、項目
表		表題、表、列・行見出し、行、表データ
画像		画像
メタ情報		メタ情報

表 8.3: 文書要素一覧 (表 8.2) に対応したタグ

文書構造		html, head, title, body
テキスト	ブロック	h1, h2, h3, h4, h5, h6, p, blockquote, pre, address
	インライン	br, cite, q, em, strong, dfn, abbr, acronym
ハイパーテキスト		a
箇条書き		ul, ol, dl, dt, dd, li
表		caption, table, th, tr, td
画像		img
メタ情報		meta

8.5　テキストと画像の要素

8.5.1　ブロック要素

段落と見出し

テキストは、最も基本的な文書要素です。テキスト要素は、大きく分けて「ブロック要素」と「インライン要素」に分類することができます。これらがどのようなものであるかということについては「8.10 文書構造とメタ情報の要素」で後述します。ここでは、「ブロック要素は文書の構成要素」「インライン要素はブロック要素内の一部を修飾する要素」であると何となく理解しておいてください。表 8.3 における要素のうち、箇条書きや表はブロック要素で、画像やハイパーテキストはインライン要素です。

さて、私たちの作成した XHTML は、現在次のようになっています。これを拡張していきますが、まずはこの意味から解説しましょう。

```
<html>
<h1>はじめての XHTML</h1>

<p>これははじめての XHTML です.</p>
</html>
```

まず<html>ですが、この文書が HTML 文書であることを意味しています。文書全体が HTML 要素であることが分かります。本当はこれでは不十分なのですが、とりあえず説明が複雑になるのを

避けるために、詳細については後述することとして、ここでは単に頭とおしりにそれぞれ<html>と</html>を入れておいてください。

次に、<h1>の h は heading の h であり、続く 1 とあわせて「レベル 1 の見出し」を意味します。Web ブラウザで表示してみると理解できることですが、これは Web ブラウザの中で文字を最も大きく表示するタグです[19]。大きい文字は、文書構造上より大きな意味上のまとまりを表現するときに利用されるものです。表示される文字の大きさとしては、h1 >h2 >h3 という関係が成立します。レベル 4 から 6 の見出しについても同じです。

保存してある「index.html」というファイルをテキストエディターで開き、次のように修正します。

```
<h1>はじめての XHTML</h1>
<h2>レベル 2 の見出し</h2>
<h3>レベル 3 の見出し</h3>

<p>これははじめての XHTML です。</p>

<p>これは 2 つ目の段落です。</p>
```

もう 1 つ、<p >というタグがあります。これは段落を表す要素です。paragraph の p と覚えてください。

修正したら「index.html」をダブルクリックして開くか、Web ブラウザを起動してから「ファイル」→「開く」と選択してこの「index.html」を指定して開き、Web ブラウザ上に表示してみてください。

確かに上記のように編集したのに Web ブラウザの表示が書き換わっていないという場合は「更新」ボタンを押してください (図 8.7 参照)。

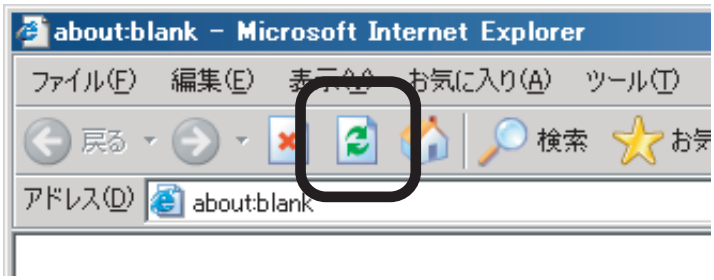

図 8.7: 更新ボタン

更新ボタンは、今回に限らず Web ページの最新状態がブラウザ上に反映されていないと感じたら、まず押してみてください。一回押してだめでも、シフトキーやコントロールキーを押しながら更新ボタンを押すと更新されることがあります[20]。

ここで、タグによるマークアップの際に注意しなければならない基本的なことを 2 つ説明しておきます。

1. タグにはすべて小文字のアルファベットを利用する
2. 要素が交差してはならない
3. 空白文字の扱いに注意が必要

[19] もっと文字を大きく表示する手段が無いこともありません。この方法については後述しますが、推奨されることではありません。

[20] これを「スーパーリロード」と呼びます。これは、ブラウザにキャッシュを無視させるための機能です。「キャッシュ」については後述します。

第 1 点目ですが、タグはすべてアルファベットの小文字を利用します。従来は大文字も小文字も、それらが混在することすら許容されていましたが、XHTML では小文字で統一されています。

第 2 点目ですが、次のようにしてはならない、ということです (してはならない例なので、入力する必要はありません)。

```
<h1>これは見出しです

<p>これは段落です。</p>

</h1>
```

どのあたりが問題であるか分かるでしょうか。見出しが始まり、その見出しが終わる前に 2 つ目の段落が開始されています。次に段落の終了（</p>）がありますので、その段落が終了したということになります。そして、次に見出しが終了しています。見出しの中に段落があるということでしょうか？

これは要素が「交差」している例で、XHTML では禁止されています。このような交差が許されないのは、ここで解説しているブロック要素だけでなく、後述するインライン要素についても同じです。

第 3 点目ですが、XHTML においてスペース、タブ、改行などの文字は、すべて空白文字 (white space characters) として扱われ、基本的に単語の境界を表すものとして扱われます。また、2 つ以上の連続する空白文字は、1 つにまとめられます。例えば、作成中の XHTML に次のように書き加えてみてください。

```
<p>これは 4 つ目の段落です。</p>

<p>
これは 5 つ目の段落です。
</p>

<p>
これは 6 つ目の
段落です。
</p>

<p>
これは
        7 つ目の
                段落です。
</p>
```

これらはいずれも表示上はほとんど同じ結果となります。4 つ目の段落と 5 つ目の段落の違いは、タグの位置です。これは何の違いももたらしません。6 つ目の段落および 7 つ目の段落はどうでしょうか。改行されているところ、つまり 6 つ目の段落では「の」と「段」の間、7 つ目の段落では「は」と「7」そして「の」と「段」の間でブラウザの表示上、少し間隔が開いている気がするかもしれません。これは、欧米系の言語の処理を Web ブラウザが引きずっているためですが、さほど気になるレベルのものではないはずです。

これをまとめると、タグは要素を正しくマークアップすれば良く、改行を含む空白文字は 2 つ以上あっても無視されるだけ、ということになります。段落の途中で改行してもブラウザに表示される際に改行されることはありません。逆に言えば、要素の定義上改行しなければならない場合は自動的に改行されます。例えば、段落が終わった場合や、見出しが終わった場合は、その時点で改行されます。段落が長くて複数行にわたる場合も改行されますが、この場合の改行幅 (ピッチ) は段落と段落の間の幅より短くなることが多いようです。これは、より長め段落を自分で書いて確認してみると良いでしょう。

XHTML も含めて、コンピューター言語を書いていくことを「コーディング」といいます。XHTML のコーディングの際には、テキストエディターでの編集上、見やすい位置で改行を入れて構わないということになります。但し、次のようにタグの途中で改行することはできません。

```
<h
1>
悪い例：タグの途中で改行されている
</h
1>
```

このように、あまり空白文字について考える必要がないというのは、ある意味で親切なことではあるのですが、時には自分が入力したエディター上で見えていたそのままでブラウザ上に表示してもらいたいときもあります。このような場合、プリフォーマット (preformatted、整形済み) という要素を利用することができます。この要素は、<pre ></pre >で表現します。

整形済み

次の例のように、7 つ目の段落の後ろに付け加えてみてください。

```
<p>
これは
        7 つ目の
                段落です。
</p>

<pre>
これは
        プリフォーマットの
                文章です。
</pre>
```

違いを比較してみてください。改行も空白も、そのまま表示されます。

引用

次に、引用したテキストを表示するための要素です。次の例を、作成している XHTML に付け加えてみてください。

```
<blockquote>
<p>
これは引用しているところ。ブロック要素なのでまとまった量の引用を表示する
 のに便利である。通常、少しインデントされて表示され、引用した文章である
 ことが分かりやすいよう表示される。
</p>
</blockquote>
```

正確にこの通り入力する必要はありませんが、2 行以上に渡るような分量を入れてやると分かりやすいものと思います。blockquote 要素は、ある程度まとまった文章を引用するための要素です。上の例で blockquote 要素は別のブロック要素を内包していることが分かります。ここでは段落が入っていますが、複数の段落が入ることも、見出しが入ることもあるでしょう。

問い合わせ先

ここで最後に紹介するブロック要素は問い合わせ先です。住所やメールアドレスなどを表示するための要素です。

```
<p>
このページに関しては、次の住所まではがきか封書でご連絡ください。
</p>

<address>
〒 169-8050 新宿区戸塚町 1-104 早稲田大学
</address>
```

上の例は少々あいまいな書き方をしていますが、この要素はもともとその Web ページの著者に関する情報を提示するのに利用するものです。郵便番号住所、電話番号、FAX 番号、メールアドレスなど、その文書に関する連絡先を表示します。

なお、間違っても自分の住所を Web 上にさらけ出すべきではありません。電子メールアドレスは提示すべきですが[21]、公開された Web ページから自動的にメールアドレスを収集し、これを迷惑メールの送信先に利用するという例が後を絶たないため、この部分だけを画像にするといった工夫をしていることも多く見られるようです。

ここで紹介するテキストのブロック要素は以上ですが、これで大半の文書のほとんどの部分を表現できることが理解できると思います。次に述べるインライン要素を利用しないといくつか困ることはあるかもしれませんが、ここで学習した要素だけを利用するという簡素なページであっても、中身さえしっかりしていれば、集客のできる Web ページを作ることも不可能ではありません。

8.5.2　インライン要素

ブロック要素が文章の基礎的要素（building block）であるのに対して、そのブロック要素内のある部分を修飾したり、追加的な意味を与えたりするのがインライン要素です。ここで取り上げる

[21] 早稲田大学の【Web ドメインにおける Web コンテンツ作成に関する要領】には、「ドキュメントの掲示責任者とその連絡先を明示しなければならない」とあります。一般的に著作権法により匿名で発言する権利が認められていますが、責任のある言論を行うために連絡先を提示すべきかどうかということを考えると、言論の府である大学では提示しなければならないと考えられるからです。

インライン要素に対応するタグは、br、cite、q、em、strong、dfn、abbr、acronym です。以下それぞれについて解説します。

改行

br は line break であり、つまり改行です。前述のように、XHTML では改行は改行として処理されず、単語の区切りを示すに過ぎません。したがって、もし段落の中で自由に改行をしたい場合は、別途指定をする必要があるというわけです。

ところで、段落内で強制的に改行をしたい場合とはどのような場合でしょうか？詩などもその 1 つでしょう。あるいは、前述の address 要素で住所を記載する場合、改行をしたいと思うかもしれません。

br 要素は、次のように利用します。

```
<address>
〒 169-8050<br />
新宿区戸塚町 1-104<br />
早稲田大学
</address>
```

br 要素については、今まで紹介したタグとはスラッシュ (/) の使い方が異なっています。今までは、スラッシュがなければ要素の開始、あれば要素の終了を意味し、そのタグで文書の該当する要素を括っていました。しかし、br 要素にはそのような括るべき要素がありません。それ 1 つで意味を持っているからです。このように、タグそのもので要素となるようなものを空要素といいます。空要素はタグを閉じることができませんので、それ自体で開始と終了を併せ持つような表記をするのです。そこで、<br / >と表記します。

なお、上記の例では最初の二行に br 要素がついていますが、最後の行には br 要素を付けていません。これは、address 要素の終了とともに改行が予定されるため、br 要素による改行は必要が無いからです。

文献の参照と引用

次に、cite 要素と q 要素です。cite は文献への参照を表現するのに利用します。

```
<blockquote>
<p>国境の長いトンネルを抜けると雪国だった。</p>

<cite>川端 康成,『雪国』, 新潮文庫, 1937 年 6 月.</cite>
</blockquote>
```

上記の例では blockquote 要素内に別のブロック要素を含めていますが、段落などなしで、いきなり文字を書いてしまっても構いません。

一方、q 要素は文中で引用したい場合に利用します。

```
<p>
川端康成の『雪国』は、<q>国境の長いトンネルを抜けると雪国だった。</q>と
いう有名な書き出しで知られている。
</p>
```

ただし、q 要素については p 要素中のその他の文章と区別できるような体裁付けが行われないのが普通です。一部の Web ブラウザではダブルクォーテーションマーク (”) で囲われるようですが、日本語でそのような体裁をつけるのが良いかというと決してよくありません。引用についてはすでに勉強したように、引用した部分と自分で書いている本文とを明確に区別することができるというのが大切ですが、マークアップとしてはきれいに区別できていても、体裁上分からないと困ったことになる可能性もあります。「」で括るなどすると分かりやすくなりますが、一部の Web ブラウザで表示されたとき、少々おかしなことになってしまいます。

これは、マークアップ言語と Web ブラウザの機能実装がうまく対応していない 1 つの例です。このような例は他にもあり、XHTML による文書表現が必ずしも意図通り行われないこともある、ということを念頭においておくと良いでしょう。

強調

次は、強調です。em 要素と strong 要素です。em は emphasize の em と覚えましょう。ここでの問題は、何で 2 つもあるのか、ということです。これは、強弱の関係があるからです。em より strong の方が強いのです。これらは、太字体 (bold) として表示されることが多いようです。同じように太字体で表示されますが、意味の違う要素として dfn があります。これは、definition の略で定義語を表します。

これら 3 つの要素の例を挙げておきます。

```
<p>
これは<em>強調</em>ですが、こちらは<strong>より強い強調</strong>です。
定義語は<dfn>dfn 要素</dfn>で表現しますが、初出の重要な語をマークアップす
るのに利用されます。
</p>
```

これらはどれも表示上太字になりますが、太字にするために利用してはいけません。あくまでも強調するために利用するのです。

省略語と頭文字

次に、abbr と acronym ですが、これらはそれぞれ abbreviation(省略語) と acronym(頭字語) を表現するためのものです。abbr は短縮された言葉です。例えば、by the way を BTW とするようなものです。頭文字語は、国際連合 (United Nations) を UN とするようなものです。以下に例をあげておきます。

```
<p>
今日、授業で<abbr title="メディアネットワークセンター">メネセン</abbr>に行っ
たら、<acronym title="Waseda university InterNet Domain">WIND</acronym>
の内規を読むという宿題を出された。
</p>
```

abbr や acronym 要素に対応した Web ブラウザで上記の XHTML を表示すると、「メネセン」および「WIND」という文字それぞれにアンダーラインが引かれ、そこにマウスポインタを合わせると title 属性が表示されますので、試してみると良いでしょう。

8.5.3 要素の属性

これらの要素には、「属性」という今までの要素には無かったものが利用されています。属性とは要素の性質や機能について細かく指定するものであり、要素の開始タグ内に記述されます。複数の属性を持つことができる場合もあります。一般的に属性は、次のように表記します。

<要素 属性１="属性１の内容" 属性２="属性２の内容">

要素と属性 1 の間、属性 1 と属性 2 の間などは空白 (スペース) で区切ります。属性の中身を属性値と呼び、イコールの右辺で指定しますが、これは必ずダブルクォーテーションマーク (") で括ることになっています。ここではいずれも title という属性を指定しています。他にも様々な属性がありますが、どの属性を適用できるかというのは要素によって異なります。

例えば、前出の blockquote 要素には cite 属性 (cite 要素とは異なる) というものを持たせることができます。

<blockquote cite="http://www.waseda.jp/mnc/" title="MNC **ホームページ**">

このように、属性は XML において重要な役割を果たします。

8.6 ハイパーテキストの要素

ハイパーテキストの要素は 1 つしかありません。a 要素です。アンカーの a と覚えましょう。1 つしかないといっても XHTML はハイパーテキストを記述するための言語であり、これは XHTML の特徴的な要素であるといえます。

そもそも、ハイパーテキストとはどのようなものかということは説明していませんでしたが、一言で言えば「関連性のある文書から文書へと辿っていくことができる文書」です。口で説明するより実際に操作をすれば簡単に理解することができます。典型的には Web ブラウザで下線が引かれている文字をクリックをすると、別のページが Web ブラウザに表示されるというシステムです。

8.6.1 リンク元のアンカー：ハイパーリファレンス

まずは例をあげるところから始めましょう。

```
<p>
早慶戦は<a href="http://www.waseda.jp/">早稲田大学</a>と
<a href="http://www.keio.ac.jp/">慶応大学</a>の伝統です.
</p>
```

次に、タグの中身を検討しましょう。href 属性が指定されていますが、これはハイパーリファレンスということで、参照先を指定する属性です。前出の例では、「早稲田大学」という文字をクリックすれば早稲田大学のトップページを、「慶応大学」という文字をクリックすれば慶応大学のトップページを参照することができます。href 属性の属性値は URL となっています。

もう 1 つ、accesskey 属性を追加してみましょう。例えば、上記の例を次のように拡張してみます。

```
<p>
早慶戦は<a href="http://www.waseda.jp/" accesskey="1">早稲田大学</a>と
<a href="http://www.keio.ac.jp/" accesskey="2">慶応大学</a>の伝統です。
</p>
```

この属性は、アクセスキーと呼ばれるショートカットキーを追加するものです。上の例では、ブラウザに表示させた際に Alt キーを押しながらキーボードの 1 というキーを押すと、そのリンクが選択された状態になります。ただし、即時にリンクを辿った状態になるわけではなく、このようにしてからエンターキーを押すことでリンクを辿ることが可能です。ただし、この挙動は Web ブラウザに依存します。携帯電話に搭載されているブラウザの場合、1 というキーを押せばすぐにこのリンクが辿られることが多いようです。

8.6.2　リンク先のアンカー：フラグメント

さて、これまのアンカーはリンク元を作成してきました。つまり、そこから別の文書へたどるというリンクです。これに対して、リンク先を作ることもできます。

```
<h1>
<a name="midashi1">これは最初の見出しです</a>
</h1>

...
...
...

<h1>
<a name="midashi2">これは 2 番目の見出しです</a>
</h1>
```

このようにすることで、それぞれの見出しに「midashi1」および「midashi2」という名前を付けることができます、名前を付けたからには、その名前を利用してこれらの見出しにアクセスすることができます。このファイルの名前が index.html であると仮定すれば、例えば、以下のような URL でアクセスができるようになります。

```
http://www.example.ac.jp/index.html#midashi1
```

つまり、文書の一部を直接指定できるようになるということです。このような文書の一部分のことをフラグメントと呼びます。

また、これまで a 要素はすべて段落の中の文字に適用してきました。実際には、見出しの文字にも、また後述する画像にもアンカーを設定することができます。これらも、例だけ示しておきます[22]。

```
<h1>
<a href="http://www.waseda.jp/">早稲田大学</a>
</h1>

<p>
<a href="http://www.waseda.jp/jp/okuma/"><img src="okuma.jpg" /></a>
</p>
```

[22] ここでは手元に okuma.jpg というファイルがあることを前提にしています。そのようなファイルがない場合は、ブラウザ上では中に赤い×の書いてある四角が表示されるはずですが、その場合でもリンクを辿ることは可能です。

ところで、a 要素はインライン要素です。したがってブロック要素である p 要素の一部、つまり段落の一部を修飾するような形で適用されていることに注意しましょう。ですから、次のような例は間違いです（インライン要素がブロック要素を内包しているため)。

```
<a href="http://www.waseda.ac.jp/">
<p>
早稲田大学
</p>
</a>
```

8.7 箇条書きの要素

箇条書き (リスト) は情報を分かりやすくまとめる手段として有効です。順序が無く並列的に並べる場合や、順序を付けて並べる場合など色々考えられますが、XHTML でも箇条書きを作成することができます。

8.7.1 並列

最初に、順序が関係ない並列的な箇条書きを作成してみましょう。ul (unordered list) 要素を利用します。

```
<p>
東京六大学は次の大学から構成されています。
</p>

<ul>
  <li>早稲田大学</li>
  <li>慶応大学</li>
  <li>東京大学</li>
  <li>立教大学</li>
  <li>明治大学</li>
  <li>法政大学</li>
</ul>
```

上の例では、新たに li(list item) 要素も出ています。少々タグが多くて煩雑に見えるかもしれません。上の例では、li 要素を少し字下げすることで包含関係も含めて XHTML のコードを見やすくする工夫をしています。

8.7.2 序列付き

次に序列付きの箇条書きです。これは ol(ordered list) 要素で作成します。

```
<ol>
  <li>早稲田大学</li>
  <li>慶応大学</li>
</ol>
```

というタグを と書き換えれば良いだけですが、結果はずいぶん違います。この例を書き加えた XHTML を表示させると分かりますが、ol 要素では上から順に番号が自動的に振られます。例えば上の例で早稲田と慶応の間に東京大学を追加すれば、自動的に番号がずれて表示されます。このように自動的に番号を振る意義は、単に一時の楽をしたいというだけのことではありません。このような番号の振りなおしの手間を省き、番号の付け間違えを無くすという効果もあるのです[23]。

8.7.3　入れ子の箇条書き

li 要素には内容としてテキストをそのまま含むことも可能ですし、段落をなどのブロック要素を含むこともできます。更には、別の箇条書きを含むことも可能で、これを「入れ子」と言います。以下に入れ子の例を示します。

```
<ul>
  <li>早稲田大学
    <ul>
      <li>西早稲田</li>
      <li>戸山</li>
      <li>大久保</li>
    </ul>
  </li>
  <li>慶応大学
    <ul>
      <li>三田</li>
      <li>日吉</li>
      <li>藤沢</li>
    </ul>
  </li>
</ul>
```

この例で注意しなければならないのは、
の対応関係です。最初の「
早稲田大学」の li 要素は、どこで閉じられているでしょうか。リスト中に含まれるリストは、あくまでも上位のリストの 1 つの項目 (li 要素) の中に包含されていることに注意してください。

次に、定義型の箇条書きを作成することができます。用語集などを作成するときに便利です。dl(definition list) 要素、dt(definition term) 要素、dd (definition definition) 要素を利用します。

[23] ことコンピューターに関しては、コンピューターに任せられるところは任せて楽をするのは美徳です。最小限の手間で最大限の効果を得るような工夫をすることは、賞賛の対象となります。

```
<dl>
  <dt>早稲田大学</dt>
    <dd>東京都新宿区西早稲田 1-6-1</dd>
  <dt>慶応大学</dt>
    <dd>東京都港区三田 2-15-45</dd>
</dl>
```

8.8 表の要素

表もまた、情報を整理して提示するのに便利な方法の 1 つです。表 8.8 に表で良く利用される要素を示します。

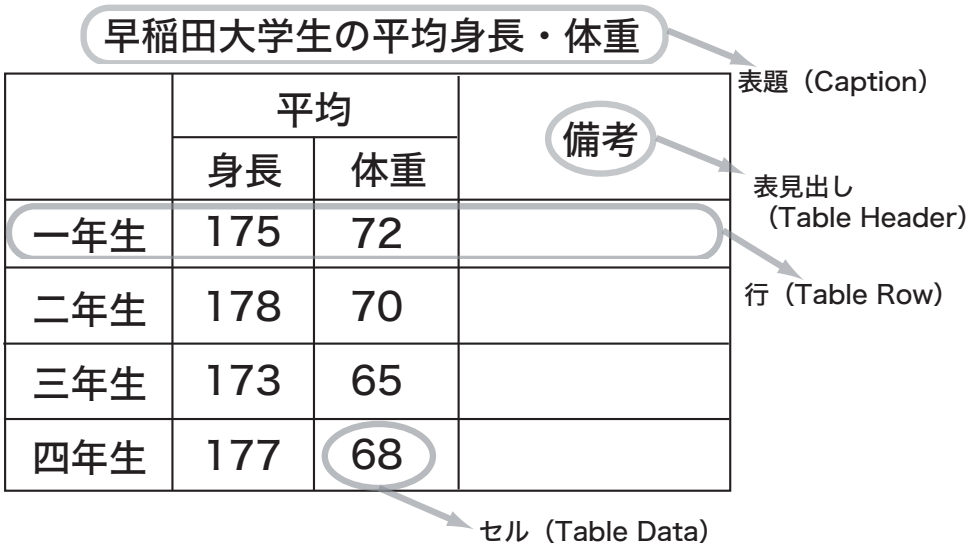

図 8.8: 表の要素

8.8.1 2 × 2 の表

まずは最も簡単な例から見ることにしましょう。枠付きで、2 行 2 列の表です。

```
<table border="1">
  <tr><td>あ</td><td>い</td></tr>
  <tr><td>う</td><td>え</td></tr>
</table>
```

上の例を見てみましょう。少々入り組んでいますが、ゆっくり眺めれば法則が見えてきます。まず、全体で 1 つの table 要素となっています。table 要素には border 属性があり、1 が属性値として指定されています。この属性は枠の太さを指定するものであり、2 や 3 を指定して 1 の場合と比較してどうであるか確認してみてください。また、border 属性を指定しない場合は枠が表示されません[24]。

[24]枠を表示しないこともできることを利用して、表組みは Web パブリッシングにおいて体裁を整えるために利用されてきました。これは必ずしも表の正しい使い方ではありません。後述する CSS(Cascading Style Sheets) を利用すれば、体裁を確保

次に、2 つの行があることが分かります。それぞれの行は 1 つの tr(table row) 要素からなっています。この tr 要素が表の行となります。tr 要素の中には、さらに 2 つの td(table data) 要素があります。これらの td 要素の中に具体的なデータ (ここではあ、い、うおよびえ) が入ります。

8.8.2　行と列の連結

表の行を増やしたければ tr 要素を増やせば、また列を増やしたければ tr 要素の中の td 要素を増やせばよいのです。ただし、すべての tr 要素中で td 要素の数、つまり列の数を合わせる必要があります。ある行では 2 列しかなく別の行では 3 列あるということをすると、表の形が崩れます。この点を確認するために、連結について見ておきましょう。表を 3 行 2 列にしますが、1 行目のセルは連結されており、実際には 1 列しかないというものです。

表の例 1：セルの連結 (列)

```
<table border="1">
  <tr><td colspan="2">五十音</td></tr>
  <tr><td>あ</td><td>い</td></tr>
  <tr><td>う</td><td>え</td></tr>
</table>
```

五十音	
あ	い
う	え

図 8.9: 表示結果

例 1 の HTML の 2 行目を見てください。td 要素は 1 つしかありませんが、その代わりに colspan 属性があるのが 3 行目以降との違いです。colspan 属性はその名の通り列の範囲を指定するものであり、1 列ではあっても 2 列分の幅を持って連結されて表示されます。

表の例 2：セル連結 (行)

```
<table border="1">
  <tr><td rowspan="2">あいうえお</td><td>あ</td><td>い</td></tr>
  <tr><td>う</td><td>え</td></tr>
</table>
```

あいうえお	あ	い
	う	え

図 8.10: 表示結果

次に行を連結することを考えてみましょう。例 2 を見てください。1 行目の一番左の列が 2 行に渡っているというのが、行方向のセル連結です。1 行目が 3 列あるのに対して、2 行目には 2 列分しかデータが無いことに注意しましょう。列であれ行であれ、連結されたらその分のセルが減るということです。

最後に、表の標題と行および列の見出しを設定しましょう。例 3 を参照してください。前出の例では「五十音」や「あいうえお」が表内の見出しになりますが、これらはその他のセルのデータとは区別されるべきです。また、表に適切な標題を付ければ読者の参考になります。

表の例 3：表の標題・表内見出し・セル連結

```
<table border="1">
  <caption>五十音</caption>
  <tr><td></td><th colspan="2">五十音</th></tr>
  <tr><th rowspan="2">あいうえお</th><td>あ</td><td>い</td></tr>
  <tr><td>う</td><td>え</td></tr>
</table>
```

五十音

	五十音	
あいうえお	あ	い
	う	え

図 8.11: 表示結果

するために表組みを利用する必要などはありません。しかし Web ブラウザによってこの CSS のサポート状況が異なり、ブラウザによって表示がされなかったり崩れたりします。そのため、デザイン性を重視する場合はやむを得ないという意見もあります。

行の連結と列の連結が組み合わされているので、少々複雑な例となっています。くじけずに、1 行ずつ解析してみてください。

上の例では新たに caption 要素が利用されています。caption 要素は table 要素の直下であればどこに書いても構いませんが、<table >の直後か </table >の直前に書くのが普通です。表内の見出しについては td 要素の代わりに th 要素が利用されているのが分かります。Web ブラウザにより異なりますが、th 要素は td 要素と区別するために太字で表示されることが多いようです。

8.9 画像

8.9.1 画像の取り込み

画像はインライン要素ですが、扱いは文字とほぼ同じであると考えると良いでしょう。文字と同じですから、段落や見出し、表などのブロック要素の中に出てくる必要があることに注意しましょう。

XHTML は、そのファイルの中に画像そのものを取り込むことはできません。そこで、画像は別のファイルとして用意しておき、XHTML からその画像を呼び出してレンダリング時にブラウザに表示されるように指定する、ということをします。一見ややこしいように思えますが、このようにすることで様々な形式のファイルを XHTML 文書の中に取り込むことを可能にしています。ここでは画像 (図や写真) のみを対象に解説しますが、動画や音楽なども同じような形で XHTML の中に取り込むことが可能です。

画像を表示するには、表示するべき画像そのものが必要です。図でもグラフでも、また絵や写真でも同じことですが、もし XHTML に取り込もうとしているその画像が自分の作成したものであれば問題ありません。しかし、もしその画像が自分以外の作である場合、著作権に関する正しい知識を持っていることが問われます。もし正しい知識を持っておらず、従ってこれを利用していいか悪いかということについて正しい判断を下すことができない場合は、利用してはいけません。著作権に関しては第 3 章を参照してください。

ただし、インターネット上には、広く利用してもらう目的で公開されている作品が数多くあります。画像についてもそのようなものがあり、「フリー素材」といったキーワードで Web ページを検索すると、自由に利用できる様々な画像を見つけることができるでしょう。ここでは、そのような活動の 1 つである、Open Clip Art Library(`http://www.openclipart.org/`) を紹介しておきます。

取り込むべき画像には様々な種類があるということはすでに紹介しました (132 ページの「8.1.2 ファイル形式」参照)。画像ファイルでよく利用されるのが GIF と JPEG、PNG です。GIF や PNG は図やグラフ、イラストなどが多く、JPEG は写真などによく利用されます。ここでは、次のファイルを利用することにします。

```
http://www.isc.tamu.edu/~lewing/linux/sit3-shine.7.gif
```

まずはこのファイルをダウンロードしましょう。Web ブラウザを起動して、上記の URL を「アドレス」欄に入力します。ブラウザにペンギンの絵が表示されるはずですので、「ファイル」→「名前を付けて保存」と選択します。ここでファイル保存のダイアログが表示されますので、ここでは「penguin.gif」というファイル名を自分で指定して保存しましょう。保存する場所は、現在作成している XHTML ファイルと同じ場所にしておくと良いでしょう。ここで利用したブラウザは閉じてしまってかまいません。

さて、画像は文字とほぼ同じ扱いであると述べましたが、画像はそれを挿入したい場所で次のように記述します。

```
<p>
<img src="penguin.gif" alt="ペンギン"/>
</p>
```

ここでは例示のために段落の内容として画像を挿入していますが、既存の段落や見出し、表の内容として画像を入れて構いません。また、img 要素は br 要素同様に空要素であるため、タグの末尾にスラッシュが入っていることに注意してください。

img 要素には src 属性と alt 属性という 2 つの必須属性があります。src は source の略で、表示されるべき画像ファイルの URL(URI) を指定します。XHTML ファイルと同じ場所に置いてある画像ファイルを指定する場合は、単にその画像ファイルのファイル名を記述するだけです。alt 属性は alternative の略で、画像などが表示できないテキストブラウザや音声ブラウザが画像の代替として表示するテキストを指定するものです。アクセシビリティの観点からも、alt 属性は必須とされています。

画像については、表示サイズの指定を行うことが可能です。

```
<p>
<img src="penguin.gif" width="100" height="100" alt="ペンギン"/>
</p>
```

ここでは、幅と高さがそれぞれ 100 ピクセル (画面上の 100 個の点) で表示されます。もとの画像ファイルの幅と高さの比率を変えてしまうような幅と高さを指定すると画像が歪むことになりますので、注意しましょう。元の画像ファイルのサイズは、Web ブラウザでその画像を表示させて画像を右クリックし、「プロパティ」を選択すると参照することができます。

なお、このようにしてできた XHTML については、公開する際には XHTML ファイルだけでなく画像ファイルも一緒に公開する必要があります。

8.9.2　Web パブリッシングにおける画像の著作権

画像については、特に著作権についての配慮が重要です。Web ブラウザに表示される画像ならほぼすべて簡単にファイルとして PC に保存することができますし、これを自分の Web ページに転用するのも簡単です。しかし、無許可の複製を行うのは明白な著作権の侵害行為であり、明示的に利用して良い旨の表示がない画像は利用してはいけません。また、許可を得たり著作権法上認められている方法で利用する場合でも、適切な著作権表示や出展の表示が必要です。

実際には、画像について著作権法上判断が難しい問題があるのが現状です。例として、次のようなケースを考えてみましょう。

img 要素の src 属性には URL が入ることになります。前出の例では XHTML 文書と同じ階層にある画像ファイルを前提としていましたが、正しい URL を利用していれば、特に画像ファイルが設置されている場所に、制約はありません。従って、例えば次のようにすると、Web 上に公開されている画像を簡単に自分の Web ページに取り込むことができます。

```
<p>
<img src="http://www.waseda.jp/img/waseda_logo.gif" alt="Waseda Logo"/>
</p>
```

この場合、画像ファイルをコピーして XHTML ファイルと同じ場所に転送しておくという必要はありません。厳密に考えると、Web ページの制作者は画像ファイルをコピーしてはいないのです。

コピーは、この Web ページを閲覧する人が利用しているブラウザによって行われ、そのコピーは Web ページの制作者を介することはありません。

では、このような利用方法なら画像をどんどん利用して良いのかというと、当然のことながら良くありません。

では、著作権法上の「引用」の定義を満たすような利用の仕方ならどうでしょうか。実のところ、文字の引用と比較して画像の引用についてはあまり判例が無く、これといった基準が存在していないのが現状です。しかし、もし自分が当事者になって上の例が他者の著作権を侵害するかどうかという判例を、貴重な費用と時間をかけてでも作ってやろうという社会的な使命を強く感じているのでなければ、このようなグレーゾーンの行為はやめておきましょう。

つまり、画像の引用は、その画像そのものの評論でない限りほぼ無理であると考えて構いません。ただのリンクならば著作権の侵害にはなりませんので、引用ではなくリンクで対応してください。

8.10 文書構造とメタ情報の要素

今までは単に文書の個別要素 (パーツ) だけを作成していましたが、実はこれだけでは妥当（valid）な XHTML とはなりません。XHTML はこれが XHTML 文書であることやその文書に関する情報、また文書全体が正しい構造を持ってはじめて妥当な XHTML 文書となります。

最初に文書構造から考えましょう。XHTML では文書の全体構造が次の 4 つの部分からなることを規定しています。

1. XML 宣言
2. XHTML のバージョン情報を記した行 (DOCTYPE 宣言)
3. その文書についての情報を記した行 (ヘッダ)
4. 本文 (ボディ)

以下、これらのそれぞれについて解説し、「妥当な XHTML」に必要な要件について説明します。

8.10.1 XML 宣言

XML 宣言とは、XML のバージョンおよびその文書で利用している文字コードについて宣言を行うものです。と、これだけ読んですんなり理解することができる人はほぼいないはずですが、まずは実際の例を見てみましょう。

```
<?xml version="1.0" encoding="Shift_JIS"?>
```

上の例では xml という要素について、バージョンが 1.0、エンコーディングが「Shift_JIS」という属性付きで定義されています。

XML 文書はテキストであり、人間がテキストエディターで開けば読むことができるというものですが、その一方でコンピューターによる自動処理を常に念頭においています。この XML 宣言は 1 つにはそのようなコンピューターによる自動処理のためのもので、コンピューターがこれから処理すべき XML のバージョンと文字コードを問題なく理解することができるように配慮されています。

ここで文字コードとは、コンピューター内部で文字を表現するためのシステムであると考えてください。コンピューターが扱うことができるのは基本的に数字だけです。文字は数字に置き換えて表現されます。つまり、コンピューター内部には文字と数字の変換表がある、ということです。

日本語については、このような「変換表」、つまり文字コードの種類が、少なくとも 4 種類あります[25]。日本語以外にも様々な言語体系 (およびそれらに対応した文字体系) があり、それに応じた文字コードがあります。XML は UTF-8 および UTF-16 が基本とされており、これらの文字コードを利用していればエンコーディングの指定は省略することができます。逆に言えば、それ以外の文字コードを利用する場合は利用した文字コードを指定しなければなりません。

TeraPad などの、複数の文字コードを扱うことのできるテキストエディターを利用している場合は注意が必要ですが、Windows でも MacOS でも何も指定しなければ基本的にシフト JIS で作成されますので、前述の例の通り記述すればよいでしょう。ただし、思い込みは非常に危険ですので文書を作成する際に文字コードは必ず確認しましょう。

8.10.2　DOCTYPE 宣言

DOCTYPE 宣言は、その XHTML が準拠する XHTML(あるいは HTML) のバージョンを宣言するものです。次の例を参照してください。

```
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

ここではスペースの関係で 3 行に分割していますが、これらは半角のスペースで区切って 1 行にまとめてしまっても構いません。

この宣言は、全部で 7 つに分割することができます。

最初が「<!」で、これはマーク宣言 (markup declaration) と呼ばれます。2 文字で 1 つの記号であると考えてください。マーク宣言の次に記述するのが、宣言の種類を表す文字列です。ここでは「DOCTYPE」が入りますが、これは文書型宣言であることを示しています。マーク宣言との間に何か文字を入れることは禁止されていますので、「<!DOCTYPE」のようにぴったりとくっつけて記述をする必要があります。

次の「html」は XML におけるルート要素を示しています。つまり、XML は必ず木構造となるように記述が行われますが、そのルート要素をここで指定しています。この文書構造については後述します。

「PUBLIC」は外部識別子です。つまり、次に説明する公開識別子と呼ばれる文書を参照すべきことを指定しています。外部識別子と公開識別子を指定する代わりに、ここで DTD(Document Type Declaration) と呼ばれる文書の書法 (つまり、XHTML においてどのようなタグが利用されるかという情報) を長々と書いても構わないのですが、普通は切り離すわけです。

次のダブルクォーテーションマークで括られた情報が、公開識別子です。この公開識別子は所有者識別子 (-//W3C)、区切り記号 (//)、文識別子 (DTD XHTML 1.0 Transitional//EN) からなっています。

所有者識別子は、この外部文書の所有者が W3C(World Wide Web Consortium) であることを示しています。文識別子はさらに「公開分種別」(DTD)、「公開文記述」(XHTML 1.0 Transitional)、区切り子 (//)、公開文言語 (EN) から成っています。公開文記述は XHTML のバージョンおよび種類を表しています。

なお、最後の公開文言語は次に説明する公開文が記述されている言語を表しており、この DOCTYPE 宣言が書かれているファイルの記述言語を表しているわけではありません。

[25] 通常 JIS コードと呼ばれる ISO-2022-JP の他、EUC（イーユーシー）、シフト JIS、UTF-8（ユーティーエフエイト、またはユニコード）などがあります。電子メールとネットニュースには ISO-2022-JP を利用するという約束事がありますが、その他の場面では何を利用しても構いません。ただし、ファイルの中では統一するべきです。Windows や MacOS では主にシフト JIS が利用されます。このような文字コードの乱立は様々な混乱をもたらしていますが、徐々に解決される方向にあります。

次に、システム識別子を記述します。これは、DTD の所在を表す URL を記述します。最後に、マーク宣言終了区切り子である「>」で DOCTYPE 宣言を閉じます。

この DOCTYPE 宣言は、これから記述しようとしている XHTML のバージョンによって違うものを利用する必要があることに注意してください。上の DOCTYPE の例をそのまま暗記する必要はありませんが、自分の作成しようとしている文書がどの規格に沿ったものであるかということを良く理解して正しい DOCTYPE を記述するよう心がけましょう。

また、DOCTYPE は基本的にソフトウェアが利用する情報です。大文字・小文字も含めて、1 文字違わず正しく記述してください。

8.10.3　head 要素

XML 宣言と DOCTYPE 宣言に続いてヘッド要素を記述します。これは、文書の題名や著者など、文書そのものに関する情報を記述するものです。言い換えれば情報に関する情報ですから、これを「メタ情報」とも言います[26]。

ヘッド要素にはいくつでも好きなだけメタ情報を記述することができますが、ここでは必須であるものと一般的なものをいくつかとりあげておきます。

```
<head>
  <title>文書の題名</title>
  <meta http-equiv="content-type" content="text/html; charset=Shift_JIS"/>
  <meta name="author" content="著者の氏名" />
  <meta name="keywords" content="キーワード" />
  <meta name="description" content="ページの概略" />
</head>
```

ここでは、<head ></head >内に 5 つの要素が記述されています。

title 要素により、文書の題名を指定します。title 要素は head 要素内において必須の要素であり、必ず記述しなければなりません。また、title 要素は Web ブラウザのタイトルバーにも表示されます (図 8.12 参照)。このページをお気に入りに登録したときに表示されるのも、通常この title 要素です。他にも様々な理由がありますが、文書の題名は最も重要なメタ情報といっても構いません。

図 8.12: title 要素が表示されたブラウザのタイトルバー

その他 4 つの要素はすべて meta 要素です。最初の meta 要素ではこの文書の種類がテキストで HTML により記述されており、文字コードが Shift JIS であることが示されています。head 要素内には、title 要素と共に必ず記述してください。

[26] このヘッダ中のメタ情報はどちらかというと Web ブラウザや検索エンジンなどが利用するものです。つまり、コンピューターによる解析や分析に利用されることを想定しています。例えば address 要素のように文書の本文中に記述されるメタ情報もあります。

その他はいずれも name 属性と content 属性があり、name 属性でそのメタ情報の種類を、content 属性で内容を指定します。良く利用されるのは author、keywords、description などです。

上の例では title 要素を除きすべて meta 要素ですが、他にも link 要素や script 要素などが head 要素に入ります。

8.10.4　html 要素と body 要素

最後になりましたが、html 要素と body 要素を紹介し、また 1 つの XHTML ファイルの全体的な構造を示します。

html 要素は、DOCTYPE 宣言 (156 ページ参照) のところで、ルート要素になる要素として紹介しました。ルート要素がどのようなものであるか説明する前に、書き方を説明しておきましょう。

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja" lang="ja">
```

html 要素に属性が 3 つ付いています。後ろから説明しておきましょう。後ろの 2 つはそれぞれ言語情報のための属性です。これは、この XHTML で記述されている文書は自然言語の日本語 (ja) で記述されていることを定義しています。どのような言語を指定することができるか、というのは ISO(国際標準化機構) の規格である ISO 639-1 による言語表記を利用し、日本語の場合 ja です。

上の例で少々変なのは、言語に関する属性が 2 つ付いている、ということです。これは、XHTML 1.0 が HTML を XML で表現したものであるという二面性の産物です。XML では `xml:lang="ja"` と表記することに、また HTML では `lang="ja"` と表記することになっているため、XHTML 1.0 ではこの両方を記述することを勧めているのです。

最初の属性である xmlns は、名前空間 (Name Space) を指定しているものです。これまで見てきたように、XHTML では決められた要素とそれに対応した語彙 (タグ) で文書をマークアップします。逆に言えば、あらかじめ決められている要素に縛られ、拡張性を欠いているとも言えます[27]。

そこで、XML では XML で記述された言語 (XHTML もその 1 つです) を相互に乗り入れた運用というのを想定しています。例えば、XHTML で Web パブリッシングをするが、その文書は企業の経営成績を分析するもので、その資料となる財務諸表は XBRL(Extensible Business Reporting Language) で記載し、分析にあたって使用した数式を記述するのは MathML、また分析結果のグラフは SVG(Scalable Vector Graphics) で描き、参考資料として引用した新聞記事は NewsML で記述されている、といった使い方を想定しているのです。つまり、1 つの文書に様々な言語（ただしすべて XML）で書かれた文書が混じるということになります。

この際に問題になるのが、語彙 (タグ) が混乱してしまう、ということなのです。例えば既に title という要素を紹介しましたが、XHTML ではその文書の題名を定義します。しかし、別の XML で定義された言語では人の役職などの肩書きを意味するかもしれませんし、スポーツの選手権（いわゆるタイトルマッチなど）を意味するかもしれません。これを解決するのが名前空間なのです。名前空間は、その一意性を保証するために URL で区別することにしています。この URL はほとんど記号的な意味しか持ちませんが、実際にはこの URL にアクセスすると、この名前空間に関する何かしらの情報が得られることが期待されています。

さて、html 要素に戻って、ルート要素とはどのような意味であるか見てみましょう。すべての XHTML 文書では必ず 1 つの html 要素があり[28]、その中にすべての要素が含まれるということです。このような階層関係を樹状図 (ツリー) として描くと、html 要素が図の一番下 (あるいはその図を 180

[27] このような拡張性を確保する手段として、プラグインという手法がよく利用されてきました。例えば Adobe Flash や Apple 社の QuickTime などがその代表です。しかし、Flash にしても QuickTime にしても本来は私企業の独自規格でしかありません。Flash Player やブラウザのプラグインが無ければ見られないコンテンツというのは、それだけで困りものであるともいえます。

[28] 複数の html 要素が並列することはありません。html 要素の中に html 要素が含まれることもありません。1 つの XHTML 文書につき 1 つだけ、またあらゆる要素を包含するようにして存在しなければなりません。

度回転した時の頂点) に描かれます。そこで、このような階層構造の頂点にあるもののことを「ルート」と言います。例えば、Windows のファイルシステムでは、典型的には「C:¥」がディレクトリ構造の最上位階層であり、そこでこれを「ルートディレクトリ」と言います。

html 要素は、その中に head 要素と body 要素を持つという構造になっています。head 要素については前述の通りですが、この body 要素は本文を表します。既に段落や見出し等については説明しましたが、これらの本文に含まれるべき要素はすべて body 要素の中に記述しなければなりません。その他の XML 宣言や DOCTYPE 宣言も含めて、どのような順序で何を書かなければならないのかということを示すために、妥当な XHTML の簡潔な例を紹介しておきます。

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<meta http-equiv="content-type" content="text/html; charset=Shift_JIS"/>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja" lang="ja">

<head>
  <title>文書の題名</title>
</head>

<body>

<h1>見出し</h1>
<p>段落</p>

</body>

</html>
```

図 8.13 に、上の XHTML の構造を図示します。それぞれの宣言と要素の位置関係、包含関係を確認してください。また、html 要素を頂点とするツリー図を自分で書いてみると良いでしょう。

8.11 体裁を整える

8.11.1 体裁情報の記述方法

構造化文書を妥当（Valid）な XHML として文書化すればそれで終わりか、というと必ずしもそれで終わりではありません。実際には世の中は見た目もまた重要であって、中身が同じ品質ならば見栄えが良い方が選ばれやすいのもまた事実だからです。

ただし、これまでに作成してきたような XHTML 文書は、あくまでも文書構造を記述すべきものであって、そこに体裁情報を混ぜ込むべきではなく、外部から読み込むのが定石です。このような体裁情報を記述したファイルを「スタイル・シート」と呼びます。スタイル（style）とはまさしく体裁という意味ですが、例えば「h1 要素はゴシック体で 18 ポイントの大きさ、中央寄せする」といった情報が書かれているファイルです。

ただし、これをゼロから作成するのは大変ですので、通常はそれぞれのブラウザが標準で持っているスタイル・シートに重ね合わせて利用します。例えば「h1 は 24 ポイント」といったように、部

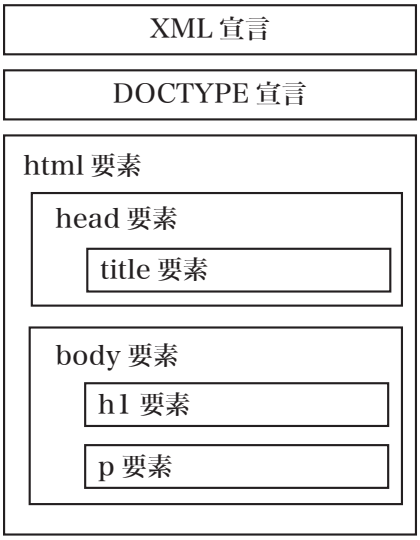

図 8.13: XHTML の構造

分的に上書きするわけです。

このように、重ね合わせて（cascade）利用することができるということで、カスケーディング・スタイル・シート（Cascading Style Sheet、CSS）と呼ばれるものを利用します。

XHTML の中に直接体裁情報を記述するのではなく、このような CSS を利用する理由はいくつか考えられます。

- XHTML 文書を複雑にすることなく簡潔に保つことができる
- 作成したデザインを使い回すことができる
- デザインの変更をすべての文書に一回で適用できる
- 文書作成とデザインを作業として切り分けることができる

8.11.2　CSS の作成と XHTML での指定方法

CSS は、ファイルとしての CSS を作成するという作業と、それを XHTML で指定して読み込むという 2 段階の作業が必要になります。まずは、CSS を作成してみましょう。

スタイルの記述方法は、次のようなものです。

```
p.strong { color: #ff0000; }
```

ここで、`p.strong` はセレクタと呼ばれます。XHTML の p 要素のうち、strong というクラス（class）を持つものについて、色を`#ff0000` にすべきことを指定しています。この、`{ color: #ff0000; }` という部分を規則集合といい、`color` をプロパティ、`#ff0000` を値（value）といいます。このプロパティと値として設定できるものは様々まものがあり、複数のプロパティを同時に設定することも可能です。

なお、ここでは色を RGB（Red、 Green、 Blue）の三色に分解してそれぞれを 00〜FF までの 16 進数表記により 256 段階の濃淡で表記しています。上の例では最も濃い赤となります。

一方、これに対応する XHTML は次のようになります。

```
<p class="strong">
ここは強調されます。
</p>
```

次のように class を指定せずに要素のみでセレクタを設定すれば、すべての p 要素について規則集合が適用されます。

```
p { color: #ff0000; }
```

一方で、要素を指定せずに class のみでセレクタを設定すれば、その class が指定されたすべての要素に付いて規則集合が適用されます。

```
.strong { color: #ff0000; }
```

```
<h1 class="strong">ここは強調表示されます</h1>
<p class="strong">ここも強調表示されます</p>
```

このようなクラスセレクタはブロック要素やインライン要素にまとめて適用することも可能です。

```
<div class="strong>
<h1>
これはブロック要素です
</h1>
</div>
<p>段落中に<span class="strong">インライン</span>でクラス指定</p>
```

プロパティには、フォント、フォントの色と背景色、高さ・幅、枠（ボーダー）、テキストの配置位置など様々なものがあり、これに対応する値にも様々なものがあります。詳細な一覧は W3C の Web ページ（英語）を参照するしかないのですが、CSS を解説している Web サイトは数多くあり、また無償で CSS をテンプレートとして提供しているサイトもありますので、検索して調べてみると良いでしょう。

最後に、CSS を「style.css」というファイル名で作成したときに、これを XHTML ファイル中で指定します。これは XHTML では以下のようにヘッダの中で指定します。

```
<head>
  <title>文書の題名</title>
  <link rel="stylesheet" href="style.css" type="text/css" />
</head>
```

Web サイトに XHTML を設置する際に、この style.css というファイルも一緒に（上の例では同じディレクトリに）設置しなければならないことに注意してください。

なお、ブラウザによって異なる CSS を使い分けるといったテクニックすら良く利用されているほどですので、CSS を利用する際は、ブラウザによって表示のされ方が異なることに注意してください。

8.12 ディスカッション

最後に、なぜ Web のようなシステムが考案されたのか考えておきましょう。

文書に限らない話ですが、知識はゼロから生み出されるものではありません。必ずどこかから発展されてきたものです。先人の積み重ねてきた知識を学ぶことは、不必要な議論を繰り返さないためにも重要なのです。そして、自分が行っている論考がどのような知識を下敷きにしたのかということを明らかにするのは、その一連の議論の流れのどこに自分の論考が位置づけられるのかということを明らかにするためにも必要なのです。

言論を行うにあたっては、自分一人が何を考えたか、ということを独りよがりで述べても意味がありません。文書が自分一人に向けて書かれているのであれば話は別ですが、他人が読むことを前提としている文書では何を参考にしたのか、議論の中でどれだけ別の議論を参考にしているか、読者が更にその問題について知りたいと思ったときに参考となる資料はどのようなものがあるかということを示すのは、当然のことです。

このように考えると、参考文献は文書の重要な構成要素であることがわかります。このような観点からも、学術論文では参考文献が重視されます。学術界では、ある論文の価値は「別の論文からどれだけ参照 (引用) されたか」ということを 1 つの基準としています。単純に考えれば、実質的に誰か別の人の論考でどれだけ役に立ったか、ということになります。

これが Web の基本的なアイデアである、ハイパーリンクの理念です。このような話はちょっと退屈かもしれません。しかし別の観点からも重要なのです。それは、Web ページの「発見されやすさ」です。

インターネットはさながら標識のない道のようなものです。Web ブラウザは直感的に操作することができますが、何も知らない人が自分の欲しい情報にたどり着くのは、さほど簡単ではありません。そこで利用するのが検索エンジンです。

Google という検索エンジンがあり、好評を得ています。この検索エンジンが特徴的なのは、シンプルな画面構成や検索が高速であること、何かキーワードを与えたときに結果として表示されるページの数が多いこと、つまり広い範囲の Web ページを調査していることなどがあげられます。

ところで、検索エンジンを利用しても検索結果はしばしば膨大で、自分の求める情報にたどり着けるとは限りません。Google の評判が良い最も大きな理由は、その検索結果の表示方法が的確であること、言い換えれば関連性の高いものから順に表示されており、自分の求めている情報に一発でたどり着けることが多い（と感じられる）ことです。この検索に利用されている技術の 1 つが Page Rank で、あるサイトの重要度を示す指標です。

この計算方法は、一言で表現すれば「多くの良質なページからリンクされているページは、やはり良質なページである」([7]) [29] ということになります。Google はスタンフォード大学で修士号を取得した 2 名の創業者によって始められた企業ですが、このアイデアが前述した学術界における論文の評価方法と基本的に同じであることが理解できます。紙であれ Web であれ、どのような文書でも適切な相互参照を行うことによって、情報はその情報自身を超えた価値を持ちうると言うこともできるでしょう。

その一方で、紙の文書で参考文献をたどっていくのは大変な作業です。参考文献一覧を参照してその資料を図書館で探すのは、ちょっとした論文なら 1 日仕事になります。ハイパーリンクは、このような参考文献をたどる作業を簡単にしてくれるのです。

一方で、文書に適切なリンクを付けるのは、文書を作成する者の責任であることを忘れてはいけません。また、自分の文書との関連性があり、リンクする必然性があるのならリンクを行うことをためらってはいけません。Web サイトの中には、その Web ページへのリンクを作成するのに事前の許諾を求めているものもあります[30]。このような場合、ある種の儀礼としてリンクを作成することをお願いするべきかもしれませんが、法律やガイドライン、マナーや倫理など、どのような価値観に

[29] この文書は文系でも理解できるように配慮されています。検索エンジンで検索されやすくする（あるキーワードで検索されたときに高い順位で表示されるようになる）ことを Search Engine Optimization、略して SEO と言いますが、検索エンジンのロジックを知ることは SEO の基本です。今後 Web パブリッシングを自分で行うことを考えている方には非常に参考になりますので、一読をすることをお勧めします。

[30] 大手新聞社などが 1 つの例です。

照らしてもリンクへの許諾を強制することはできません。これは逆も同じことで、自分が作成した文書を公開しておきながらそこへのリンクを禁止したり、リンクの許諾を強制してはいけません。

8.13 ファイル転送

以上のようにして作成した HTML ですが、このままでは世界のどこからも参照してもらうことができません。インターネット全体から参照することの可能な、Web サーバーという機能を持ったコンピュータのにファイルを転送し、設置する必要があります。ここではその手順を述べます。

この節では FFFTP について解説しますが、2010 年 1 月末に、FFFTP にはセキュリティ上の欠陥があることが判明しています。早稲田大学のコンピューター教室環境では大きな問題にはならないと判断しており、2010 年度も引き続きコンピューター教室で利用し続けることを検討しています。しかし、自宅のコンピューターに新たにインストールしたり、今後利用することがないように注意して下さい[31]。

8.13.1 FFFTP の利用

以下の手順で FFFTP によりファイルを転送します。

1. FFFTP を起動すると「ホスト一覧」ダイアログが表示されますので、「新規ホスト」をクリックしてください（図 8.14 左）。コンピューター教室の FFFTP では予めホストの設定がされています（図 8.14 右）。この場合は利用したいホストを選択して「設定変更」をクリックした後、ユーザ名などの必要な情報を入力します。

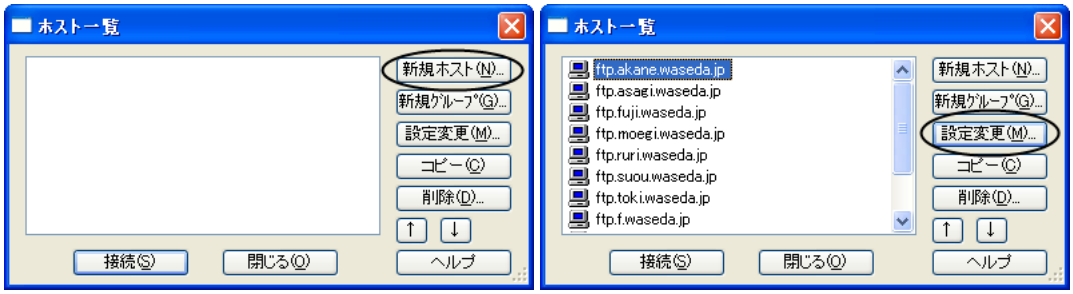

図 8.14: ホスト一覧

2. 「ホストの設定」ダイアログが表示されますので（図 8.15）、以下の項目を入力してください。設定終了後「OK」をクリックしてください。

ホストの設定名	FFFTP 起動後に表示されるホスト一覧に表示される名前です（任意の名称で構いません）。
ホスト名（アドレス）	ファイル転送先サーバのホスト名を指定します。このホスト名は利用する WWW サービスにより異なりますので、サービス申請時に配布される利用者控などを参照してください。
ユーザ名、パスワード	それぞれ Waseda-net のユーザー名（メールアドレスの@より左側の部分）と Waseda-net のパスワード。ゼミや学生団体用の場合は利用者控えを参照して下さい。
ローカルの初期フォルダ	これから転送するコンテンツが保存されているフォルダを指定してください。

[31]代わりに、WinSCP など別のプログラムを利用することを検討して下さい。

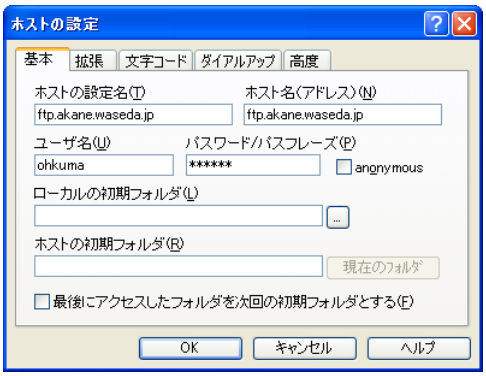

図 8.15: ホストの設定

3. 設定完了後、ホスト一覧に新規設定したホストが登録されますので（図 8.16）、これを選択し「接続」をクリックします。手順 1 で設定変更を行った場合は、そのホストを選択し「接続」をクリックします。

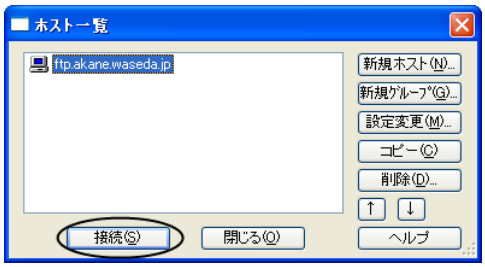

図 8.16: ホストに接続

4. 接続が正しく行われるとファイル転送用ウィンドウが表示されます（図 8.17）。左側にローカル側（利用している PC）フォルダが、右側にサーバ側フォルダが表示されています。

5. アップロードしたいローカル側のファイルを選択し、サーバ側のフォルダへドラッグアンドドロップを行うか、メニューの二段目にある上向き矢印アイコン（アップロードボタン）をクリックすることでアップロードできます。アップロード後、サーバ側のフォルダにアップロードしたファイル名が表示されていることを確認してください。

FFFTP の利用方法は、Web ページ上にも詳しい説明が掲示されていますので、そちらも参照して下さい[32]。

8.14　演習問題

テキストエディタを用いて Web ページを制作し、FTP により Web サーバーに送って Web パブリッシングを行いなさい。内容について、教員からの指示が無い場合は、「第 6 章 レポート・論文と作成支援」の演習問題にある「あたらしい憲法のはなし」をマークアップしなさい。この際、HTML-lint gateway サービス
`http://www.mnc.waseda.ac.jp/htmllint/htmllint.html`
を利用して、HTML を正しく作成できているかどうかチェックすること。

[32]「FFFTP 接続方法」`http://www.waseda.jp/itc/waseda-net/www-ftp.html`

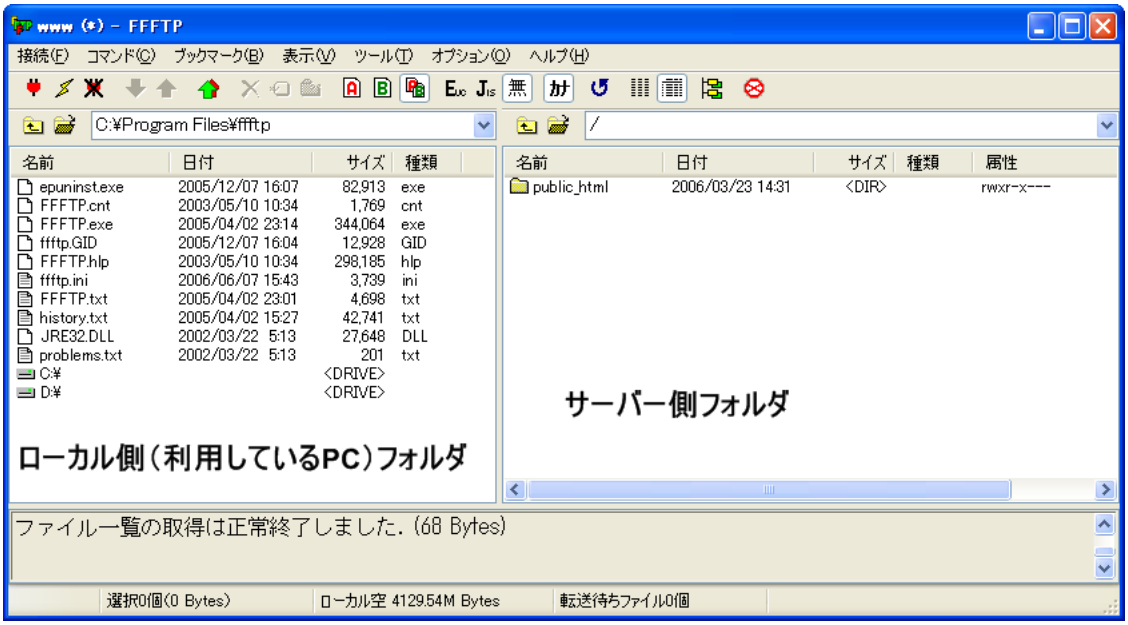

図 8.17: ファイル転送ウィンドウ