ヨーロッパ学 ICT 講義テキスト（IIA/IIB）

永 田 善 久

目次

［ヨーロッパ学 ICT 講義テキスト（IA/IB）より続く］

ヨーロッパ学 ICT IIA

第 1 講

欧文フォント

論理構成のみならず見栄えの点においても美しい文書を作成したいのであれば，文書中に用いるフォントにも注意を払わねばならない。

TEX はフォントを「エンコーディング（文字への番号の振り方），ファミリ（書体），シリーズ（線の太さ，ウェイトとも），シェイプ（同一ファミリにおける Upright, *Italic*, Small Caps のバリエーション），サイズ（文字の大きさ）」の 5 要素で管理しており，これらによりフォントを繊細かつ高度に制御する。TEX におけるフォントの取り扱いを深く学びたい場合は，各自参考書籍等を参照すると良い。

まず，エンコーディングについては，現段階（つまり XeTEX や LuaTEX といった TEX のユニコード・ネイティブ代替エンジンを使わない段階）では Preamble に以下の 2 つのパッケージ使用を指定する。なお，本講義で用いる TEX の入力ファイル（ソースファイル，ソースとも）の文字コードはユニコードの UTF-8，改行コードは LF に統一する。

- \usepackage[T1]{fontenc}
- \usepackage[utf8]{inputenc}

TEX では多くのフォントを扱うことができるが，多言語テキスト処理を目的とする本講義では，通常使用（論文やレポートの作成等）の欧文フォントを Times, Palatino, Linux Libertine, Noto に限定して解説する。

欧文フォントを指定する以下のコマンドを学べ。いずれか一つを Preamble に記しておく。

- \usepackage{newtxtext} (Times)
- \usepackage{newpxtext} (Palatino)
- \usepackage[sb]{libertine} (Linux Libertine)
- \usepackage{noto} (Noto)

使い分けの基準は「多言語テキスト表記におけるフォントをどこまで統一するか」（Pan-Language Harmony）に拠る。言うまでもなく，一つのテキスト内では複数のフォントを用いるよりデザインが揃っている一つのフォントを使った方が調和の取れた美的な見栄えとなる。上に挙げた各欧文フォントがカバーする文字の範囲は以下の通り。

- Times/Palatino: ラテン文字（英語，ドイツ語，フランス語等々）
- Linux Libertine: ラテン文字，キリル文字（ロシア語等），ギリシア文字，ヘブライ文字
- Noto: ラテン文字，キリル文字，ギリシア文字，ヘブライ文字，アラビア文字，漢字，ひらがな，カタカナ，ハングル文字，・・・

Noto は，実際には，単なる欧文フォントのレベルを超える多言語に対応した多種の字形を備える巨大なフォント・ファミリである。欧文・邦文を含むテキストを一種類のフォント・デザインで処理したい，という場合には Noto を選択するしかない（ただしこの場合はユニコード完全対応の XeTEX や LuaTEX を用いねばならない）。Linux Libertine を使う場合のオプション sb (semibold) は，これを付けておかないと**太字**（**boldface**）部分と地の文との間にやや不調和が生じるための措置である。なお，数式をも含めた調和美を追求する場合，Times/Palatino では {newtxtext,newtxmath}/{newpxtext,newpxmath} とするだけで良いが，Linux Libertine や Noto においては少々工夫が必要である。詳しくは参考書籍を参照すること。

上に挙げたフォントにつき，ファミリ（書体）をセリ

フ体（線の端に飾りの付いた字体，ローマン体とも），サンセリフ体（飾りの付かない字体），タイプライタ体（等幅のモノスペース体とも）に変更する以下のコマンドを学べ。これらのコマンドはいずれもテキスト本文に対して用いる。入力 → 出力 のように例示してある。

- \textrm{serif} → serif (default)
- \textsf{sans serif} → sans serif
- \texttt{typewriter type} → `typewriter type`

次に，シリーズ（ウェイト）を切り替えるコマンドと，欧文通常体の upright のシェイプを変更するコマンドを学べ。

- \textmd{medium} → medium (default)
- \textbf{Boldface} → **Boldface**
- \textup{Upright} → Upright (default)
- \textit{Italic} → *Italic*
- \textsc{Small Caps} → Small Caps

最後に，サイズに関しては（3 センチ，15 ミリ，2.54 インチ，12 ポイント，等々といった「絶対的」数値を指定して変更するのではなく），「相対的」にサイズを変更する以下のコマンドを学べ。昇順に並べてある。normalsize が標準文字サイズであり，この大きさはデフォルトで 10pt となっている。これを 11 や 12 にしたい場合は \documentclass コマンドのオプション部に 11pt あるいは 12pt のように記しておけば良い。いずれも宣言型コマンドであるから，これらのコマンドが発行された「後」の全テキストが有効範囲（スコープ）となることに注意。効果の範囲を区切りたい場合は {\small …} などとする。大文字・小文字の書き分けにも注意。

- \tiny, \scriptsize, \footnotesize, \small, \normalsize, \large, \Large, \LARGE, \huge, \Huge

和文フォント

PC 教室では「IPAex」（Windows 端末）と「ヒラギノ」（Mac 端末）をデフォルトの和文フォントとして設定してあり[*1]，いずれも明朝体が基本である。明朝体・ゴシック体のファミリを切り替える以下のコマンドを学べ。

- \textmc{明朝体} → 明朝体 (default)
- \textgt{ゴシック体} → ゴシック体

第 1 講演習

演習用ファイル： font_uptex.tex

1. font_uptex.tex を uplatex（複数回）と dvipdfmx で処理せよ
2. 入・出力ファイルを注意深く比較し，フォント要素の制御法を確認せよ
3. 各種コマンドは組み合わせて用いることもできる。色々と試してみよ。ただし，あらゆる組み合わせが有効というわけではない
4. 欧文フォント（デフォルトは Times）を Palatino, Linux Libertine, Noto にそれぞれ切り替えてみよ
5. 和文フォント（デフォルトは Windows では IPAex[*2], Mac ではヒラギノ）を切り替えてみよ（Windows では MS 明朝あるいは游明朝へ，Mac では IPAex へ）

コマンドを組み合わせて用いた際，当該字形の欠如により要素の切り替えがうまくいかない場合はデフォルトの字形で代用されることに注意。

発展： 論理的 vs. 視覚的デザイン

次のようなテキストを考えてみよう。

> 戯曲作品として **Goethe** は *Faust* を，**Schiller** は *Wilhelm Tell* を書いた。*Faust* の中には「セミ」（Zikade）という語が現れるが，実際には直翅目（バッタ，コオロギ等）を指している。日本よりも緯度の高いドイツに鳴くセミは存在しない。日本にはニイニイゼミ（*Platypleura kaempferi*），アブラゼミ（*Graptopsaltria nigrofuscata*），ミンミンゼミ（*Hyalessa maculaticollis*），クマゼミ（*Cryptotympana facialis*），ヒグラシ（*Tanna japonensis*），ツクツクボウシ（*Meimuna opalifera*）をはじめ，多種の鳴くセミが生息している。

[*1] 2021 年 8 月 24 日現在，Windows 端末における和文フォントは「游書体」（游明朝・游ゴシック）に変更してある。Mac 教室は本学に新しい「教育研究メディアシステム」が導入された 2020 年 9 月に廃止された。

[*2] TeX Live 2020 以降では「原の味フォント」（源ノ明朝・源ノ角ゴシックを Adobe-Japan-1 対応させたもの）がデフォルトの日本語フォントとなった。）

入力ファイルにおいて著者名に \textbf 命令，作品名と学名（属名および種小名からなる 2 名式）に \textit コマンドを施せば上掲の出力が得られるが，これは賢明なやり方であろうか。例えば，後になって「著者名を太字のスモールキャップス（Goethe）」へ，「作品名を対象言語固有の引用符号で括る方式（„Faust“）」へ，「学名をサンセリフのイタリック（*Platypleura kaempferi*）」へそれぞれ変更したくなった場合，\textbf や \textit の箇所を一つ一つ変更しなくてはならないが（一括置換という手もあるがそれでも），これはテキスト規模が大きいと時間のかかる大変な作業となる。さらに，\textit が論理（中身）の異なる「作品名」と「学名」の双方にマークアップされていれば，人間の目がその都度一々これらを区別して作業しなくてはならない。

\textbf や \textit という直接指定は「ここを太字に」「ここをイタリックに」というようにいわば「視覚的デザイン」をテキストにそのまま施していることに他ならない。TEX ではこのような流儀を取らず，テキストには「論理的デザイン」に基づいたマークアップをすることが推奨されている。

具体的には「著者名，作品名，学名」に相当する \authorname, \worktitle, \binomen といった新コマンドを Preamble で定義しておき（名前は既存のものと重複していなければ何でもよい），これらのコマンドを \textbf や \textit や\textsc 等に結び付けておく。こうすれば後でフォント要素を変更したくなった場合，Preamble の新コマンド箇所のみ書き換えれば済むので作業効率が格段に高まる。

発展演習

演習用ファイル： logical_visual.tex

1. logical_visual.tex を uplatex（複数回）と dvipdfmx で処理せよ
2. 入・出力ファイルを注意深く比較せよ
3. Preamble における新コマンドの定義法（引数を取る・取らない場合の違い）を確認せよ
4. 新コマンドの定義を色々と変えてみて，入・出力ファイルを比較してみよ
5. デフォルトでは「ドイツ語を欧文基底語」としているため「ドイツ語引用符」が出力されるが，これをフランス語用にしたい場合は Preamble において main=ngerman とある箇所の main を移動し main=french のように書き換えよ

第 2 講

TEX で作表する場合，テキストモードでは tabular 環境，数式モードでは array 環境が用いられることが多いが，本講では「横幅をユーザが指定できる」ように tabular を改良した tabularx（これは 1 ページ内に表が収まる場合に用いる）と「複数ページにまたがる表」を取り扱う場合の longtable パッケージについて解説する。

表組み： 1 ページに収まる表

1 ページに収まる表を作成するには，Preamble に次のように記しておく。

- \usepackage{tabularx}

そして，表を出力したいところには

```
\begin{center}
  \begin{tabularx}{幅}{列指定}
    表本体
  \end{tabularx}
\end{center}
```

と入力する。上例では tabularx 環境をそのまま center 環境で括っているので，tabularx で作成される表は「中央寄せ」される。これを「右寄せ」や「左寄せ」にしたい場合は，引数 center を flushright や flushleft に変えれば良い。

「幅」とある箇所に 10cm と入力しておけば，全体の横幅が 10cm の表が作成される。しかし，ここには\textwidth（テキスト領域幅）や 0.8\textwidth（テキスト領域幅の 0.8 倍）というような指定をする方が（自分で適当な数値を見出してくる必要がない分）簡単だし望ましい。

「列指定」箇所には，l（左寄せ）c（中央寄せ）r（右寄せ）X（改行を伴う列，折り返し幅は自動計算される）を指定する。欧文ではほぼ全く用いられない「縦罫」を付加したい場合は | を追記する。例えば |r|l|c|X| と指定すれば，各列間と両端に縦罫が引かれ，第 1 列は右寄せで，第 2 列は左寄せで，第 3 列は中央寄せで，第 4 列は複数行に自動改行されてそれぞれ出力される。

「表本体」では，列の区切りは & で，行の区切りは \\ で示す。\hline で「横罫」を引ける。最下行に \\ は付け

ない（\hline を施す場合を除く）。

複数の列をまとめて 1 列にするコマンドが \multicolumn{まとめる列数}{列指定}{中身} である。先の |r|l|c|X| を例に取ると，\multicolumn{4}{|c|}{\textbf 大見出し} とすることで 4 列をまとめて |**大見出し**| と中央寄せで出力できる。

表組み： 複数ページにまたがる表

tabularx パッケージを使えるのは表が 1 ページに収まる場合のみである。複数ページにまたがる「長い」表を作成する場合は，代わりに longtable パッケージを用いる。Preamble には次のように記しておく。

- \usepackage{longtable}

longtable の基本書式は

```
\begin{center}
\begin{longtable}[位置]{列指定}
 \caption{表の見出し}
  表分割時の付記設定
  表本体
\end{longtable}
\end{center}
```

である。tabularx では flushleft/center/flushright 環境で表の出力位置（左・中央・右寄せ）を制御したが，longtable ではオプション部の「位置」に l/c/r のいずれかを指定すれば良い。「列指定」箇所では l/c/r/p のパラメータが使える（X は不可）。p は tabularx における X におよそ相当するもので，改行を伴う列に対して用いる。ただし，p{0.5\textwidth} のように引数として「折り返し幅を明示的に指定」せねばならない。

「表の見出し」箇所には表のタイトルを書き入れる。後で「表目次」を自動生成させる場合，ここに書き入れた見出しが表目次に現れることになる。実際の「表の見出し」が長目で，表目次には代わりに短く切り詰めたものを用いたい場合 \caption[短い見出し]{実際の長い見出し} のように記しておく。

ページをまたぐことで表が分割される際，「表は次ページに続く」とか「前ページからの続き」とか「表はこのページで終わり」といったようなナビゲーション用の付記があると，長い表の全体像を見失わずに済む。「表分割時の付記設定」項目があるのはこうした理由による。

横罫，縦罫，列のまとめ方，については tabularx と全く同じ。tabularx や longtable ではもっと高度な作表もできるが，詳しく知りたい場合は，ターミナルから以下のように打ち込み，マニュアル（英文）を参照すること。

- texdoc tabularx
- texdoc longtable

第 2 講演習

演習用ファイル： tabularx_sample.tex, longtable_sample.tex

1. tabularx_sample.tex を uplatex（複数回）と dvipdfmx で処理せよ
2. 入・出力ファイルを注意深く比較し，1 ページに収まる表の作成法を理解せよ
3. longtable_sample.tex を uplatex（複数回）と dvipdfmx で処理せよ
4. 入・出力ファイルを注意深く比較し，ページをまたぐ表の作成法を理解せよ
5. 行，列を適当に増やしてみよ（もちろん対応する中身も）
6. 列指定を色々と変更してみよ

なお，インターネットで検索すれば CSV 形式のファイルを tabular 環境用のソースに変換するフリーのツール（Web アプリ，スタンドアロン・アプリとも）が色々と見つかるので，TEX で作表する際にはこうしたソフトを利用するのも一つの手である。例えば MS Excel で作表したものを CSV で保存し，ここから変換用ツールを使って tabular 環境用のソースを得る，というワークフローの方が作業効率が上がる場合も有り得よう。tabular 環境用ソースは tabularx のそれと大きな違いはない。もちろん，ソース細部への手入れは必要となる。

第 3 講

表の配置

longtable では TEX が表を複数ページに塩梅良く配置するよう自動分割してくれたが，tabularx を使う場合でも同様に表の配置は TEX に任せた方が良い。というのは TEX では図も表も原則として「一つの塊」として取り扱うため，これを 1 ページ内に配置できるだけのスペース量をユーザ自身が考えたり計算したりするのは大変だからである。このことは，特に完成稿を得るまでテキス

ト（量）が常に可変状態にあるような文書を書いている場面を想起すると理解しやすい。後から数行のテキストを追加しただけでもう当初予定していた図をはめ込むだけのスペースが確保できなくなり，結局その都度図の配置場所を手動で変更せざるを得なくなるとすれば，草稿の段階から図・表をテキスト内に決め打ちで配置していく文書作成法は賢明なやり方とは言えまい。

TEX では図・表と連携して用いるべく「浮動」（float）型の環境 figure と table が予め用意されている。これらの環境を使った場合，図・表の出力位置は TEX が適当に判断するので「今書いているこの文言と同じページ」に図・表が配置されるとは限らない。代わりに「相互参照」（Cross-Reference）という方式でこれら自動配置された図・表を参照するのが TEX の流儀である（詳しくは「相互参照」（8 ページ）で解説する）。

tabularx 環境を table 環境に入れてフロートとする以下の基本書式を学べ。

```
\begin{table}
 \caption{表の見出し\label{tab:midashi}}
  tabularx 環境
\end{table}
```

キャプションとは図・表の見出し（タイトル，説明）のことであるが，図では当該図下部に，表では当該表上部に付けることになっており（JIS X 4051『日本語文書の組版方法』より），欧文においても凡そ同様のルールである。

\caption 情報から「表目次」を自動生成させることもできる。この場合は表目次出力用のコマンド \listoftables をソースファイルに記しておくだけで良い。実際の見出しがとても長くなってしまい，図目次用の出力には短い見出しで代替したいときは \caption[短い見出し]{長い見出し} とする。

上の基本書式例では \label コマンドを \caption コマンド内に書き入れているが，代わりに \caption コマンドの外に出し，その直後に記してもよい。この記法は longtable でのやり方に合わせた次第である（longtable は tabularx に比べやや複雑な書式となっているので \label は \caption 内に一緒に記しておくとトラブルに見舞われにくくなる）。\label コマンドの使い方については次講で解説する。

図の配置

図も表の扱いとほぼ同じであるが，TEX で図を扱いたい場合は graphicx パッケージを用いる。最初に Preamble に次のように記しておく。

- \usepackage[dvipdfmx]{graphicx}

オプション部の dvipdfmx の箇所は，欧文のみからなるテキストを pdflatex で処理する場合には pdftex とする（LuaTEX や XETEX を使う場合はそれぞれ luatex あるいは xetex とする）。

図を figure 環境に入れてフロートとする以下の基本書式を学べ。

```
\begin{figure}
 \includegraphics[width=0.8\textwidth]{pi.jpg}
 \caption{図の見出し}
 \label{fig:midashi}
\end{figure}
```

図（画像）の形式としては *.pdf, *.png, *.jpg, *.eps が使える。上の例において図を実際に出力するコマンドは \includegraphics であるが，引数に画像ファイル名（例では pi.jpg）を記し，オプション部には画像の大きさ（例ではテキスト領域幅の 0.8 倍）を指定することもできる。

\includegraphics コマンド箇所全体を \fbox{…} で括れば図に「囲み線」（frame）を付けることもできる。

なお，TEX では 2 つの図を左右に並べたり（minipage 環境を使う），図の周りにテキストを回り込ませたり（wrapfig あるいは mawarikomi パッケージを用いる）することもできる。詳しくは参考書籍等を参照すること。

第 3 講演習

演習用ファイル：fig_tab.tex, 20160620_pubicornis.jpg

1. fig_tab.tex を uplatex（複数回）と dvipdfmx で処理せよ
2. 入・出力ファイルを注意深く比較し，図・表の自動配置および相互参照のさせ方を確認せよ
3. 入力ファイル中の \listoffigures（図目次出力コマンド）や \listoftables（表目次出力コマンド）をコメントアウトしてコンパイルしてみよ
4. 図に「囲み線」を付けよ

発展： ダミー・テキスト

fig_tab.tex には出力テスト用の「ダミー・テキスト」を挿入するための新コマンドを Preamble で定義してある。\dummytext[1] とすればそのダミー・テキストを 1 つ挿入でき，オプション部の数値を 100 にすれば 100 個のダミー・テキストが挿入される（オプション部には 1 以上の整数を代入する）。ダミー・テキストはライト・グレー表示（xcolor パッケージ使用）されるようにもしてある。

この新コマンドを定義する際には @ 記号も用いたが，この記号を使って新コマンドを作成する場合は必ず \makeatletter と\makeatother で囲わねばならないことに注意。

もう一つ。オリジナルの longtable におけるキャプション出力が table 環境におけるそれと少々異なるため，出力を table に合わせるべく longtable のソースコードに手を入れたものも Preamble に書き入れておいた。

なお，TEX を使う上でダミー・テキストを出力したいという要請は結構あるようで，このため lipsum（ヨーロッパの印刷・組版業界で伝統的に用いられてきた標準のダミー・テキスト *Lorem ipsum dolor sit amet* を出力する。元はキケロの『善と悪の究極について』から取られたものだが，オリジナルのラテン語テキストは相当崩されている）や blindtext（英語，ドイツ語，フランス語，ラテン語のダミー・テキストを出力できる）といったパッケージが用意されている。画像を扱う graphicx やカラーを扱う xcolor も含め，これらの使用法を詳しく知りたい場合はターミナルから以下のコマンドを打ち込む。

- texdoc lipsum
- texdoc blindtext
- texdoc graphicx
- texdoc xcolor

第 4 講

相互参照

次のような文章を考えてみよう。

> この術語については既に第 2.3 節（5 ページ）で詳しく説明した。関連して表 8（12 ページ）と図 5（11 ページ）も参照されたい。

このように，同一テキスト内で参照されるべきページ・章・節・図・表・数式・脚注等の番号を同テキスト内に入れることを「相互参照」（Cross-Reference）と呼ぶ。従って，目次や索引も広い意味での相互参照と言える。

例えば後から図を一つ挿入しただけで，関連する番号（当該図番号，当該図より後の図番号，当該図以降の図参照ページ番号等）が全て変わってしまうことを考えれば，相互参照処理は手作業では行わず，TEX に任せた方が賢明であることが分かる。

TEX における相互参照のやり方は，まず，参照したい番号を出力するコマンドの直後に \label コマンドを用いて「ラベル」を貼る。コマンドの「中」に貼っても良い。以下の例を参照。

```
\caption{この画像}
\label{fig:this_pic}
\caption{あの表\label{tab:that_tab}}
\section{これこれ\label{sec:this}}
\subsection{あれあれ\label{subsec:that}}
\footnote{なになに\label{fn:some}}
```

ラベル名は何でも良いが一意でなくてはならない。従って fig:（図），tab:（表），sec:（節），subsec:（小節），fn:（脚注），等々といった一種の前置文字列とともにラベル名を付けることが，その都度ラベル名を新たに考え出す労苦からかなりの程度解放されるためのコツである。

次に，参照したい箇所は \ref や \pageref コマンドの引数にラベルを記すことで，次のように呼び出す。

```
「これこれ」は
第\ref{sec:this}節
（\pageref{sec:this}ページ）で，
「あれあれ」は
第\label{subsec:that}節
（\pageref{subsec:that}ページ）で言及した。
図\ref{fig:this_pic}「この画像」
（\pageref{fig:this_pic}ページ）と
表\ref{tab:that_tab}「あの表」
（\pageref{tab:that_tab}ページ）も参照すること。
```

TEX で自動処理される相互参照は，畢竟，参照される番号（数字）の出力だけであるから，意味のある文にするためには「第（出力番号）節」とか「図（出力番号）」とか「（出力番号）ページ」といったような文脈に相応しい語句も併せて記さねばならない。

目次

目次を出したい箇所に \tableofcontents というコマンドを書いておくだけで，後は TEX が \section や\subsection といった箇所から自動的に目次を作成してくれる。欧文のみのテキスト処理において babel や polyglossia（いずれも多言語処理用パッケージ）を用いれば，「目次」という見出しではなく Inhaltsverzeichnis（基底言語がドイツ語の場合），Table des matières（フランス語），Contents（英語）といった見出しになる。「図」や「表」における前置語も自動的に Tabelle/Table/Table や Abbildung/Figure/Figure といった当該言語表記となる。

\section*{見出し} のようにアスタリスクを付けておくと，見出しの前に「番号が付かない」処理となる。従って相互参照もされなくなる。この「番号なし見出し」を目次に出力したい場合は

```
\section*{見出し}
\addcontentsline{toc}{section}{見出し}
```

のように \addcontentsline コマンドを用いて目次出力（toc）を明示しておけば良い。2 番目の引数には「どの目次階層に位置させたいか」を記す。例えば subsection レベルに出力したければ subsection とする。見出し階層のどのレベル（section, subsection, subsubsection 等）まで目次に出力させるか，といったことも制御できる。詳しくは参考文献等を参照すること。

索引

レポートや論文の作成において「索引」まで付けることはほぼないが，TEX では比較的簡単に索引を作成することができ，これも相互参照の一つであるから，併せて紹介しておく。索引に挙がる語句は自動的にアルファベット順，50 音順にソートされ，2 段組みで出力される。

索引を自動生成させるには，まず Preamble に

```
\usepackage{makeidx}
\makeindex
```

と記しておき，\end{document} の直前（言い換えるとテキストの末尾）に

```
\addcontentsline{toc}{section}{索引}
\printindex
```

と書いておくだけで良い。索引に載せたい語句には

```
Snow White\index{Snow White}
シンデレラ\index{シンデレラ}
白雪姫\index{しらゆきひめ@白雪姫}
König\index{Koenig@König}
```

のように \index{索引語} コマンドをマークアップしておく。このコマンドは \index{シンデレラ}シンデレラ のように前置しても良い。英字 52 文字やひらがな・カタカナではそのまま索引語を記すだけだが，漢字やドイツ語ウムラウト，フランス語アクサン記号等を含む場合は \index{よみ@索引語} のように処理する。こうすることで初めてきちんとソートされるようになる。

索引マークアップを含む邦文ソースファイル（例では file.tex）は次の手順でコンパイルする。

1. uplatex *file*
2. upmendex *file*
3. uplatex *file*（複数回）
4. dvipdfmx *file*

なお，索引スタイルも細かく制御できる。詳しくは参考書籍等を参照すること。

発展： 欧文索引語のソート

索引を含む邦文ソースファイルは upmendex プログラムで処理するが，欧文のみかならなるソースであれば欧文多言語対応がなされた xindy プログラムを使うと良い。この場合は \index{よみ@索引語} といった記法は不要となり，直接 Ärger\index{Ärger} のようにマークアップできる。

索引マークアップを含む欧文ソースファイル（例では file.tex）は以下の手順でコンパイルする。

1. pdflatex *file*
2. texindy -L german-din *file.idx*
3. pdflatex *file*（複数回）

索引語のソート順は言語によって異なるため，xindy では -L という言語オプションによってソート順を指定する。english（英語），french（フランス語），german-din（ドイツ語，ドイツ工業規格準拠国内仕様；ä は ae としてソート），german-duden（ドイツ語，Duden 辞書準拠グローバル仕様；ä は a としてソート）等が使える。なお，xindy を用いる場合のコマンドは texindy であるこ

とと，file.idx のように拡張子まで打ち込まねばならないことに注意すること。

xindy や upmendex の使用法を詳しく知りたい場合はターミナルから以下のコマンドを打ち込む。

- texdoc xindy
- texdoc texindy
- texdoc upmendex

ハイパーリンク

テキスト内に埋め込まれた，外部ファイルや同テキスト内番号位置への参照情報をハイパーリンクと言い，ハイパーリンクが埋め込まれたテキストをハイパーテキストと呼ぶ。ハイパーテキストの代表は HTML (HyperText Markup Language) で記述された Web ページであるが，hyperref パッケージを用いることで TEX ソースも簡単にハイパーテキスト化できる。

TEX ソースをハイパーテキスト化するには hyperref パッケージを読み込む。Preamble に次のように記しておく。

- \usepackage[dvipdfmx]{hyperref}

オプション部には処理エンジンに応じて pdftex や xetex 等を指定する。LuaTEX の場合は pdftex,unicode=true と指定する。こうしておかないと日本語の「しおり」が文字化けする。なお hyperref パッケージの指定は Preamble の後ろの方で行う。hyperref は他パッケージ内のコマンドを書き換えてしまうことがあるので，最後の方で指定しないとパッケージ間の相性によっては意図した出力が得られないことも出来し得る。

次に，外部ファイルへのハイパーリンクは \href あるいは \url コマンドを用いて以下のように書く。

```
\href{http://apapa.hum.fukuoka-u.ac.jp/}{Apapa
サーバ}
\url{http://apapa.hum.fukuoka-u.ac.jp/}
```

前者では第 2 引数「Apapa サーバ」が，後者では引数部分がそのまま出力され，それぞれハイパーリンク箇所となる。

さて，TEX ソースをハイパーテキスト化する醍醐味は \href, \url コマンドを使用すること以上に，先に解説した「相互参照」箇所が全て自動的にハイパーリンク処理される点にこそある。また，\hypersetup コマンドを使えば出力される PDF ファイルに「しおり」(bookmarks) や文書メタ情報 (タイトル，作成者，サブタイトル，キーワード等) を付加することもできる。その際，邦文 TEX ソースにおいては，日本語部分の文字化け回避のため pxjahyper パッケージを hyperref の後に指定しなくてはならない。LuaTEX や XeTEX を用いる場合は pxjahyper は不要である。

PDF に「しおり」や文書メタ情報を付加するための以下の基本書式を学べ。

```
\usepackage[dvipdfmx]{hyperref}
\usepackage{pxjahyper}% 日本語 PDF「しおり」用
\hypersetup{%
 bookmarksnumbered=true,
 colorlinks=false,
 setpagesize=false,
 pdftitle={タイトル},
 pdfauthor={著者},
 pdfsubject={サブタイトル},
 pdfkeywords={キーワード}}
```

bookmarksnumbered は節番号等まで「しおり」に含めるか否か，colorlinks はリンク箇所を赤文字とするか否か（false にすると「赤枠」リンクとなる），setpagesize はページサイズ変更を許可するかどうか，pdftitle は文書プロパティの「タイトル」，pdfauthor は「著者」，pdfsubject は「サブタイトル」，pdfkeywords は「キーワード」，をそれぞれ制御する引数となっている。

hyperref とともに \printindex コマンドを用いる際は，次のように \clearpage および \phantomsection コマンドをすぐ上に追記しておくこと。

```
\clearpage% 索引は改ページして出力
\phantomsection% hyperref で必要
\addcontentsline{toc}{section}{索引}
\printindex
```

第 4 講演習

演習用ファイル：cr_ind.tex, cr_ind_hyp.tex, sneewittchen.jpg

1. cr_ind.tex は cr_ind_hyp.tex 内の hyperref に関わる箇所をコメントアウトしただけのソースファイルとなっている。このことを確認せよ
2. cr_ind.tex を uplatex, upmendex, uplatex（複数回），dvipdfmx で処理せよ

3. TEX ソースファイルと生成された PDF ファイルを良く見比べよ
4. cr_ind_hyp.tex についても同様にコンパイルし，入・出力ファイルを良く比較した上で，ハイパーリンク箇所を全て検証せよ

発展： PDF をパスワードで保護

有償ソフトウェアを用いずとも，例えば以下のオプションを与えて dvipdfmx を実行するだけで PDF にパスワードを付与する（暗号化を施す）こともできる。

- dvipdfmx -S -P 0x0804 -K 128 *file*

S は暗号化許可，P は「印刷，変更，編集，コピー，テキスト抽出，ページ抽出，注釈，フォーム入力，署名，テンプレートページ作成」のいずれを認めるかについてのフラグ値，K は鍵長，に関するオプションとなっており，上例では「閲覧にはユーザ・パスワード，設定変更にはオーナー・パスワードが必要，高解像度での印刷以外は全て不許可」の PDF が生成される。ターミナルでコマンドを実行すると，Owner password や User password を尋ねられるので，Re-enter owner password 等も含め，その都度正しく入力する。

dvipdfmx のマニュアルはターミナルから以下のように打ち込めば参照できる。その中の Encryption Support が暗号化に関する説明となっている。

- texdoc dvipdfmx

欧文のみから成るテキストは pdflatex でコンパイルし PDF を出力するが，pdflatex にはファイルに暗号化を施す機能は実装されていない。この場合は PDFtk（TEX システムに含まれないので別途ダウンロードが必要）というフリーソフトのコマンドラインツールを用いる（もちろん Adobe Acrobat 等の有償ソフトウェアを使っても良い）。

- pdftk *orig.pdf* output *new.pdf* owner_pw *hoge* user_pw *hage* allow *Printing*

とターミナルに打ち込めば，元ファイル orig.pdf から上で説明した dvipdfmx と同じ暗号化の施された new.pdf を生成できる。

発展演習

演習用ファイル： cr_ind_hyp.tex

1. cr_ind_hyp.tex を uplatex, upmendex, uplatex（複数回），dvipdfmx で処理せよ
2. 次に，dvipdfmx -S -P 0x0804 -K 128 で処理せよ。その際パスワードを適当に設定せよ
3. 生成された PDF を開く際にはパスワード入力を求められることを確認せよ

第 5 講

文献参照の仕方

これまで言及してこなかったが，前講の演習用ファイル cr_ind[_hyp].tex では「参考文献」も TEX により相互参照処理されていた。本講では TEX における文献参照の仕方について説明する。

まず，参考文献リストはユーザが作り，文献参照番号は TEX に付けさせる方法を学ぶ。この場合，参考文献はユーザが並べ（ソートし）た順に出力される。このためには以下のような基本書式から成る参考文献リストを入力ファイルの末尾に記しておく。cr_ind[_hyp].tex においても本方式で参考文献を処理している。邦文文献そして欧文文献の具体的な記載例は演習用ファイルを参照すること。

```
\begin{thebibliography}{9}
\addcontentsline{toc}{section}{参考文献}
\bibitem{参照キー}
 著者名『著書タイトル』. 出版社, 出版年.
\bibitem{anykey}
 Familyname, F.: \textit{Booktitle}.
 Publisher. Address, Year.
\end{thebibliography}
```

thebibliography 環境の 9 という引数は，リストアップする文献数の「桁」に合わせて 9, 99, 999, … 等とする。つまり参考文献数が 8 点であれば 9，14 点であれば 99 となる。これは TEX が自動的に振る文献番号に関してその「幅」を決めるために参考とする数値としての役割を果たすだけのものであり，特別の意味はない（従って 1, 42, 365, … でも同じ効果となる）。

\addcontentsline コマンドにより「目次」の「セクション（節）」レベルに「参考文献」が出力されるよう指定

できる。参照キー（anykey）には一意の文献参照キーを設定する。以上を踏まえて，本文には

```
・・・とある（グリム／金田~\cite{khm_ja} p.130）。
... die Königin\footnote{Grimm~\cite{khm_de}
 p.269.}.
```

のように書く。各文献に与えておいた一意の参照キーを \cite コマンドの引数に指定することで当該文献を呼び出す。TEX により自動出力されるのは参照番号だけであるから，意味のある記述にするため「著者姓」を \cite コマンドの前に付けておくことは，相互参照に共通の処置である。

なお「著者姓~\cite」のように \cite コマンドの前に~（チルダ）記号が付加されている点にも注意すること。~ は「そこでの改行をさせない空白」を出力するコマンドである。なぜこのコマンド併記が必要であるのか，良く考えてみると良い。

また，文献参照に限らず相互参照を含む TEX ソースは複数回コンパイルに掛けねばならないことにも注意すること（様々の補助ファイルが自動生成され，それらをさらに読み込んで最終処理がなされるため）。

文献データベースとの連携

次に，ソースファイルとは別に文献データベースを用意しておき，この文献データベースから upBIBTEX（欧文ソースの場合は BIBTEX）というプログラムを使って自動的に参考文献リストを作成する方法を紹介する。

用意する文献データベースは *.bib という拡張子を持つテキストファイルである。この中に以下の書式にて文献情報を記しておく。とりあえず「書籍」と「論文」の基本書式を挙げておいたが，他にも様々な種類の文献を取り扱えるので，詳しくは参考文献等を参照すること。EURO ICT Emacs では *.bib ファイルを開くと major mode が BibTeX となる。本講では bibtex-mode の使い方までは解説しない。興味のある者は Emacs 上で C-h f と打ち Describe function: に bibtex-mode と入力すればマニュアルを参照できる。

なお，邦文ソースを upBIBTEX で処理する場合，文献データベース内のウムラウトやアクサンが含まれる語は例えば Brüder, Frères の代わりに `Br{\"u}der`, `Fr{\`e}res` 等と書かねばならない。欧文のみから成るソースを BIBTEX で処理する場合はこの措置は不要である。

```
@book{参照キー,
 author = "著者名",
 title = "『著書タイトル』",
 publisher = "出版社",
 address = "出版地",
 year = "出版年",
}

@article{参照キー,
 author = "著者名",
 yomi = "ちょしゃめい",
 title = "「論文タイトル」",
 journal = "掲載誌名",
 number = "巻数",
 year = "出版年",
 month = "出版月",
 pages = "掲載ページ",
}
```

参照キーは欧文・邦文のいずれでも良いが一意でなくてはならない。著者名は「Familyname, Firstname」（欧文），「森 鷗外」（邦文；姓名間にスペースを挿入）のように記す。

邦文においては『著書タイトル』，「論文タイトル」のように対応するカギ括弧も含めて記入する。欧文の場合は自動的にイタリック体処理されるので，括弧は付けない。邦文文献そして欧文文献の具体的な記載例は演習用ファイルを参照すること。

なお @book エントリにおいては address を除き，上に挙げた author 欄以下全て入力必須である。@article では yomi, number, month, pages 以外が入力必須欄となっている。

yomi 欄に「ひらがな」で入力しておけば，漢字を含む邦文タイトルでも正確にソートされるようになる。yomi を使えるのは upBIBTEX の場合のみである。

さて，このような文献データベースを用意したら，TEX ソースに以下のように書き込む。

```
\nocite{*}
\bibliographystyle{jplain}
\phantomsection% hyperref で必要
\addcontentsline{toc}{section}{参考文献}
\bibliography{文献データベース名}
```

\bibliography の引数には拡張子を除いた文献データベース名を入れる（khm.bib であれば khm）。\bibliographystyle で文献スタイルファイルを指定できる。jplain（欧文の場合は plain）以外にも多種のスタイルファイ

ルが使えるので，詳しくは参考文献等を参照すること。\phantomsection と\addcontentsline については既に説明した。

文献データベースはコンマで区切ることによっていくつでも指定できる（例： \bibliography{khm1,khm2,khm3}）。\cite{参照キー} で文献を呼び出す方法は全く同じであるが，文献データベースを使う場合は原則として \cite 参照された文献のみが参考文献にリストアップされる。これに対し \nocite{*} と書き入れておけば，\cite 参照の有無に拘わらず文献データベースにある全文献が参考文献として出力される。

文献データベースと連携した TEX 邦文ソース（例では file.tex）のコンパイル法は以下の通り。

1. uplatex *file*
2. upbibtex *file*
3. upmendex *file*
4. uplatex *file*（複数回）
5. dvipdfmx *file*

以上により，著者の姓（同姓の場合は名，さらに同名であれば出版年，さらに同出版年ならタイトル）の「アルファベット，50 音」順に並んだ参考文献リストが自動作成される。

第 5 講演習

演習用ファイル： cr_ind_hyp_bib.tex, khm.bib

1. cr_ind_hyp_bib.tex および khm.bib の中身を精査せよ
2. cr_ind_hyp_bib.tex を uplatex, upbibtex, upmendex, uplatex（複数回），dvipdfmx で処理せよ
3. TEX ソースファイルと生成された PDF ファイルを良く見比べよ
4. ハイパーリンク箇所も全て検証せよ

発展： (iso-iso2tex)

邦文ソースを upBIBTEX で処理する場合，文献データベース内では `Br{\"u}der`, `Fr{\`e}res` のようにウムラウトやアクサン等を入力（これが TEX における本来のマークアップ法）しておかねばならないことは既に述べたが，この記法はやはり面倒くさい。しかし，Emacs を使えばこの種の変換は以下のコマンドにより簡単に行えるので紹介しておく。

- (iso-iso2tex)

リージョン指定してから M-x を前置しコマンドを実行すること。逆コマンドは (iso-tex2iso) である。

発展演習

演習用ファイル： eu_bib.tex, khm_eu.bib

1. khm_eu.bib 内のウムラウトやアクサンが含まれる語を TEX 式マークアップに変換し，khm_eu2.bib というファイル名で保存せよ
2. eu_bib.tex を pdflatex, bibtex, texindy -L germanduden *.idx, pdflatex（複数回）としてコンパイルしてみよ
3. 入・出力ファイルを良く比較せよ

発展： 文献ソート順の調整

文献データベースからは著者の姓（姓が同じなら名，等々）を基に参考文献がアルファベット，50 音」順に並べられることは既に述べた。演習用ファイル cr_ind_hyp_bib.tex をコンパイルすると 確かに Brothers Grimm, Brüder Grimm, Frères Grimm の順に参考文献が挙がる様を確認できる。場合によっては，しかし，この並び順を変えたいこともあろう。

文献の自動ソートを制御するには *.bib ファイル先頭に

```
@preamble{"\newcommand{\noop}[1]{}"}
```

と書いておき，並び順を変えたい箇所に，例えば

```
\noop{grimm3}Grimm, Br{\"u}der
\noop{grimm1}Grimm, Fr{\`e}res
\noop{grimm2}Grimm, Brothers
```

のように新たな「変更参照キー」grimm1/2/3 を付加すれば良い。もちろん望むソートとなるように変更参照キーを付けねばならない。

発展演習

演習用ファイル： cr_ind_hyp_bib_sort.tex, khm2.bib

1. khm2.bib の中身を点検せよ
2. cr_ind_hyp_bib_sort.tex を uplatex, upbibtex, upmendex, uplatex（複数回），dvipdfmx で処理せよ

3. 入・出力ファイルを良く比較せよ

第 6 講

アウトライン的文書作成法

EURO ICT Emacs では *.tex ファイルを扱う際の major mode を AUCTEX（デンマークの Aalborg University Center で開発されたためこのアクロニムとなっている）に設定してあり，mode line に LaTeX/MPS（M: Math., P: PDF, S: [Correlate Input/Output] Search）と表示されることでこのことが確認できる。AUCTEX は実に多機能で，TEX に関するコマンド入力補完，自動コンパイル，デバッグ等々を Emacs から全て制御できるような，洗練された TEX 処理系をユーザに提供してくれる（Syntax Highlighting もそれらの機能のうちの一つである）。

本講義では，時間の制約上，AUCTEX の使い方には触れない。興味のある者は EURO ICT Emacs 上で

- C-h i m AUCTeX

と打ち込み，マニュアルを参照のこと。

さて，*.tex ファイルを EURO ICT Emacs で開くと mode line の LaTeX/MPS の後に Ref と表示されていることにも気付くであろう。EURO ICT Emacs では TEX ソースを開くと同時に RefTEX も有効となるように設定してある。RefTEX は TEX の相互参照に関わる入力をサポートしてくれる Emacs の minor mode である。

AUCTEX 同様 RefTEX も多機能である。詳しい使用法を知りたい場合は

- C-h i m RefTeX

と打ち込み，マニュアルを参照のこと。

RefTEX はラベル作成，ラベル参照，参考文献からの引用，索引作成といった事柄に関わる入力を強力に支援してくれるが，それと同時に本講のテーマである「アウトライン的文書作成法」の基礎となる TEX ソースファイルの「目次」をも，コンパイル後の出力を待たずして，入力作業中にリアルタイムで表示してくれる。

論文のような構造化された文書の「アウトライン編集」については，既に「ヨーロッパ学 ICT IB 講義テキスト」（第 6 講「アウトライン編集」）の中で Org を major mode としてテキスト処理する方法を学んだ。論文のラフ・スケッチを練るような草案段階で用いる文書作成ツールとしては Org モードの方がより相応しいかも知れないが，完成稿を見据えた TEX ソースの高度かつ効率的な編集には AUCTEX および RefTEX モードを用いるのが良い。

EURO ICT Emacs 上で作業中の TEX ソースファイルの目次を即座に作成する以下のコマンドを学べ。

- TEX ソースの目次作成： C-c = (reftex-toc)

コマンドを打ち込めば，同一 frame 内の上側別 window に TABLE-OF-CONTENTS が直ちに出現する様を確認できる。併せてこの別 window 内の目次 buffer をコントロールする以下のキーを学べ。全て目次 buffer 内で操作する。

- n: 次のエントリへ [n]ext
- p: 前のエントリへ [p]revious
- TAB: TEX ソースの当該箇所へジャンプ
- RET: 当該箇所へジャンプして目次 buffer を閉じる
- SPC: ジャンプせずに当該場所を確認
- q: 目次 buffer を閉じる [q]uit
- r: TEX ソースの変更を反映させる [r]escan
- f: TEX ソースと連動 [f]ollow（toggle キー）
- l: ラベルも表示 [l]abel（toggle キー）
- i: 索引語も表示 [i]ndex（toggle キー）
- C-c C-n: 次の見出し（section 系のみ）エントリへ
- C-c C-p: 前の見出し（section 系のみ）エントリへ
- <: 見出しを昇格
- >: 見出しを降格
- ?: 目次 buffer の使い方を調べる

昇・降格の際サブツリーも対象としたい場合は，予め目次 buffer 内で region 指定しておく。

Org モードには見出しをサブツリーごと同レベルの見出しの前・後に移動する機能までもが備わっていたが，同様のことを RefTEX の目次 buffer で行うことは，残念ながら，できない。その代わり，EURO ICT Emacs では TEX ソース buffer 側で以下のキー操作を行うことで，同じことができるように設定してある（Outline という minor mode を読み込ませている）。cursor はツリーの最上層に置くこと。

- 見出しをサブツリーごと上へ： C-c @ C-^ (outline-move-subtree-up)
- 見出しをサブツリーごと下へ： C-c @ C-v

(outline-move-subtree-down)

以下では，アウトライン的文書作成法以外の，相互参照に関する RefTEX のコマンドをまとめておく。

ラベルの処理法

TEX ソース内にラベルを手作業で入力していくことはもちろんできるが，以下のコマンドを知っているとラベル貼付の作業効率を高められる。

- cursor 位置に label 挿入： C-c ((reftex-label)
- label 参照： C-c) (reftex-reference)

C-c (を打ち込むと section 系領域では Label: sec: のように，table や figure 環境では Label: tab: や Label: fig: のようにそれぞれ minibuffer で入力を促されるから，そこで適当なラベル名を打ち込めばよい。: の後には RefTEX が予め適当な前置語句を引数情報から引っ張ってくることもある。

C-c) を打つと SELECT A REFERENCE FORMAT と Emacs が言ってくるので [^M] (\ref) あるいは [p] (\pageref) のいずれかのキーを叩いて答える。^M とは C-m のことである。すると次に，一瞬の間（この間が無駄と感じる上級者は所望の選択キーを直ちに押しても良い）を置いて SELECT A LABEL TYPE: と尋ねられるので [ ] つまり SPC を押す。するとソース内の全てのラベルが別 buffer に挙がるので，そこから望みのラベルを選択すれば良い。SPC の代わりに [t]able, [f]igure, [n]footNote, [s]ection を押せば，それぞれ表，図，脚注，見出し系に領域を絞ってラベルが列挙される。

ラベル参照 buffer 内は n/p/<up>/<down> キーで移動できる。RET を叩いてラベルを選択するとこの buffer は消えるが，選ぶ前に消したい場合は q を押す。

なお，RefTEX はデフォルトが欧文仕様となっているので C-c) でラベル参照をすると ~\ref のように \ref/\pageref の前に ~（そこでの改行を禁じる空白）が自動的に付いてしまう。邦文ソース内ではこの ~ は不要なので，少々面倒ではあるが，その都度削除する。

文献引用の処理法

文献引用の際は，以下のコマンドを打つ。

- 引用文献のサーチと選択： C-c [(reftex-citation)

C-c [と打つと Regex { && Regex…}: […] のように minibuffer にメッセージが出るので，文献タイトルの一部や文献参照キー（もし覚えていれば）等を入力してやると，その入力語句の正規表現に対応する文献が別 buffer にリストアップされる。ここから目当ての文献を選べば良い。文献参照 buffer 内はラベル参照時同様 n/p/<up>/<down> キーで移動できるし，q で抜けられる。

なお，\cite は ~\ref の場合と異なり ~ が自動的には付かないので，手作業で ~\cite のようにすること。

索引の処理法

索引語としたい語の上に cursor を置き以下のコマンドを打つ。欧文のように語と語の間に space がある場合は索引語のどの位置でコマンドを打っても狙った語を把捉できるが，邦文の場合は region 指定してからコマンドを実行しないと索引語をうまく捉えられない。

- cursor 位置の語を索引語とする： C-c / (reftex-index-selection-or-word)

なお上記コマンドでは \index が索引語の前に付されることに注意（もちろん \index は前置・後置のどちらでも可）。

索引関連では，その場で索引語を編集できる C-c < (reftex-index) や，大掛かりな索引作成のために専用の索引語句データベース（*.rip）を作成・活用する C-c \ (reftex-index-phrase-selection-or-word) をはじめ多くのコマンドが用意されているが，本講では他には「索引語を一覧表示」する以下のコマンドのみを紹介する。

- TEX ソース内の索引語を別 buffer に一覧表示： C-c > (reftex-display-index)

索引語一覧 buffer の使い方は目次 buffer とほぼ全く同じであるため説明は省くが，e（編集 [e]dit）というキー押下により索引語を minibuffer で編集できる機能もある。minibuffer での編集が済めば RET を打つ。この場合，索引語一覧 buffer には EDITED と表示されるので [r]escan キーを押して変更を反映させておく。

第 6 講演習

演習用ファイル： cr_ind_hyp_bib.tex, khm.bib

1. TEX ソースを開き，目次 buffer を検証せよ
2. 同様に索引語一覧 buffer を検証せよ

3. C-c (, C-c), C-c [, C-c / コマンドを用いてラベル，文献引用，索引，それぞれの処理法を試してみよ

第 7 講

奇数・偶数ページのレイアウトが変わるスタイル

これまで学んできた EURO ICT Emacs & TEX に関する諸コマンドを駆使することで，今や，見栄えのする論文（中身ではないことに注意！）を執筆するための初歩的スキルを身に付けることが可能となったと言えよう。

本講では，そこから一歩進んで，本格的な論文を作成するための「雛形書式」となる TEX ソースファイルを提示することにより，論文の組版に関わる一連のテーマを締め括りたい。

具体的には，福岡大学大学院人文科学研究科独語独文学専攻における邦文による修士論文作成要領である「とびら，目次，論文の要旨，本文（1 ページ 25 行，1 行 32 文字；1 ページあたり 800 字が目安），引用文献」を満たすような雛形 TEX ソース twoside_master.tex を掲げる。この雛形ソースには

- 奇数・偶数ページのレイアウトを別々にする
- 柱（各ページ上部に出力する章・節タイトル）を付ける
- 偶数（左）ページの柱には章タイトルとノンブル（ページ番号）
- 奇数（右）ページの柱にはノンブルと節タイトル
- 章を奇数（右）ページ起こしにする
- 表・図の番号は章ごとに分けて振らず，論文全体における通番とする

というような「両面印刷用に最適化されたページレイアウト」を施したドキュメントクラス jsbook_lglf.cls を用いる（オリジナルの jsbook.cls を少々変更し，リネームして使っている）。

twoside_master.tex の構造

奇数・偶数ページのレイアウトを変える組版のための雛形 TEX ソースファイル twoside_master.tex の基本構造は以下の通り。

```
\documentclass[...,titlepage]{jsbook_lglf}%
% 「とびら」を独立ページに
% Preamble
\title{...}% タイトル
\author{...}% 執筆者名
\date{\today}% 今日の日付； 引数ブランクで無出力
\usepackage{makeidx}% 索引作成用パッケージ
\makeindex% 索引作成
\begin{document}
\maketitle% 「とびら」作成
\frontmatter% 前付け
\tableofcontents% 「目次」作成
\listoftables% 「表目次」作成
\listoffigures% 「図目次」作成
\chapter{論文の要旨}% 「論文の要旨」
\mainmatter% 「本文」
\chapter{...}% 章タイトル
\section{...}% 節タイトル
\bibliographystyle{jplain}% 文献スタイルファイル指定
\bibliography{...}% 文献データベース指定
\backmatter% 後付け
\printindex% 索引出力
\end{document}
```

実際には twoside_master.tex を自らコンパイルし，入・出力ファイルをじっくりと比較することではじめて奇数・偶数ページのレイアウトを変える組版のエッセンスが理解できる。以下にいくつかコメントを付す。

\frontmatter コマンドにより「目次」「表目次」「図目次」が自動的に「章（chapter）」レベルで，かつ，奇数ページ起こしで作成される。その際章番号は付かない。表・図目次が不要であれば当該記述箇所をコメントアウトする。雛形ファイルにおいては表・図目次に関しては奇数ページ起こしとしない設定（\let\clear[double]page\relax 命令）を加えてある。\frontmatter 箇所ではローマ数字（小文字）でページ番号が振られる。

\chapter{論文の要旨} 箇所は \frontmatter 内であるから番号なし見出しとなり，目次にもそのように載る。

\mainmatter 以降は「本文」の開始となる。最上位の見出しは \chapter（その下は \section さらにその下は \subsection）であり，章（節・小節，等々）番号は自動的に振られる。\mainmatter 以降の見出しに番号を振らせたくない場合は \chapter*{...} のようにアスタリスクを付け，番号なし見出しを目次にも載せたい場合は \addcontentsline コマンドを追加する。\mainmatter 箇所ではアラビア（算用）数字でページ番号が振られる。

段落は空白行で区切る。脚注は

```
邦文\footnote{脚注。} の場合
a pen\footnote{This is a footnote.}.
```

のように付ける。邦文では \footnote コマンドの後に space を置く。

quote は 1 段落から成る引用に，quotation は複数段落から成る引用に対して用いる。雛形ファイルにおいては双方の環境内に宣言型コマンド \small を書き加え，引用箇所が本文より小さめに出力される（こうすれば引用であることがよりはっきりする）ようにしてある。quote/quotation 環境では「ディスプレイ表示」（各行が適度にインデントされるような表示）される。

まとまった量の外国語を扱う場合は babel パッケージを読み込んだ上で \selectlanguage 指定をする。これは宣言型コマンドであるから Scope（宣言型コマンドの有効範囲）にも注意すること。\selectlanguage 指定下では，処理対象言語が正しく設定されていれば \enquote{…} コマンドで各言語の引用符が正書法に則って正しく出力される（例： „Deutsch“ « Français » “English” ）。

詩に対しては verse 環境を用いる。各詩行には \\ コマンドで改行位置を入力する。

TEX における箇条書きの書式には多々あるが，雛形ファイルでは itemize（番号なし）と enumerate（番号付き）環境のみを使っている。いずれも「入れ子」処理させてあるのでソースを良く見ること。

「参考文献」も「索引」も \addcontentsline コマンドの追記なしで「目次」に載る。目次の「参考文献」には番号は振られない。索引は \backmatter 扱いとなるためである。

第 7 講演習

演習用ファイル： twoside_master.tex, khm.bib, jsbook_lglf.cls, sneewittchen.jpg, sw_leg.jpg

1. TEX ソースを uplatex, upbibtex, upmendex, uplatex（複数回），dvipdfmx でコンパイルせよ
2. 出力 PDF を Acrobat Reader で確認せよ
3. Acrobat Reader 上で M-v p p と打ち込み「見開きページ表示」にせよ
4. 入・出力ファイルを注意深く比較せよ
5. twoside_master.tex ファイルでは hyperref 関連箇所を全てコメントアウトしてあるが，これらのコメントを外し twoside_master_hyp.tex と別名保存した上でコンパイルしてみよ
6. 同様に入・出力ファイルを比較せよ
7. Acrobat Reader では Ctrl + D（Windows），Command + D（Mac）のキー押下で PDF 文書のプロパティを確認できるのでチェックしてみよ

発展： 大規模文書の編集

\input あるいは \include というコマンドを使えば，例えば「論文集」のような大規模文書を小分けにして管理そして編集できるようになる。

具体的には 1 つの親ファイルと複数の子ファイル（いずれも拡張子は *.tex）を用意する。親ファイルには文書クラス指定を始め Preamble の全てを記述し，その上で以下のような書式で読み込む子ファイルを列挙するだけである。

```
\documentclass[a4paper]{article}
% ここに Preamble
\begin{document}
 \input{file1}
 \input{file2}
 \input{file3}
 ...
\end{document}
```

子ファイルの拡張子が *.tex であれば \input コマンドの引数はファイル名だけとする。子ファイルには \section コマンド等で始まる本文箇所のみを書く。

このようにしておけば，読み込む子ファイルの順番を入れ替えるだけで簡単に章・節の並び順を変更できる。Emacs では C-x C-t で前後行の交換ができたことも思い出すと良い（IA 第 3 講，12 ページ）。あるいは特定の子ファイルだけコメントアウトする，つまり読み込まれないようにする，といったことも簡単にできる。

input という箇所を include で置き換えても同様のことが可能であるが，この場合は各子ファイル間で「改ページ」される点が異なる。\include コマンドではコメントアウトする方法以外に Preamble で

```
\includeonly{file1,file3}
```

のように記しても file1.tex と file3.tex だけがコンパイルされるようになる。

発展演習

演習用ファイル： snee3_div.tex, sw_en.tex, sw_de.tex, sw_fr.tex

1. 親ファイル snee3_div.tex と子ファイル sw_en/de/fr.tex ファイルの中身をチェックせよ
2. snee3_div.tex を pdflatex（複数回）でコンパイルし，入・出力ファイルを比較せよ
3. 読み込む子ファイルの順番を入れ替えてコンパイルしてみよ
4. \input を \include に置換した上でコンパイルし，入・出力ファイルを比較せよ

第 8 講

異体字の処理

「同じ文字である」と見なされている複数の字体を「異体字」（Ideographic Variation）と呼ぶ。同じ文字とされるため異体字のうちのどちら（3 つ以上の異体字が存在する場合はどれ）を使っても本来の意味は変わらない。具体的には

> 森鷗外と森鴎外，内田百閒（閒）と内田百間，吉野屋と吉野屋，高島屋と髙島屋，東京都葛飾区と奈良県葛城市，辻と辻，叱ると叱ると吒る，補填と補塡，剥奪と剝奪，頬と頬，崎と﨑と嵜と嵜，永田善久と永田善久

等々が異体字である。

しかし文字コードという点から見た場合，上に挙げた異体字間にもさらなる差異がある。「閒」と「間」(U+9592),「葛」と「葛」(U+845b),「辻」と「辻」(U+8fbb),「叱」と「叱」(U+53f1),「永」と「永」(U+6c38),「善」と「善」(U+5584) には，カッコ内にユニコード 16 進数のコードポイントを記しておいたように，文字コードとしての区別がない。これは「統合」あるいは「包摂」と呼ばれており，例えば A という PC で「葛」の字形しか出力できず，一方 B の PC では「葛」の字形しか持っていないという場合には，それぞれのユーザ間で「見ている実際の字形が異なる」ということが起きる。

「鷗」(U+9dd7) と「鴎」(U+9d0e) …は統合漢字ではない異体字であり，各文字にはそれぞれ一意の文字コードが振られている。

異体字を含む漢字の多字形を使いたいという場合，主に出版・印刷業界で用いられている Adobe-Japan1（2018 年 4 月現在での最新バージョンは Adobe-Japan1-6）という異体字データベースに登録されている規格に準拠したフォントを用いることになる。日本語の文字表記については，異体字も含め，ほとんどの文字がこの Adobe-Japan1-6 の範囲内で実用的に間に合うと思われる。例えば「小塚明朝 Pr6N」・「小塚ゴシック Pr6N」や「ヒラギノ Pr6N フォント」（いずれも有償）が Adobe-Japan1-6 の全 23,058 字[*3] を収録している（Mac に標準バンドルされている「ヒラギノ」は Adobe-Japan1-6 のサブセット）。Adobe-Japan1-6 に完全対応している無償フォントには Noto（Noto Serif/Sans CJK JP） 等がある。これらは OpenType と呼ばれる（従来の PostScript Type 1 形式と TrueType 形式を包含する）新しいフォント形式である。

さて，TEX から Adobe-Japan1-6 範囲の全文字にアクセスするには otf（OpenType Font）パッケージを用いる。Preamble に以下のように記す。

- \usepackage[…]{otf}

オプション部には expert（OpenType フォントで縦組・横組・ルビ専用の仮名字形を使う），deluxe（7 ウェイトとプロポーショナル組を使う），multi（简体字，繁體字，한글を使う），jis2004（jis2004 字形を使う）といった指定が可能。詳しくは参考書籍等を参照すること。

望む漢字がキーボードから通常のインプットメソッドを用いて入力可能であれば何もすることはないが，そうでない場合，Adobe-Japan1 が定める CID（Character IDentifier）番号を用いて当該文字を入力する。例えば \CID{13706} と入力すれば「吉」（「つちよし」）が出力される。その他の具体例については，演習用ファイルを参考にすること。

The Adobe-Japan1 の全字形表と CID 番号は参考書籍の巻末に載っている。インターネットで *The Adobe-Japan1-6 Character Collection* を検索して全字形表と CID 番号が載っている PDF をダウンロードしても良い。otf では \CID{…} の代わりに \UTF{4 桁 16 進数ユニコード番号} と入力することもできる。

発展： 欧文の異体字

漢字のみならず，ラテン文字の長い ſ や小添字 e 式のウムラウトも s やウムラウトの異字体と見なすことができよう（Königsstraße vs. Kǫ̊nigsſtraße）。ドイツ旧字

[*3] 2021 年 8 月 24 日時点での最新バージョンは Adobe-Japan1-7 であるが，これは「令和」合字 2 種（横組み用および縦組み用の 2 グリフが追加された 23,060 字を規定している。）。

体も同様である（Königsstraße）。演習用ファイルにはこれらの入力サンプルも盛り込んでおいた。ドイツ旧字体にイタリック体はない。強調する場合は隔字体（K ö n i g s s t r a ß e）とする。ドイツ旧字体の隔字体では ch ck st tz sz=ß は隔字されないことにも注意（これらは ch ck st tz ß のように合字されたままであるのが正書法）。

なお，長い f は語頭・語中で用いられ，語末・語区切りには丸い s が用いられる。語頭・中・末は TEX に自動認識させられるが，語中の語区切り箇所は無理である。従ってこの場合 "| を挿入することで TEX に語区切り箇所であることを明示する。

小添字 e 式ウムラウトの出力に関し，ラテン文字では Preamble でマクロ（自前のコマンド）を作成し，\uml{a} という入力で å が出力されるようにしてある。マクロなので Å も出せてしまうが，ドイツ旧字体を使う場合はこのような記法とはならず Ae のように A の後に e を送って書くのが正しい。ドイツ旧字体の処理には「仮想フォント」という技術を用いている。しかし，マクロ処理，仮想フォント処理のいずれにおいても「出力ファイル上で正確な文字検索ができない，コピー & ペーストもできない」という制約があり，今となっては obsolete な使用法となっている。こうした制約を乗り越えて小添字 e 式ウムラウトやドイツ旧字体を使いたい場合，TEX の代わりにユニコード・ネイティブエンジンである LuaTEX あるいは XETEX を用いねばならない（ICT IIA 第 13 講，27 ページ参照）。

第 8 講演習

演習用ファイル： iv.tex

1. TEX ソースを uplatex, dvipdfmx でコンパイルせよ
2. iv.tex と iv.pdf を比較せよ
3. ソースの Preamble にある otf のオプション引数から jis2004 を削除し，iv2.tex という別名で保存した上で同様にコンパイルせよ
4. iv.pdf と iv2.pdf を比較し，差異を見出せ
5. {\usefont{encoding}{family}{series}{shape} …} というフォント指定コマンドも参考にせよ

第 9 講

本講からは 3 回に分けて TEX における特殊組版に関する技術をいくつか学ぶ。それぞれに深い内容と広い応用性を持つ技術であるが，講義時間の制約上，演習用ソースを具体的に提示するので，そこから実践的に各種技術を習得していくこと。興味を持った事柄については原典マニュアルを参照する等して，自ら発展的に自学自習する姿勢が望まれる。

2 段組み

1 段組みの文章全体を 2 段組みにするには \documentclass コマンドのオプション引数に twocolumn と追記するだけで良い。

ただし，1 段組み用に作成（最適化）されているはずの図や表の箇所については 1 段組みのままにしておいた方が良いことが多い。2 段組みにすると図は縮小され，表のフォーマットは崩れがちになるからだ。この場合，table や figure 環境を table* や figure* 環境に置き換える（つまり * を付加）だけで図・表箇所を 1 段組みのままとすることができる。

table 環境とは異なる処理を行う longtable ではこの手は使えないので，代わりに多段組み用パッケージ multicol を使う。

多段組み

2 段，3 段，といった多段組みを行いたければ multicol パッケージを用いる。Preamble に次のように記す。

- \usepackage{multicol}

その上で，多段組みとしたい箇所を multicols 環境で括る。具体的には

```
\begin{multicols}{2}
多段組みにしたい箇所
\end{multicols}
```

とすれば良い。引数には正整数値を指定する。2 を指定すれば 2 段組み，3 を指定すれば 3 段組みとなる。10 を指定してもコンパイルは可能であるが，大抵の場合，意味のある出力とはならない。

figure, table, longtable 箇所を除いて multicols 環境で括れば，図や 1 ページ内に収まる表やページをまたぐ表

「以外」の箇所を多段組みにできる。つまり，multicols 環境で括られない箇所は通常の 1 段組みとなる。

行送りの変更

文書全体の行送り（改行幅，行取りとも）を変更したい場合，Preamble に次の指定をする。TEX における行送り（\baselineskip と呼ぶ）のデフォルトは，特定のパッケージを除いて，一般的に 1.2 となっている。行送りを多目に取っておけば，出力ファイルをプリントアウトした際，そこが書き込み用スペースともなる。つまり，欧文テキストの自学・予習用に活用できる。

```
\linespread{正数値}
```

こうすることで行送りの値は「正数値 × 1.2」となる。例えば引数を 1.25 とすれば One-and-a-half-spacing となり，1.67 では Double-spacing となる。もちろん 3 とか 4 といったキリの良い数値を指定しても良い。

この方式では，文書全体，つまり，脚注箇所の行送りまで全て一律に変更されてしまうが，「脚注は除外して行送りを変更」したい場合には，代わりに setspace パッケージを用いる。

- \usepackage{setspace}

と Preamble に記しておき，同じく Preamble に \onehalfspacing や\doublespacing 等と指定する。\setstretch{3} のような指定もできる。この場合は 3 × \baselineskip の行送りとなる。

setspace パッケージでは spacing（正数値の引数を取る），singlespacing, onehalfspacing, doublespacing 環境も使えるので，例えば \maketitle や\tableofcontents コマンドを singlespacing 環境で括っておけば，タイトルや目次部分の行送りは通常のままとすることができる。

行番号

lineno パッケージを使えば各行に自動的に行番号を振ることができる。まず Preamble に次の指定をする。

- \usepackage[オプション]{lineno}

その上で行番号を振りたい箇所を次のように環境に括る。

```
\begin{linenumbers}
...
\end{linenumbers}
```

オプション部に modulo と書いておけば番号は 5 行毎に振られる。\modulolinenumbers[正整数] とすることで「正整数」行毎に振らせることができる。pagewise とすればページ毎に 1 から振り直しとなる。right あるいは left（デフォルト）で番号の出力位置を指定する。switch とすれば「奇数ページで右，偶数ページで左」に行番号が出力される。

本講で扱った各種パッケージの詳しい使用法については

- texdoc multicol
- texdoc lineno

とターミナルで打ち込みマニュアルを参照すること。

第 9 講演習

演習用ファイル： cr_ind_hyp_bib_2col.tex, cr_ind_hyp_bib_multicol.tex, snee3_linespread.tex, snee3_setspace.tex, snee3_lineno.tex

1. cr_ind_hyp_bib_2col.tex を uplatex, upbibtex, upmendex, uplatex（複数回），dvipdfmx で処理せよ
2. 入・出力ファイルを比較せよ
3. cr_ind_hyp_bib_multicol.tex を uplatex, upbibtex, upmendex, uplatex（複数回），dvipdfmx で処理せよ
4. 入・出力ファイルを比較せよ
5. snee3_linespread.tex を pdflatex（複数回）で処理せよ
6. 入・出力ファイルを比較せよ
7. \linespread コマンドの引数の値を色々と変えて処理してみよ
8. \documentclass コマンド部のオプション引数に twocolumn を追記した上でコンパイルしてみよ
9. snee3_setspace.tex を pdflatex（複数回）で処理せよ
10. 入・出力ファイルを比較せよ
11. 行送りをさらに色々と変更してみよ
12. snee3_lineno.tex を pdflatex（複数回）で処理せよ
13. 入・出力ファイルを比較せよ
14. \modulolinenumbers の値を操作してみよ
15. twocolumn で処理してみよ

16. \linespread も活かしてみよ

なお，\columnsep は 2 段組み時の段間スペースを，\columnseprule は段間縦罫の太さを，それぞれ指定するコマンドである。

第 10 講

隔字体，下線，打消し線，ハイライト

論文に使うことはまずないが，soul パッケージを使えばテキストに隔字体（Spacing out），下線（Underlining），打消し線（Overstriking）処理を施すことができる。併せて xcolor パッケージを用いることでハイライト（Highlighting）処理も可能となる。Preamble で次のように指定する。

- \usepackage{soulutf8}
- \usepackage[svgnames]{xcolor}

下線，打消し線，ハイライトに用いる色は

```
\setulcolor{Lime}
\setstcolor{OrangeRed}
\sethlcolor{LavenderBlush}
```

のように指定できる。これらの「色名」は xcolor のオプションに記した svgnames（SVG: Scalable Vector Graphics）で定義されている。他にも dvipsnames や x11names といったオプションが使える。blue, green, yellow, red, cyan, magenta 等々といった基本色であればオプションなしでいつでも使用可能である。詳しくはマニュアルを参照のこと。

以上の準備が整えば

```
\ul{...}
\so{...}
\st{...}
\hl{...}
\textcolor{色指定}{...}
```

とすることで引数箇所に下線を付けたり，隔字体としたり，打消し線を施したり，ハイライト表示できる。最後のコマンドはテキストそのものに色を付ける方法である。

ページをまたぐ領域の強調

framed パッケージによってページをまたぐ領域に枠線，背景色，左マージン部の縦棒線を付けることができる。Preamble に次の指定をしておく。

- \usepackage{framed}
- \colorlet{shadecolor}{色指定}

色指定の箇所は xcolor（とそのオプション）に準じる。svgnames であれば例えば LightCyan 等が使える。

oframed (open framed), shaded, leftbar という各環境に対象としたい領域を括ることによって望む効果が得られる。

イニシャル

イニシャル（Initail, Drop Cap とも。ドイツ語では Initiale，フランス語では Lettrine）とは文書冒頭段落始め等で用いられる通常文字よりも大きくて目立つ大文字のことを言う。

イニシャルを用いるには Preamble に次の指定をする。

- \usepackage{lettrine}
- \renewcommand{\LettrineTextFont}{\upshape}

2 番目のコマンドはイニシャル以降の単語内文字列を通常シェイプに変更するもので，この指定は「フランス語だけ」を扱う場合は不要である。というのはフランス語では単語内でイニシャルに続く文字列は Small Caps で処理することが決まっており，lettrine パッケージのデフォルトではそのようになっているからである。

lettrine では

```
\lettrine[オプション]{V}{or}
```

のようにイニシャル（V）とそれに続く文字列（or）を指定する。lettrine のデフォルトでは 2 行取りのイニシャルとなっているが，オプションを付けることで非常に細かいイニシャル処理が可能となっている。具体的には lines（イニシャルを何行取りにするか），loversize（イニシャルの高さ（大きさ）を指定），slope（改行毎のインデント量を制御），lhang（マージン部へのイニシャルの飛び出し具合を制御），findent（最初のインデント量の指定），nindent（次のインデント量の指定），lraise（大きさは変えずにイニシャルを持ち上げる）等々といった

パラメータを操作するが，詳しくは演習用 TEX ソースやマニュアルを参照すること。

カロリング小文字体，ドイツ旧字体

演習用ファイル lettrine.tex には，西暦 800 から 1200 年頃にかけて神聖ローマ帝国内で用いられていたカロリング小文字体を使って『ヒルデブラントの』（現存するドイツ最古かつ唯一の頭韻英雄詩）の冒頭箇所を組んだサンプルも載せてある。

このような特殊フォント（cmin: Carolingian Minuscule）は

```
{\usefont{T1}{cmin}{m}{n} ...}
```

のように範囲を区切って用いることができる。第 1 引数ではフォントエンコーディング（T1），第 2 引数ではフォントファミリ（cmin），第 3 引数ではシリーズ（m: medium），第 4 引数ではシェイプ（n: normal）をそれぞれ指定する。

cmin を使う際は，通常の行送り（12pt）では寸詰まりとなるので 18pt に拡大してある。

演習用ファイルには，さらに，グーテンベルクによる『42 行聖書』で使われていた活字「B42」に基づいてデザインされたフォントを用いて「ヨハネ福音書」の冒頭を組んだ例（フォントファミリ名は ntgoth）と Breitkopf Fraktur に倣ったフォント等（ntfrakv, ntswabv, yinitas）を用いて組んだ『正しいクラヴィーア奏法に関する試論』（C.P.E. バッハ）をサンプルとして挙げておいた。

ntfrakv と ntswabv では同じ s という入力から語頭・語中では長い ſ が，語末では丸い s がそれぞれ自動的に出力される設定を施してあるが，語中にある s について丸い s を出したい場合は "| を明示的に入力する必要がある。ウムラウトは小添字 e 式の変種ウムラウトとして出力される。

小添字 e 式ではない通常のウムラウトを出したい場合は ntfrakv の代わりに ntfrak，ntswabv の代わりに ntswab とすれば良い。ドイツ旧字体においては必要に応じて自動出力されるハイフン記号にも注目すること。

牛耕式，ロンゴロンゴ

本講最後に，古代ギリシア（ヘレニズム時代以前）の碑文等に用いられていた「牛耕式」（βουστροφηδόν）記法を TEX で処理する方法を紹介する。牛耕式では各行交互に「左から右，右から左」とテキストを読み進める方向が入れ替わる。しかも右から左の場合は「鏡文字」となる。

牛耕式の変種が「ロンゴロンゴ」である。19 世紀にイースター島（ラパ・ヌイ）で発見された文字（記号体系）ロンゴロンゴは今なお未解読であるが，この文字は各行毎に読み進める方向が変わるとともに文字の天地もひっくり返って記載される。牛耕式とロンゴロンゴを使う場合は

- \usepackage{boustrophedon}

を Preamble に記しておき，牛耕式としたいテキストを boustrophedon 環境に括る。同様にロンゴロンゴは rongorongo 環境で括る。いずれも偶数行に現れるテキストを赤色表示させる場合は boustrophedon の代わりに boustrophedon_color パッケージを読み込ませる。なお，boustrophedon は試作の段階を超えないものであるから多段組み等の複雑な処理はできない。

本講で扱った各種パッケージの詳しい使用法については

- texdoc soul
- texdoc soulutf8
- texdoc xcolor
- texdoc framed
- texdoc lettrine
- texdoc lettrine-doc

とターミナルで打ち込みマニュアルを参照すること。

第 10 講演習

演習用ファイル： snee3_soul.tex, lettrine.tex, bou.tex

1. snee3_soul.tex を pdflatex（複数回）でコンパイルせよ
2. 入・出力ファイルを比較せよ
3. lettrine.tex を pdflatex（複数回）でコンパイルせよ
4. 入・出力ファイルを比較せよ
5. lettrine.tex 内の ntfrakv, ntswabv とある箇所をそれぞれ ntfrak, ntswab に書き換えてから，上と同様のことを行え
6. bou.tex を uplatex, dvipdfmx で処理せよ
7. 入・出力ファイルを比較せよ

第 11 講

詩

TEX で詩を扱う際は，原則，詩を verse 環境に括って各詩行末に \\ コマンドを記しておくだけで良いが，もう少し複雑な処理をさせたい場合はデフォルトの verse 環境を拡張する verse パッケージを使うと良い。Preamble に次のように記す。

- \usepackage{verse}

さらに詩の作者を付記するため \attrib コマンドを Preamble に定義しておく。

```
\newcommand{\attrib}[1]{%
 \nopagebreak{\raggedleft\footnotesize #1\par}}
```

準備が整えば，以下のような書式で詩を入力する。具体例として，まず，ゲーテ作バラード「魔法使いの弟子」を挙げる。

```
\poemtitle{Der Zauberlehrling}
\settowidth{\versewidth}{Hat der alte Hexenmeister}
\begin{verse}[\versewidth]
  \poemlines{5}
  Hat der alte Hexenmeister \\
  Sich doch einmal wegbegen! \\
  ...
  Tu ich Wunder auch. \\!
  \indentpattern{111111}
  \begin{patverse}
    \vin
    Walle! walle \\
    Manche Strecke, \\
    ...
    Zu dem Bade sich ergieße. \\!
  \end{patverse}
  ...
\end{verse}
\attrib{Johann Wolfgang Goethe, 1797}
```

\poemtitle には「詩のタイトル」を入れる。\settowidth の第 2 引数には「第 1 詩行」を書き込んでおく。これは TEX に当該詩の自動整形に関する計算をさせるための情報として使われる。詩全体は verse 環境で括るが，上例ではその中にさらに入れ子として patverse 環境をも組み入れている。これは第 2 詩節（呪文部分）を第 1 詩節（地の詩文）とは別整形のインデント処理をさせるためである。

インデントのパターンを記すのが \indentpattern である。第 2 詩節（6 行より成る）の各行を予め設定されている \vin（verse indent）値の「何倍」（例では 1 倍）とするかを指定する。なお，patverse 環境の先頭第 1 行はその指定が効かないため明示的に \vin コマンドを記しておく。\poemlines に正整数値を入れておけば，その数値毎に詩行番号が自動的に振られるようになる。各詩行末には \\ コマンド，各詩節末には \\! コマンドを打つ。

次は「本来は 1 行であるにも拘らず，見かけ上の改行を施す」詩行の処理法である。同じくゲーテの長編小説『ヴィルヘルム・マイスターの修行時代』の中に登場する詩「ミニヨン」を例に取り，当該箇所のみ説明する。

```
...
Die Myrte still und hoch der Lorbeer steht, \\
Kennst du es wohl? \\>[6em] Dahin! Dahin \\
...
```

\\> が見かけ上の改行コマンドであり，改行時のインデント量を [ ] 内で指定する。例では 6em（大文字 M の幅の 6 倍）としてあるが，例えば 20mm のように指定しても良い。

2 行詩用には専用の altverse 環境が用意されている（演習用ファイルでは *Nänie* の箇所を参照）。

最後に \flagverse について解説する。このコマンドは引数部分を左マージン（欄外）に出力するので，例えばシラーの詩「歓喜に寄す」等を組む場合に使える。具体的には演習用ファイル内の記法を参照すること。

なお「歓喜に寄す」では altverse や patverse 環境を適切に使い分けているため，「交替韻」や「抱擁韻」の箇所が視覚的にも分かりやすく組まれていることに着目すること。

対訳

本格的な対訳形式の組版には reledmac と reledpar パッケージを用いる。reledmac という（やや分りにくい）名称は本来 TEX 専用の「編集用マクロ」（EDition MACro）が LATEX でも使えるようになり，それが拡張（Extended）され，その後さらに新しくなった（Renewed）という経緯による。reledpar（対訳組版 PARallel typesetting から）も同様である。

reledmac/reledpar を使えば欧文学術校訂における apparatus criticus（註解，考証，異文等の研究資料）をも高度に処理できるが，本講ではこうした方面での活用法

までは扱わない。

reledmac/reledpar を用いるには Preamble に

- \usepackage{reledmac}
- \usepackage{reledpar}

と記しておく。対訳左・右段の各テキスト幅は全テキスト幅の 0.45 倍となっているが，この値は Preamble に

```
\setlength{\Lcolwidth}{0.46\textwidth}
\setlength{\Rcolwidth}{0.46\textwidth}
```

のように書くことで変更できる。対訳形式の大まかな書式は

```
\begin{pairs}
 \begin{Leftside}
  \begin{otherlanguage}{french}
   \beginnumbering
   \pstart
   ...
   \pend
   \endnumbering
  \end{otherlanguage}
 \end{Leftside}
 \begin{Rightside}
  \begin{otherlanguage}{ngerman}
   \beginnumbering
   \pstart
   ...
   \pend
   \endnumbering
  \end{otherlanguage}
 \end{Rightside}
\end{pairs}
\Columns
```

となっている。一番大きな枠組みが pairs 環境で，その中に Leftside（左段）と Rightside（右段）環境を入れる。それぞれの段には babel における言語指定用環境コマンド otherlanguage も組み込んでおく。そうすればその環境内が対象言語に特化して処理されるようになる。

テキストを正確な対訳形式に組むには，双方の言語における段落数が完全に一致していなくてはならない。そしてそれぞれの段落を \pstart と \pend コマンドで括る（p は paragraph）。

テキストを \beginnumbering と\endnumbering で括っておけば，5 行毎に行番号が振られるようになる。その際，デフォルトでは左段にアラビア（算用）数字，右段にはアラビア数字の横に R（right）が付加されて行番号が出力されるが，右段をローマ数字に変更したい場合は次のように指定する。

```
\linenumberstyleR{Roman}
\setRlineflag{}
```

段上にタイトルを付けたい場合は \eledsection コマンドを使う（自動連番も振られる）。番号なしとするにはコマンド名の後ろに * を付加する。

pairs 環境を抜けた直後に \Columns コマンドを書く。こうすることで当該 2 言語によるテキストが左・右段に，各段落における開始行も自動的に揃って，組まれることとなる。

本講で扱った各種パッケージの詳しい使用法については

- texdoc verse
- texdoc reledmac
- texdoc reledpar

とターミナルで打ち込みマニュアルを参照すること。

第 11 講演習

演習用ファイル： verse.tex, para.tex

1. verse.tex を pdflatex でコンパイルし，入・出力ファイルを比較せよ
2. Mignon の詩において行番号（見かけ上の改行）をチェックせよ
3. verse.tex を twocolumn で処理せよ。その際 4 つの版面パラメータも有効にせよ
4. 入・出力ファイルを比較せよ
5. para.tex を pdflatex（複数回）でコンパイルし，入・出力ファイルを比較せよ

発展： PGF/Ti*k*Z

PGF（Portable Graphics Format）および Ti*k*Z（Ti*k*Z ist *kein* Zeichenprogramm という自己言及型頭字語）は TEX 用の強力な描画パッケージである。PGF はバックエンドの（ユーザから見えないところで各種処理を行う）プログラムとして機能しており，ユーザはフロントエンドとして Ti*k*Z パッケージを用いる。PGF は次講で説明する Beamer の描画の際にも使われている。

Ti*k*Z を使う場合は

- \usepackage{tikz}
- \usetikzlibrary{decorations.text}

と Preamble に記しておく。\usetikzlibrary コマンドで付属のライブラリ（上例はテキストの装飾に関するライブラリ）を読み込むことで，さらに多彩な表現力が得られる。そして

```
\begin{tikzpicture}
 \draw ... ;
 \path ... ;
 ...
\end{tikzpicture}
```

のように tikzpicture 環境の中に各種コマンドを記していく。コマンドの終了は ; で示す。

Ti*k*Z は巨大なパッケージであり，そのマニュアルは 1200 ページ近いボリュームを持つ。従って本講では Ti*k*Z を用いた具体的な TEX ソース

- ドイツ語テキストを正弦波に沿わせる
- 同上のものに色を付ける
- 矩形螺旋状に書かれたモーツァルト父が妻へ宛てた手紙（1772 年 12 月 18 日付け）
- エッシャーのレンガとペンローズの三角形（いわゆる「不可能図形」）
- ハノイの塔（Beamer パッケージによるプレゼンテーション用 PDF）

を示すに留め，後は各受講生の興味に委ねる。

本講で扱った各種パッケージの詳しい使用法については

- texdoc tikz
- texdoc visualtikz

とターミナルで打ち込みマニュアルを参照すること。

発展演習

演習用ファイル：sinus1.tex, sinus2.tex, rect_spiral.tex, escher-brick-penrose-triangle.tex, tower-of-hanoi.tex

1. sinus1.tex, sinus2.tex を pdflatex でコンパイルし，入・出力ファイルを比較せよ
2. rect_spiral.tex を lualatex でコンパイルし，入・出力ファイルを比較せよ
3. escher-brick-penrose-triangle.tex, tower-of-hanoi.tex を pdflatex でコンパイルせよ

発展： pdfcrop

Ti*k*Z で作成した図形（テキストも含む）は，余白部分等の不要箇所を断ち落とし（これをクロップするという），新たな図形として TEX ソースに組み入れることができる。

file.pdf をクロップするにはターミナルで

```
pdfcrop file.pdf
```

と打ち込む。すると同作業ディレクトリ内に file-crop.pdf が生成される。

発展演習

演習用ファイル： sinus1.pdf, sinus2.pdf, rect_spiral.pdf, cr_ind_hyp_bib_tikz.tex

1. sinus1.pdf, sinus2.pdf, rect_spiral.pdf に pdfcrop 処理をかけよ
2. cr_ind_hyp_bib_tikz.tex を uplatex, upbibtex, upmendex, uplatex（複数回），dvipdfmx で処理せよ
3. 入・出力ファイルを比較せよ

なお，演習用ファイル内には €（ユーロ），¢（セント），$（ドル），¥（円），£（リラ），₱（ペソ），₤（ポンド）等々の他にも，marvosym パッケージを使った ℌ（デナリウス），ℭ（プフント），ℬ（シリング），ℱ（フローリン）等といった古い通貨記号の入力法も示しておいた。

通貨記号に限らず TEX では実に様々な文字や記号を使うことができる。*The Comprehensive LATEX Symbol List* にはそれらが網羅的にリストアップされている。

- texdoc comprehensive

でこのマニュアルを参照できる。

第 12 講

Beamer（ドイツ語 Beamer は英語からの借用語，英語では projector）パッケージを使えば，TEX ソースからプレゼンテーション用 PDF を生成できる。本講では Beamer を用いた TEX ソースの具体例を示す。

Org ファイルからプレゼンテーション用ファイルを生成

演習用ファイル beamer0.txt を EURO ICT Emacs で開くと org が major mode となって Emacs が起動する。beamer0.txt のヘッダ部（ここに org からプレゼンテーション用 PDF を生成させるための諸設定が記されている）を除けば，基本書式は（ICT IB 第 6 講で学んだように）

```
* 第 1 階層見出し
** 第 2 階層見出し
   1. 番号付き箇条書き
   2. ...
* 第 1 階層見出し
** 第 2 階層見出し
   - 番号なし箇条書き
   - ...
```

というとてもシンプルなものである。ICT IB 第 11 講で org モードでのエクスポート（TEX や HTML）法を学んだが，実は，EURO ICT Emacs 上では次のコマンドを打つことによりバックエンドで beamer を作動させ，プレゼンテーション用 PDF をも作成できる。

- (org-beamer-mode)
- C-c C-e l P (org-beamer-export-to-pdf)

前者は toggle コマンドとなっている。( ) 内は Emacs のコマンドであるからもちろん M-x を前置してから打ち込まねばならない。出来上がった PDF は全画面表示（Adobe Acrobat Reader であれば C-l）にして閲覧すると良い。

Beamer でプレゼンテーション

beamer0.txt を雛型とすればいつでも簡単にプレゼンテーション用 PDF を作成できる。しかし，実際には，強調表示をしたり，アニメーション効果を付加したり，もう少し見栄えのするプレゼンテーション用 PDF を作成したい場合もある（やり過ぎは禁物！）。この場合は TEX ソースに介入する。

プレゼンテーション用 PDF と言ってもこれまで学んできた TEX の書式に関する知識をそのまま活かせるが，「1 つの画面に収めたいテキストを frame 環境で括る」という点が新しく学ぶ事柄である。beamer で処理する TEX ソースのおおまかな基本書式は

```
\section{...}
\subsection{...}
\begin{frame}\frametitle{...}
...
\end{frame}
```

となっている。frame 環境（フレームにタイトルを付けることもできる）には「1 画面に収まるだけのテキスト量」としなくてはならない（そもそもプレゼンテーション用のスライドに多くの文字を盛り込むのは御法度である）。

1 つのフレーム内には複数枚のスライドを配置する。個々のスライドには itemize や enumerate 環境を使って項目を箇条書きしていくのが一般的な記法となろう。この際

```
\begin{itemize}
 \item<1> ...
 \item<2> ...
 ...
\end{itemize}
```

のように < > 内に数字を付けることでスライドの出現具合を制御できる。上例では第 1 項目は 1 番目のスライドのみに出現し，2 番目のスライドが映し出される際には消えて見えなくなる，ということを表している。<1-> とすれば当該フレーム内 1 番目以降の全スライドで出現するようになる。<2,4> とすれば 2 番目と 4 番目のスライドに現れる。

演習用ファイルでは，この他にも，block/exampleblock 環境，半透明効果（\setbeamercovered{transparent}），ディゾルブ効果（\transdissolve[duration=2]）等を用いている。参考にすること。ディゾルブ効果を用いる際は PDF を全画面表示にする必要がある。なお，Beamer には Bergen, Berlin, Copenhagen, Hannover, … といったヨーロッパ都市名を名前に持つ「テーマ」が複数用意されており，これを書き換える（例：\usetheme{Warsaw}）だけで外見をがらりと変えることもできる。

縮刷ハンドアウトの作成

出来上がったプレゼンテーション用 PDF からは，pdfpages パッケージを使うことで縮刷ハンドアウトを作成することもできる。1 例として演習用ファイルに allslides.tex ソースを用意した。A4 版縦置き 2 段配置（1 段に 4 スライド）で出力する。その中にある \inclu-

depdf コマンドの引数に対象とするプレゼンテーション用 PDF のファイル名を（拡張子まで含めて）書き入れれば良い。

出力したいページ番号はオプション引数部で pages={1-3,4,7,12} のように記す。1-3 は連続ページ番号を表す。全ページは pages=- とすることで出力できるが，スライドの出現具合をコントロールしている場合，全ページ出力はナンセンスである。

本講で扱った各種パッケージの詳しい使用法については

- texdoc beamer
- texdoc pdfpages

とターミナルで打ち込みマニュアルを参照すること。

第 12 講演習

演習用ファイル： beamer0.txt, beamer1.tex, beamer2.tex, allslides.tex

1. beamer0.txt を EURO ICT Emacs で開き，M-x org-beamer-mode と打て
2. 次に C-c C-e l O と打て
3. beamer0.txt と beamer0.pdf を比較せよ
4. beamer1.tex を uplatex（複数回），dvipdfmx で処理し，入・出力ファイルを比較せよ
5. beamer2.tex を uplatex（複数回），dvipdfmx で処理し，入・出力ファイルを比較せよ
6. allslides.tex を pdflatex で処理し，入・出力ファイルを比較せよ

第 13 講

LuaTEX と XETEX

TEX に代わる LuaTEX や XETEX といった新しいユニコード・ネイティブ対応エンジンを使えばユニコードへのフルアクセスが可能となる。使用フォントについてもこれまでの TEX システム内部での標準処理（TFM: TEX Font Metric ファイルを参照しながら文字を配置していく）を必要とせず，Windows や Mac, Linux といった OS のシステムにインストールされている OpenType フォントをそのまま使うことができるので，将来の TEX による組版処理は間違いなくこの方向に進んでいくだろう。

とは言え，現時点では関連プログラムである BIBTEX（参考文献処理用）や MakeIndex（索引処理用）が完全にはユニコード・ネイティブ対応していないため，邦文処理をも含めた TEX 代替システムとして LuaTEX や XETEX に全面的に移行するには，やや，時期尚早と言えるかも知れない。

本講では LuaTEX と XETEX 用のソースファイルをそれぞれ 1 点ずつ具体的に提示することで，テキスト組版に関してこれら次世代の TEX 代替エンジンが備え持つ潜在能力の一端を垣間見てもらうこととする。

まず，第 5 講演習（13 ページ）で扱った「相互参照，索引，ハイパーリンク，参考文献」処理を含むソースファイルを LuaTEX 用に少々書き換えた演習用ファイル lua_cr_ind_hyp_bib.tex の説明をする。ソース内にもコメントを書き入れておいたが，upTEX で処理する場合と異なる点をまとめておくと以下のようになる。

- 文書クラスが jsarticle から ltjsarticle となる
- dvipdfmx は使わない（のでオプション引数には一切使用しない）
- fontenc, inputenc パッケージも使わない
- 代わりに fontspec パッケージを読み込む
- 実際には色々なフォントが使えるが，ユニコード対応の多言語フォント Noto を使う設定としてある
- otf の代わりに luatexja-otf を使う
- pxbabel は使わない（使えない）ので babel のオプション部に japanese を追記する
- （ルビや圏点処理をするための）okumacro の代わりに bxokumacro を使う
- （傍点処理をするための）plext の代わりに lltjext を使う
- \ltjdefcharrange コマンドにより LuaTEX-ja デフォルトで邦文扱いとなっている文字を欧文グループに区分け変更する
- \ltjsetparameter コマンドによりギリシア文字およびキリル文字を丸ごと欧文グループに区分け変更する
- pxjahyper（PDF 邦文「しおり」用）は不要
- 長さ単位の zw（Zenkaku Width）と zh（Zenkaku Height）はそれぞれ \zw と \zh というコマンドとなる
- \selectlanguage コマンドの発行位置を figure や table 環境の「後」に移動してある

特に最後の項目については説明が必要である。本講義では babel を使っての基底語を邦文とする多言語テキスト処理には原則として upTEX エンジンを用いており，その際 pxbabel パッケージを japanese 指定で読み込むことで \selectlanguage コマンドの発行位置に拘らず表や図のキャプションを常に日本語に統一することができていた。しかし pxbabel を使えない LuaTEX エンジンにおいては \selectlanguage コマンドにより切り替えられた言語用のキャプション見出し（例えばフランス語下では「表」の代わりに Table, ドイツ語下では「図」の代わりに Abblidung）が出力されるため，\selectlanguage コマンドをどこに挿入するか良く吟味しなくてはならない。同様に，邦文が基底語であるならば最後の \selectlanguage コマンドの引数は japanese でなくてはならない。

以上の点を踏まえると LuaTEX や XeTEX において babel を使う際は，日本語以外の外国語テキストが「ある程度まとまった量で出現する」箇所を

```
\begin{otherlanguage}{ngerman}% 他に french など
 ...
\end{otherlanguage}
```

のように otherlanguage 環境で括るか，\foreignlanguage{french}{ここにフランス語テキスト} のように処理する方式が現実解としてより相応しいかも知れない。

最後に XeTEX エンジンで処理するソースの 1 例 xet_iv_uml.tex について説明する。Preamble には

```
\documentclass[xelatex]{bxjsarticle}
\usepackage{zxjatype}
\setCJKmainfont{Noto Serif CJK JP}
\setCJKsansfont{Noto Sans CJK JP}
\usepackage[ngerman]{babel}
\usepackage{fontspec,xspace}
\usepackage{noto}
 ...
```

のように記してある。邦文を基底語とする場合，xelatex オプションを付けた bxjsarticle 文書クラス指定と zxjatype パッケージ指定は必須である。上例では和文フォントとして Noto Serif CJK JP（明朝）と Noto Sans CJK JP（ゴシック）を指定しているが，これらのフォント名は「OS に登録されている名前」となっている。noto パッケージにより欧文 Noto フォントが設定される。フォント周りのこうした諸設定は fontspec パッケージを読み込んで行う。babel をはじめ他のパッケージの使用については upTEX の場合とほぼ同様である。

xet_iv_uml.tex では異体字を入力するのに otf パッケージに依らず「Emacs から直接 IVS（Ideographic Variation Sequence: 異体字符号列）を打ち込ん」でいる。LuaTEX や XeTEX はニコード・ネイティブ対応エンジンであるからこそこのような入力法も可能となっている。例えば「葛」を出力するには Emacs から C-x 8 RET 845b C-x 8 RET e0100 と連続して入力しておけば良い。845b が「葛・葛」双方の統合（あるいは包摂）されたユニコード・コードポイントであり，枝番号（異体字セレクタと呼ばれる）e0100 によって「葛（こちらの異体字セレクタは e0101）」ではない「葛」を一意に指定できるという仕組みである。当該文字がきちんと出力されるためには，もちろん，相応のフォントがシステムに備わっていなくてはならない。

IVS はデータベースに登録して（登録されたものをコレクションと呼ぶ）利用することが国際的に定められており，フリーの Noto CJK JP フォントはこのうちの Adobe-Japan1-6 コレクションに対応する全 23,058 字を収録している。人名漢字等 Adobe-Japan1-6 を超える異体字については Moji_Joho コレクションに対応するフリーの IPAmj 明朝フォント（58,862 字）が使える。インターネットで

- IVD_Charts_Adobe-Japan1.pdf
- IVD_Charts_Moji_Joho.pdf

を検索し，当該ファイルをダウンロードすればユニコード・コードポイントをはじめ異体字セレクタ等を正確に知ることができる。

さて，LuaTEX と異なり XeTEX では「スマートフォント技術」も使える。具体的には OpenType 形式等の対応フォントであれば旧 TEX 式の TFM に依らず合字，ペアカーニング等々といった文字組版に関する高度な機能を利用できる。例えば演習用ファイルの xet_iv_uml.tex では fontspec パッケージに備わる \newfontfamily コマンドによって XeTEX ソース内で用いる \frakturls というドイツ旧字体のフォントファミリを定義しているが，オプション部にある LetterSpace=20 という記述からも類推できるように，このフォントファミリが使われる際は隔字体で組まれるように設定してある。しかもドイツ旧字体の組版規則に則り ch, ck, tz, st 箇所では隔字されない（出力におけるラテン字体との違いを比較すること）。

長い ſ（C-x 8 RET 017f）や小添字 e 式ウムラウト（C-x

8 RET 364）も Emacs から直接入力できる。

本講で扱った各種パッケージの詳しい使用法については

- texdoc luatexja-ja
- texdoc xetex
- texdoc bxjscls-manual
- texdoc zxjatype
- texdoc fontspec

とターミナルで打ち込みマニュアルを参照すること。

TEX パッケージのアップデート

EURO ICT TEX の本体部分は TeX Live（国際的に最も普及している最新の TEX ディストリビューション）であり，そこに LG/LF 用に特化させたカスタマイズを施している。TeX Live には膨大な数のパッケージが含まれており，これらはターミナルで以下のコマンドを（管理者権限で）打ち込むことで一括アップデートすることができる。必要に応じて適宜このコマンドを実行すると良い。

```
tlmgr update --self --all
```

例年 3 月には各種パッケージのアップデートが凍結され，翌年の 6 月にリリースされる新 TeX Live に向けての準備が始まる。従ってこの期間は上記のコマンドを打っても TeX Live 2017 is frozen forever and will no longer be updated. This happens in preparation for a new release. というようなメッセージが出て，アップデートが無効となる。

稀に skipping forcibly removed package *package-name* というエラーメッセージを返されることがあるが，この場合は

```
tlmgr install --reinstall package-name
```

とパッケージ名を明示的に打ち込み，当該パッケージを再インストールすること。tlmgr については tlmgr --help コマンドで詳しい使い方を参照できる。

第 13 講演習

演習用ファイル： lua_cr_ind_hyp_bib.tex, xet_iv_uml.tex

1. lua_cr_ind_hyp_bib.tex を lualatex, upbibtex, upmendex, lualatex（複数回）で処理し，入・出力ファイルを比較せよ
2. xet_iv_uml.tex を xelatex で処理し，入・出力ファイルを比較せよ
3. ドイツ旧字体を隔字体で組んだ時の合字（Ligature）部分をチェックせよ

発展： IPAmj 明朝フォント

Adobe-Japan1-6 レベルでの異体字処理については既に学んだ（18 ページ参照）。住民基本台帳や戸籍上の人名に用いられる漢字や変体仮名等 Adobe-Japan1-6 を超えるレベルの異体字を用いる必要があれば，もっと多くの字形を収める Moji_Joho コレクションとこれに対応したフリーの IPAmj 明朝フォント（IPA: Information-technology Promotion Agency 独立行政法人情報処理推進機構）を使うと良い。ただし現時点では明朝体のみリリースされており，ゴシック体は存在しない。

TEX で IPAmj 明朝を使うには ipamjm パッケージを用いるが，IPAmj 明朝フォントも ipamjm パッケージも現時点では TEX Live ディストリビューションに含まれていないので，「TEX と外字」をキーワードにインターネット検索し，そこから「IPAmj 明朝の利用，IPAmj 明朝フォント，ipamjm パッケージ」と辿り必要なファイル等をダウンロードし，TEX システムに組み入れなくてはならない。

ipamjm パッケージを使えば IVD_Charts_Moji_Joho.pdf 内に記されている全字形を \MJMZM{MJ 字形番号} のようにして呼び出すことができる。Preamble を含む具体的な入力法については演習用ファイル ipamjm.tex を参照せよ（講義担当者側でコンパイル済みの PDF も用意した）。

IVD_Charts_Adobe-Japan1.pdf であれ IVD_Charts_Moji_Joho.pdf であれ，これらの文書では 1 つのユニコード・コードポイントに 2 つ以上の異体字が存在する字形のみがリストアップされており，コードポイントが異なる「単体」で存在する異体字は含まれていないことに注意すること。IPAmj 明朝フォントに含まれるこうした単体字形は（異体字セレクタを含む）異体字符号列を Emacs から直接打ち込んでやり（C-x 8 RET *IVS*），LuaTEX あるいは XETEX で処理することで全ての出力が得られる。Preamble を含む具体的な入力法につ

いては演習用ファイル lua_ipamjm.tex を参照せよ。なお, 変体仮名は lua_ipamjm.tex 内で使っている luatexja-fontspec パッケージの守備範囲外であるため，変体仮名を用いる場合，通常は欧文フォント指定に用いる fontspec パッケージの \setmainfont コマンドで IPAmj 明朝指定をすることにも注意。

発展演習

演習用ファイル： ipamjm.tex, ipamjm.pdf, lua_ipamjm.tex, lua_ipamjm.pdf

1. ipamjm.tex を uplatex, dvipdfmx で処理し，入・出力ファイルを比較せよ
2. lua_ipamjm.tex を lualatex で処理し, 入・出力ファイルを比較せよ

第 14 講

本講からは HTML と CSS の学習に入る。HTML (HyperText Markup Language）とは，文字通り，ハイパーテキストを記述するためのマークアップ言語の 1 つで，TEX 同様，地の文に「タグ（tag)」と呼ばれるコマンドを埋め込んでいく形でテキスト形式のソースファイル (*.html という拡張子を持つ）を作成する。HTML ファイルは通常 Google Chrome, Mozilla Firefox, Internet Explorer, Microsoft Edge, Safari 等々といった Web ブラウザで閲覧するために用いられるため，これらで構成される全体は Web ページ（規模の大きなものは Web サイトとも）と呼ばれる。

HTML のタグは と （p: paragraph 段落）といったように開始タグと終了タグ（こちらには / が付く）で括って用いられるのが原則であり，単独で使われるタグは
（br: break 改行）のように書かれる。開始タグと終了タグで括られた一塊を「要素（element)」と呼ぶ。

CSS（Cascading Style Sheets）とはテキストの論理構造と視覚的デザインを分離する目的で導入されたスタイルシートという言語で，主にテキストの表示形式（スタイル，つまり視覚的デザイン）を制御する。現在の Web ページ作成においては，テキストの論理構造に関わる記述は HTML で行い，デザインに関する箇所は CSS の守備範囲とするのが主流となっている。実際には HTML ソースの中にデザインに関する記述を組み込むこともできるが，このように Web ページを作成してしまうと，後から「見栄え」（デザイン）を変更したくなった場合，一つ一つの HTML ソースに手を入れざるを得なくなり，Web ページ全体の改訂や保守に手間と時間がかかる。一方，HTML と CSS を分けて管理しておけば，デザインの変更は基本的に CSS の差し替えだけで済むため Web ページ管理の効率性が高まる。なお CSS は文字通り滝の様に次々と連続させて（cascade)，つまり重ね合わせて，用いることもできる。

HTML/CSS の学習は，畢竟，HTML の記法やタグの使い方，そして CSS のシンタクス（構文規則）を学ぶことに他ならないが，これらの中身は膨大であるため，本講義では主に「多言語テキスト処理」という観点から HTML/CSS を実践的に学んでいくこととする。受講生は各々の興味に従い，参考文献やインターネット上の諸情報等を参照しながら，HTML/CSS を広くそしてまた深く自学自習していくと良い。

なお，HTML/CSS には様々なバージョンがあるが，本講義では現時点で最新の HTML5 と CSS3 を取り扱う。

HTML の基礎

HTML のソースは原則として *.html という拡張子を持つテキストファイルであり，大まかには以下のような書式で記述する。

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <title>...</title>
    <meta charset="utf-8">
    ...
  </head>
<body>
  <h1>...</h1>
  <p>...</p>
  ...
</body>
</html>
```

この書式は，最初にある「HTML5 用 DOCTYPE 宣言」を除いて，全て開始・終了タグのセットを用いて「入れ子」状に構造化されていることに注意。一番外側は html タグである。html タグでは lang という「属性 (attribute)」に ja（日本語）という「値（value)」がセットされており，この Web ページの基底言語は日本語となっていることを示している。

title や meta タグが含まれる head 要素には，Web ブラウザに表示されない「コンピュータに読み取らせる情報」を盛り込む。多言語に対応させるため meta タグ箇所で Web ページの文字コードを UTF-8（ユニコード）に設定している点にも着目せよ。

一方 body 要素は人間とコンピュータの双方が読み取る情報であり，ここに記された内容（コンテンツ）が Web ブラウザに表示される。h1 タグは最上位の見出し（heading）に用いられ，h2 以降 h6 までの見出しタグがある。p タグは段落（paragraph）を括るのに用いる。

演習用ファイル first.html では，これら以外にも，dl（definition list），dt（definition term），b（boldface），a（anchor），dd（definition description），ul（unorderd list），li（list），hr（horizontal rule），q（quotation），img（image），span, i（italic），div（division），address, br（break）といったタグを使っている。これらの意味するところは，英語による原語を参考にしつつ，HTML ソースと出力（Web ブラウザでの見え方）をじっくりと比較することで理解できるだろう。

a タグの name 属性（これは TEX の \label に相当）に値を与えることで，その箇所は href（hyper reference）で # を付けて呼び出せるハイパーリンクとなる。ハイパーリンクは同一テキスト内のみならず，外部リソースに対しても用いることができる。

img タグの src（source）属性における ./jpg/sw_leg.jpg という記法に注意すること。.（dot）は同階層，/ はディレクトリ区切りを意味するので，この記述の全体は「画像ファイル sw_leg.jpg は HTML ソースと同じディレクトリ内にある jpg ディレクトリの中に入っている」ということを表している。.. は 1 階層上を示す記法で ../../abc.png と書けば現在の位置から 2 階層上のディレクトリ内にある abc.png 画像を参照させることができる（このような記法を「相対パス」表記と呼ぶ）。alt（alternate）属性に画像の「テキストによる説明や名前等」を記しておくと，画像が表示されない場合でも代替の情報提供ができる。

同じく img タグ内の align 属性により画像の出力位置を右（right）や左（left）に指定できる。その際テキストは必要に応じて画像に回り込む。width 属性によりブラウザ表示幅（width）を基準としたパーセント表示で画像の大きさを変更できる（height 指定で高さを制御）。style 属性については html ではなく CSS 記法となるので ; で締めることが必須。ここの margin: 0px 0px 0px 10px; という記述は画像とテキストとの間の余白を「上 右 下 左」の順に当該ピクセル値分取る，という意味。

q タグは短めの引用に用いる（長めの引用には blockquote タグ）が，引用箇所は目で見てすぐ分かるようにカッコ（引用符号）で括られて出力される。さらに q 要素は span タグの lang 属性値に応じて処理させることも可能であるから，span を使って適当に言語の範囲指定をしてやれば，基底言語以外のテキスト箇所にも各言語に対応した正しい引用符号を出力させることができる（ただしブラウザ依存。Microsoft Edge や Internet Explorer では fr を指定すると « Français » のように引用符号横のアキまでもが自動挿入されて出力される）。

なお，Emacs では *.html ファイルを開くと major mode が HTML となり，シンタクス・ハイライト機能をはじめ HTML ソースの編集に特化した諸機能を使うことができる。

CSS の基礎

さて，演習用ファイル first.html の第 6 行目にある link タグの箇所はコメントアウトされている（HTML でのコメント箇所は <!-- と --> で括る）。このコメント記号を外して（Emacs では C-x C-; でコメントを操作できたことを思い出すこと）ソースを保存し，改めて first.html を開いてみるとどうなるか（このテストを行う際は，もちろん，必要ファイルが全てしかるべき位置に存在していなくてはならない）。

見栄え（デザイン）ががらりと変わったことを確認できただろうか。コメント記号を削除した以外 HTML ソースには一切手を付けていない。これが CSS の役割である。link タグの href 属性で lglf_base.css ファイルを読み込んでいることに注意せよ。

lglf_base.css もテキストファイルである。Emacs で *.css ファイルを開くと mode line を見ても分かるように CSS major mode に入る。CSS ファイルは

```
body {
    background-color: whitesmoke;
    margin-left: 3%;
    margin-right: 3%;
    font-family: "DejaVu Sans", ..., ;
}
```

のように記述する。body とある箇所は「セレクタ（selecor）」（正確には「要素セレクタ（element selec-

tor)」，セレクタには他にも class selector, id selector, descendant selector 等々の種類がある）と呼ばれ，指定するデザイン効果が及ぶ範囲を定める働きを持つ。background-color のような部分は「プロパティ（property)」と呼ばれ : 記号の後に「値（value)」を記し最後は必ず ; 記号で閉じる。property: value; の 1 組を「宣言（declaration)」といい，{ から } まで含めた部分を「宣言ブロック（declaration block)」と呼ぶ。セレクタと宣言ブロックを合わせた全体は「規則集合（rule-set)」と呼ばれる。

上例では body 要素，つまり Web ブラウザで閲覧した場合の画面全体，の背景色がやや目に優しい white-smoke となり，テキスト描出時の左右マージンには Web ブラウザのウィンドウ枠幅の 3% 分の余白が取られ，欧文フォントとして DejaVu Sans（ユーザのシステムに当該フォントが存在しなければ Arial, それもなければ Consolas, それもなければ何であれ sans-serif 体のフォント)，和文フォントとしてヒラギノ（なければメイリオ，・・・）が用いられるような設定を施してある。

演習用ファイル lglf_base.css では「クラスセレクタ（class selector)」と「疑似クラスセレクタ（pseudo-class selector)」も用いている。クラスとは HTML で class 属性が付けられた特定の要素を対象にスタイルを適用するセレクタで，疑似クラスとは要素が特定の状態にあるときにのみスタイルを適用するセレクタである。以下，具体例で説明する。

```
.red-emph {
    color: red;
}

p:lang(ja) {
    text-indent: 1em;
}
```

クラスセレクタは . とそれに続く任意のクラス名で記述する。上例では新たに red-emph クラスを定義しているが，そこでは color プロパティを使ってテキスト文字色を red に指定している。こうした記述が含まれる CSS ファイルを読み込ませ，HTML で <q class="red-emph">…</q> と記しておけば red-emph クラス指定のある q 要素が赤色で出力されるようになる。.red-emph の代わりに p.red-emph と書けばスタイル適用の範囲が「p 要素で class 属性を与えた場合のみ」に限定される。

疑似クラスセレクタは : の後に疑似クラス名を指定して記述するが，疑似クラス名は link, visited, hover, focus, lang, first-child のように既に決められている。例では p:lang(ja) としてあるので「段落が日本語指定のもとにあれば，その段落冒頭は 1 文字分（1em）だけインデントする」という設定となっている。

要素セレクタとしての見出し（h1, h2, h3, h4）には font-size プロパティを設定しているが，いずれも値を xx-large (2 倍；基準は medium=1 倍)，x-large (3/2 倍)，large（6/5 倍）というように相対サイズ表記としてある。こうしておけばユーザがブラウザ上でテキストを拡大表示させた（つまり medium の大きさが変わった）場合でも，xx-large 等の相対サイズはそれに追従して変更される。絶対サイズ表記であればこのようには行かない。text-align プロパティの値に center を与えれば中央寄せとなる（他に left, right, justify 等が使える。justify は均等割り付けに用いられるが，この場合はさらに text-justify: distribute-all-lines; の追記が必要となる)。line-height プロパティで行高を指定できる。数値だけを記すと基準文字サイズの数値倍となる。

なお，見出しのテキスト文字色と背景色の組み合わせには互いの色を引き立てあう「補色（complementary color)」を使っている。補色はインターネットで検索すればすぐに調べられる。midnightblue のように CSS で名称が定義されていない色は 16 進数表記 (例：#e6e68f;) で書けば良い。これもインターネットで簡単に調べられる。

CSS では文字列を /* と */ で括るとそこがコメントとなる。Emacs ではもちろん C-x C-; という共通のキー操作でコメントを扱える。コメントが 1 行のみであれば，行頭に // を付けておいても良い。

第 14 講演習

演習用ファイル： first.html, css.zip, jpg.zip, png.zip

1. first.html のある同ディレクトリ内に css.zip, jpg.zip, png.zip を展開せよ
2. 各ファイルが正しい位置に格納されたかを確認した上で first.html をブラウザで開け
3. first.html をブラウザで開いたまま，今度は first.html を EURO ICT Emacs で開き，双方を比較せよ
4. first.html の第 6 行のコメントを外し（C-x C-;）て，保存（C-x C-s）せよ

5. first.html をブラウザでリロード（再読み込み）せよ
6. ブラウザ上での見え方はどのように変化したか
7. lglf_base.css に手を入れ，p セレクタに疑似クラス lang を設定し，ドイツ語段落内のテキストの文字色を #ffcc00; にせよ
8. 同様にフランス語段落内のテキスト文字色を #0050a4; にせよ
9. first.html に手を入れ, Sneewittchen (Schneeweißchen) と Blancheneige の文字色を red にせよ（定義済みの red-emph クラス指定をすれば良い）

第 15 講

Emacs の HTML モード

EURO ICT Emacs の HTML モード時における以下の Key Bindings を知っていると HTML ソースを効率良く編集できる。

- 次のタグへジャンプ：C-c <right> (sgml-skip-tag-forward)
- 前のタグへジャンプ： C-c <left> (sgml-skip-tag-backward)
- タグをペアで削除： C-c C-d (sgml-delete-tag)
- h1 タグ挿入： C-c 1 (html-heading-1)
- p タグ挿入： C-c RET (html-paragraph)
- 終了タグ挿入： C-c / (sgml-close-tag)
- ハイパーリンク挿入： C-c C-c h (html-href-anchor)
- ネーム挿入： C-c C-c n (html-name-anchor)
- 属性挿入： C-c C-a (sgml-attribute)
- 番号なし箇条書き作成： C-c C-c u (html-unordered-list)
- 番号付き箇条書き作成：C-c C-c o (html-ordered-list)
- 画像挿入： C-c C-c i (html-image)
- タグ挿入： C-c C-t (sgml-tag)
- 自動ビュー： C-c C-s (html-autoview-mode)
- ä 挿入： C-x 8 " a
- à 挿入： C-x 8 ` a
- C-x 8 を使って入力できる文字の確認： C-x 8 C-h
- HTML/CSS モードの使い方を確認： C-h m (describe-mode)

タグ・ジャンプは開始・終了タグの行き来に用いると便利である。タグのペア削除機能を用いる際は Cursor を開始・終了タグのいずれかの上に置く。h1 タグ挿入コマンドでは 1 の他に 2, 3, 4, 5, 6 も使える。p タグ挿入では開始タグのみ入力されるので併せて終了タグ挿入コマンドも用いること。なお，終了タグはシンタクスに応じた適切なものが自動的に選ばれる。ハイパーリンクや画像挿入では URL を尋ねられるが，内部・外部いずれのリソースを書いても良い（相対パスあるいは絶対パス表記で）。

属性を挿入する場合は Cursor を当該開始タグ右 > の上に置いて作業する。タグ挿入コマンドを使えばあらゆるタグに関し対話的に値等を記入して行ける。自動ビュー機能をオン（C-c C-s は toggle キー）にすると *.html ファイルを C-x C-s で保存する度に OS デフォルトの Web ブラウザが立ち上がり出力を確認することができる（その都度新規タブで開かれることに注意）。

C-x 8 は欧文で用いられる代表的な特殊文字（ウムラウトやアクサン付き文字等々）を入力するための前置キーとなっている。C-x 8 C-h と打ち込んでやれば，この方式で入力可能な文字の一覧を別 buffer で参照できる。一覧 buffer はそこを active にし q を押せば消える。

*.html あるいは *.css の編集の際 C-h m を押してやれば HTML あるいは CSS モードの簡単な使い方を確認できる（ただし CSS モードで覚えるべき Key Bindings は特にない）。EURO ICT Emacs では HTML/CSS 編集時に minor mode として rainbow も有効となるよう設定してある。こうしておくと「色名」（16 進数も含む）のテキスト箇所には色が着き実際の色がすぐ分かるため便利である。

address 要素内ではメールアドレスを掲げているが，ここは意図的に画像としてある。テキストでメールアドレスを記してしまうと Web ボット（Web 上で自動的に情報を収集するプログラムの総称）にメールアドレス情報を取られてしまい後々面倒なことになることが有り得るためである。

コピーライト・マークは `&copy;` で入力してあるが，この記法を「文字実体参照（Character Entity Reference）」方式という。16 進数表記で `&#x00a9;` と書いても良い。こちらは「数値文字参照（Numeric Character Reference）」と呼ぶ。HTML ソースをユニコード（utf-8）で作成していれば，もちろん，© のように直接コピーライト・マークを書いても良い。

Web ページの公開法

Web ページは大まかに以下の作業手順で作成・公開する。

1. ローカル（私用 PC や大学内 PC 等）でコンテンツ（HTML/CSS ファイルや画像等々）を作成
2. それらのハイパーリンク箇所等が正しく機能しているかチェック
3. コンテンツ一式を Web サーバにアップロード

福岡大学内 PC 教室においては H:\public_html（Windows 端末），~/public_html（Mac 端末）が公開するコンテンツの設置場所（指定ディレクトリ）となっている。つまり，この中に Web コンテンツを入れるだけで良い[*4]。

ただし，福岡大学の情報公開システムを使って Web ページを公開するにはオンラインでの申請が必要である。詳しい手続きは『USER'S GUIDE: FUTURE 5』の「第 5 章 情報公開システム」以下で説明されているので，そちらを熟読すること。

全ての手続きが済めば

```
https://www.cis.fukuoka-u.ac.jp/~アカウント名/
```

の URL（Uniform Resource Locator）で世界のどこからでも 24 時間アクセスできるようになる。アカウント名は学籍番号となる。チルダ（~）は半角で入力する（~ は UNIX 系 OS でホームディレクトリを表す）。例えば first.html を閲覧するには

```
https://www.cis.fukuoka-u.ac.jp/~アカウント名/first.html
```

が URL となる。ファイル名が index.html となっていれば「~アカウント名」まで記しておけば良い（index.html まで全部記す必要はない）。

EURO ICT Emacs では *.html ファイルの先頭から 400 行以内に *Last modified:*, *Updated:*, *Aktualisiert:*,「更新日時:」の内のいずれかの文字列（case sensitive であることに注意）が含まれていれば C-x C-s でファイルを保存する度に *2018-04-06 15:30:25 [JST: UTC+9]* という書式の現在時タイムスタンプがコロン以下に自動入力されるよう設定してある（init.el, .emacs.el 内の当該箇所を書き換えれば書式を変更できる）。

第 15 講演習

演習用ファイル： first.html, css.zip, jpg.zip, png.zip

1. 演習用ファイル一式を「情報公開システム」を使って Web 公開せよ
2. インターネット経由で first.html を Web ブラウザで閲覧し，ハイパーリンク等を含め全てが正常に機能していることをチェックせよ
3. first.html を見本として「自分」の Web ページを作成し，練習公開してみよ

ヨーロッパ学 ICT IIB

第 1 講

先ずは「ヨーロッパ学 ICT 課題ファイル置場」から euro_ict2b.zip をダウンロードし，自身のローカル環境（私用 PC や大学内 PC 等）で展開せよ。次に index.html をブラウザで開き，メニューからそれぞれのトピックを扱った Web ページへ飛び，「ヨーロッパ学 ICT IIB サンプル・ウェブ」の全体（単独の Web ページではなく複数のページから成る Web サイトとなっている）をざっと眺めてみよ。そして「ヨーロッパ学 ICT IIB」で学ぶことでどのような Web ページ（Web サイト）を構築できるようになるのかをイメージしてみよ。

ICT IIA の第 14 講（30 ページ）でも言及したが，Web ページを作成するには HTML/CSS の記法やタグの使い方，シンタクス（構文規則）を学ばねばならない。その全体は無尽蔵と言っても良いくらい広範囲に及ぶものであるから，本講義では「ヨーロッパ学 ICT IIB サンプル・ウェブ」で用いている各種記法や技法についてのみ解説するに留める。受講生には HTML/CSS，そしてさらには JavaScript 等のプログラミング言語，に関する発展的な自学自習を期待している。

[*4] このサービスは「情報公開システム」という名称で 2020 年 7 月まで提供されていたが，2020 年 8 月からは「ホームページ公開システム」という名称に変更となり，Web コンテンツも SFTP（SSH File Transfer Protocol）をサポートするファイル転送ソフトウェアを用いてアップロードする方式へと切り替わった。保存可能なデータ容量も 2.0 GB から 1.0 GB へと減少した。詳しくは情報基盤センターの「ホームページ公開システム操作マニュアル（簡易版）」を参照。Web コンテンツ公開に関わる以下の記述内容も，従って，適宜読み替えられたい。

HTML5 における文字コード・言語指定

「ヨーロッパ学 ICT IIB サンプル・ウェブ」の HTML ソースファイル（*.html）では，html 要素の lang 属性の値を `<html lang="ja">` のように全て ja（日本語）に設定してある。この設定により当該 HTML ファイルの基底言語が「日本語」となるが，これをドイツ語，フランス語，英語にしたければ，属性値をそれぞれ de, fr, en とすれば良い。これらは「言語コード」と呼ばれる ISO (International Organization for Standardization) 639 で規定されているアルファベット小文字 2 文字で言語を表すコードである。その他の言語コードを知りたい場合，ISO639 をキーワードにインターネットで検索すること。

親要素の lang 属性は親要素以下の子孫要素に継承される。つまり，lang 属性が明示的に指定されていない子孫要素の言語は親要素の lang 属性で指定された言語と同じになる。最大の親要素である html 要素（Root Element とも）の lang 属性値を ja に設定することの意味を考えよ。

1 つの HTML 文書内に異種言語を混在させるには各要素に個別の lang 属性を指定すれば良い。

ところで body 要素内に記述する要素は「ブロックレベル」と「インライン」のどちらかに属する。ブロックレベルは情報を 1 つの塊としてまとめて取り扱うため，デフォルトで前後が改行される。h[1–6], p, pre, blockquote, ul, ol, dl, table, form, hr, address, div といったものが代表的なブロックレベル要素であり，Web ブラウザで閲覧した場合，ブロックレベル要素同士は原則として横には並ばず，縦置きで表示される。ブロックレベル要素の中にはブロックレベル，インライン，どちらの要素も含めることができる。

これに対しインライン要素同士は横並びで表示される。インラインには a, b, br, em i, img, q, sub, sup, tr, span といった要素がある。インラインの中にインラインを含めることはできるが，インラインの中にブロックレベルを書き込むことは（通常）しない。

「ヨーロッパ学 ICT IIB サンプル・ウェブ」の HTML ソースではブロックレベル（p, table, caption），インラインレベル（tr, span）のいずれの要素でも lang 属性を設定しているので，各 HTML ソースに現れる当該箇所を Web ブラウザ上での出力と対比させながら良く確認すること。

次に，`<meta charset="utf-8">` のように各 HTML ソースの head 要素内 meta 要素の charset 属性で HTML 文書の文字エンコーディングを utf-8（ユニコード）に指定している点にも注意せよ。こうしておけば同一文書内でユニコード文字の直接表記による多言語混在が可能となる。日本語を基底語とする文字エンコーディングには utf-8 の他にも shift_jis, euc-jp, iso-2022-jp が使えるが，これらのエンコーディングにする場合，ユニコード文字の直接表記による多言語混在は不可となる。

文字実体参照・数値文字参照

では，meta 要素の charset 属性を例えば shift_jis にした上で，なおかつドイツ語ウムラウトやフランス語アクサン記号等を混在させたい場合にはどうしたらよいのか。

それには「文字実体参照」(Character Entity Reference) もしくは「数値文字参照」(Numeric Character Reference) という手法を用いれば良い。トップページのメニューから「表とリスト」(tables.html) ページを開いてみよ。ここにいくつか具体例を示しておいた。

これらはそもそも HTML のマークアップに使われる記号（< > & 等々）自体や日本語・英語キーボードからは通常直接入力できない文字（ä á â ç ß 等々）を表記するために使われていた方法である。文字実体参照方式では ASCII (American Standard Code for Information Interchange) 文字しか使用しないので shift_jis, euc-jp, iso-2022-jp のような日本語文字コードと衝突しない。例えば `&Auml;` と入力しておけば Ä と出力される。数値文字参照を使って `&#xc4;` としても同様に Ä が出力される。後者の場合，書式のうち c4 が 16 進数による Ä のユニコード文字番号（Unicode Code Point）である。これを残りの文字列で挟み込めば Ä が出力される。この他にも目的とする文字のユニコード文字番号が知りたければインターネットで検索すれば良い。Emacs であれば文字上に Cursor を置き C-u C-x = と叩けばコードポイントのみならず当該文字に関する詳細な情報が得られる。

SEO 対策： 検索エンジン向けのキーワード指定など

meta 要素には，他にも name 属性と content 属性を使って，主に検索エンジン向けの HTML 文書に関する説明を記述できる。具体例として「トップページ」（index.html）のソースを確認してみよ。

name="keywords" には当該 HTML の中身をキーワードで，content 属性の箇所にキーワード間を , で区切って書いておく。name="description" には当該 HTML の簡潔な説明文を content 属性の値として記す。この箇所は検索エンジンによる検索結果に表示されることも考慮に入れること。name="author" は当該 HTML の作者を示す際に指定する。

meta 要素内に記した各種コンテンツは Web ページの表示画面には現れないが，「検索エンジン向けの（Web ページ）最適化」（SEO: Search Engine Optimization）のためにもこうした記載は重要である。検索結果で自身の Web サイトがより多くリストアップされるように行う一連の取り組みのことを「SEO 対策」と呼ぶが，現在では Web ページを公開する際 SEO 対策も併せて考慮することが当たり前となっている。

第 1 講演習

演習用ファイル： euro_ict2b.zip

1. 演習用ファイル一式をローカル環境で展開し，index.html を Web ブラウザで開き，メニューからそれぞれのトピックを扱ったページを確認せよ
2. *.html ファイルを Emacs で開き，それぞれの lang 属性値を点検せよ（C-s lang= と打ち込んでみよ）
3. 「表とリスト」（tables.html）ページの文字実体参照と数値文字参照の表をブラウザで確認せよ
4. test.html を新規作成し，Emacs で head 要素内 meta 要素の charset 属性を shift_jis にした上で，test.html ファイルの文字コードを Shift_Jis に変更（C-x RET f sjis RET）し，文字実体参照および数値文字参照を用いた欧文特殊文字の入出力をテストせよ
5. *.html の meta 要素内 author 箇所の content 属性を全て自分の名前で書き換えよ（ヒント： Emacs で grep または find-name-dired コマンドを使う）

第 2 講

絶対パスと相対パス

階層型ファイル構造においてファイルやディレクトリの位置を指定する方法には 2 種類あり，最上位であるルートディレクトリからの階層構造（道順）を全て記すことによってこれらの位置を示すやり方を「絶対パス（Absolute Path）」表記，現在位置（カレントディレクトリ）を起点として対象となるファイル等の位置を相対的に指定する方法を「相対パス（Relative Path）」表記と呼ぶ。

HTML における絶対パス表記は，HTML 文書内のリンク箇所で外部サイト（外部の Web サーバ）にあるファイル等を参照する際 `http[s]://` で始まる URL を全て書き切る形で用いられる。

これに対しコンテンツ内部でのリンク指定には，通常，相対パス表記を使う。. が同階層，.. は 1 階層上を示す記法であることは既に説明した（ICT IIA 第 14 講 31 ページ参照）。

例えば「トップページ」（index.html）のソースを見ると，ハイパーリンク箇所（href 属性値）に ./multilang.html とか ./css/all.css のように記述されている様が確認できる。これらが相対パス表記であるが，それぞれの記法から multilang.html は index.html と同階層に，all.css は index.html と同階層にある css ディレクトリの中に存在していることが分かる。css のような新規ディレクトリを適宜作成し，その中に関連するファイルをまとめて格納すれば，Web コンテンツの全体を整理整頓できる。なお，2 階層上の相対パス表記は ../../ となる。

b, i, em, strong, small, hr 要素

`<b>K</b>äfer`, `K<i>ä</i>fer` という入力から **K**äfer, K*ä*fer という出力が得られる。b（Bold）や i（Italic）は他と区別したい文字列を表し，em（Emphasis）や strong は強調したり注意を喚起したい文字列に対して使われる。実際には，多くのブラウザで em 要素はイタリック（斜字体）で strong 要素は太字で表示される。small は文字列を縮小する。hr（Horizontal Rule）は区切りとなる「水平線」を出力する。これは単独で用いられる要素であるから `<hr />` のように書く。

演習用ファイルの *html では small 要素は使っていない。

第 2 講演習

演習用ファイル： euro_ict2b.zip を展開した一式

1. *.html 内の href 属性値を確認せよ（ヒント： Emacs で occur または multi-occur コマンドを使う）
2. index.html 内の b, i タグをそれぞれ strong, em タ

グに書き換え，index2.html として保存せよ
3. index.html と index2.html をブラウザで開き，相互比較せよ

第 3 講

div と span 要素

CSS で視覚的デザインを付与したりレイアウトを行ったりする際は，HTML 内の特定箇所を「一塊の情報の区画」としてグループ化し，このグループを対象に処理を行うのが常である。こうした情報の区画化に使われる要素が div（document Division; ブロックレベル）と span（Span of content; インライン）である。

「トップページ」(index.html) を Emacs で開き，C-s で div と I-search してやると という箇所にマッチする。style は CSS を指定するための専用の属性であるから，ここの値は property: value; のように（HTML ではなく）CSS のシンタクスで書かれていることに注意せよ。本箇所の div 開始タグ上で C-c <right> を打てば，対応する div 終了タグに飛べる（C-c <left> で終了タグから開始タグに戻れる）。さて，最初の div 要素のすぐ下には，また別の div タグで括られた区画があることが分かる。つまり，index.html における div 要素は「入れ子」（Nesting）構造となっており，親 div の中に class 属性値 left, main, right を取る子 div, 子 main, 子 aside が収まっている。入れ子になったこれらの要素は全て「レスポンシブ・ウェブ・デザイン」（Responsive Web Design）に関わるレイアウト処理のための情報区画に用いている。

一方，span は class 属性値 tex，fine, copyleft を取る 3 箇所で用いられており，これらの指定はいずれも「文字（列）出力」に関わる視覚的デザイン付与に関連している。

Responsive Web Design

Responsive Web Design（以下 RWD）とはクライアント（閲覧者）が使っているデバイス（PC やスマートフォン等）の画面サイズあるいは Web ブラウザの幅に応じて Web ページが具合よく表示されることを目的とする Web デザインの手法で，*Content is like water* という言い回しに象徴されるように，「容器」（デバイス）に応じて Web コンテンツが「最適化されて収まる」（表示される）ことを原則とする。

モバイル・トラフィック（スマートフォンやタブレット等からのインターネット接続）がインターネット・トラフィック全体の半分以上を占めるといわれる現在，RWD に基づいた Web ページ作成は益々重要性を帯びている。実際，検索大手の Google は，RWD に対応したモバイル機器に優しい Web サイトが検索結果の上位に表示されるよう検索エンジンに手を加えている（2015 年から．その影響力の大きさから Armageddon をもじって Mobilegeddon と呼ばれる）。

さて，「トップページ」(index.html) における RWD の設定を確認してみよう。style 属性の親 div の中に class 属性値 left, main, right を取る 3 つの子要素が収まっている構造については既に言及したが，まず overflow プロパティの値を auto とすることによって「ボックス」（コンテンツを取り巻く padding, border, margin の各プロパティから成る一塊）の範囲内にコンテンツが収まらない場合，はみ出た部分の表示処理をクライアント（閲覧者）の Web ブラウザに任せている（値を visible とするとはみ出し部分も表示，scroll にするとはみ出し部分についてスクロール・バーを表示，等々）。

RWD の肝となる記述は head 要素内の meta 要素 name 属性値 viewport 他，以下に説明する一連の CSS における設定である。viewport では content 属性値を width=device-width, initial-scale=1.0 とすることによって表示領域の幅をクライアント（閲覧者）の使用デバイスの幅に合わせ，かつ，初期表示を最適化させている。閲覧者は必要に応じて，またデバイスが対応していれば，ピンチアウトによる拡大表示もできる。

class 属性値 left, main, right の中身は，head 要素内の link 要素 href 属性値に記した responsive.css で定義している。responsive.css で行頭が .left で始まる「クラスセレクタ」箇所を参照せよ。background-color, float, width, text-align, padding といった各種プロパティにより，left の class 属性を与えられた要素は「背景色は淡い灰色，左寄せで（十分な表示領域がある限り）横並び，表示幅は画面の 20%，テキストは中央寄せ，パディング（コンテンツ・ボーダー間の余白）は 10 ピクセル」，さらに，.left a というある要素の中の特定の要素だけを対象としてスタイルを適用する「子孫セレクタ」(Descendant Selector）を用いた指定により，left クラスの中で a 要素が使われた場合「背景色は重めの灰色，文字色は薄い灰色，パディングは 8 ピクセル，上マージンは 5 ピクセル，見た目をブロックレベル要素とする，表示幅は

left クラス環境内で 100%，リンクに下線等の装飾は施さない」スタイルで表示される。後で解説する「疑似クラス」:hover により，カーソルがリンクに重なると「背景色が灰色，文字色が黒色」に反転する。なお子孫セレクタは要素「間」に半角スペースを挿入することで表現する。

main および right クラスについてもほぼ同様の記述となっている。これらの float プロパティが全て left となっている点に注意すること。float で left 指定するとその後に続くコンテンツは float 指定された反対側（今の場合は右）に回り込む仕様となっている。つまり，left, main, right クラス指定されたコンテンツは「十分な表示幅がある限り横並び一列で表示される」のである。

スマートフォン等の表示領域幅の狭いデバイスから index.html にアクセスした場合に left, main, right クラス指定のコンテンツが横並びではなく「縦」に並ぶ仕掛けは `@media` 部分の記述による。

`@media` は「メディアクエリ」（Media Query）と呼ばれる条件付きでスタイルを適用する記法で，ここでは「メディアタイプ」（screen）と「メディア特性」（表示領域の最大幅が 620 ピクセル max-width: 620px）を「and 演算子」で結び付け，最終的には「only 演算子」により「クエリ全体が条件を満たす」場合にのみスタイルをあてがう仕組みにしてある。この結果，表示領域幅が 620 ピクセル以下のスマートフォン等からアクセスした場合は，left, main, right クラスセレクタの width プロパティを 100% とする指定により，left, main, right クラス指定のコンテンツは横に並ぶことができず，縦に並ぶことになる。画面表示が切り替わる分岐点を「ブレイクポイント」（Break Point）と呼ぶ。

TEX ロゴ，Copyleft ロゴ，フランス語の二重句読点スペース等

Emacs で responsive.css を開き I-search で .tex と打ち込み TEX ロゴ関連のスタイルを記述した箇所に移動せよ。さらに frame を C-x 2 で 2 分割し，別 window に index.html を読み込んでおくと，以下の説明内容を追試しやすい。

TEX ロゴは，見ての通り，E を少し下げ，字間を詰めて表記する（これができない場合は TeX と書くことになっている）。さて，resonsive.css では .tex sub クラス・子孫セレクタを使って「HTML ソースで tex クラス指定された要素中に sub 要素があれば，これを大文字（uppercase）で表示」し，さらに「この sub 要素は縦方向に 0.5ex 下げ，左を 0.1667em，右を 0.125em 詰めよ」と指定している。CSS におけるこの設定により，HTML（index.html）の当該箇所で `<sub>e</sub>` と記してあっても e は E として出力され，しかも左右に字詰め（Kerning）される。なお，sub/sup は「下付文字」（Subscript），「上付文字」（Superscript）から取った文字列である。responsive.css は HTML ソースの head 要素内 link 要素 rel 属性値 stylesheet 箇所で読み込んでいる。CSS は上から下へ順番に読み込まれていくため，もし重複指定があれば，下の CSS での設定が優先される。

ex, em はそれぞれ「小文字 x の高さ」と「1 文字分の長さ」を表す相対単位である。

Copyleft[*5] ロゴの設定に際しては transform プロパティで scaleX() 関数を用いている。これは指定された要素を水平（X）方向に引数値分だけ拡大・縮小させる関数であり，1（現在値）より大きい値で拡大，小さい値で縮小となる（上下（Y）と前後（Z）方向の比率は 1 のまま）。引数に負数を指定すると左右反転された形で拡大・縮小される。

なお copyleft クラス指定では Vendor Prefix と呼ばれる識別子を前置し，各ブラウザ向けの対応（-moz: Mozilla Firefox; -o: Opera; -webkit: Google Chrome, Safari, -ms: Microsoft Internet Explorer）も施している。これは各ブラウザにおける草案段階の仕様を先行実装する場合，それが拡張機能であることを明示する役割を持つ。将来的には Vendor Prefix 表記が一切不要となるケースも有り得る。Vendor Prefix のない通常のプロパティ指定も当然行っておく。

フランス語の正書法では「二重句読点」（Double Punctuation）！？：；« » の前後に「アキ」（Space）を挿入するが，ここでの改行は禁じられている。HTML では ` `（Non-Breaking SPace）コマンド（数値文字参照表記では ` `）を用いることで，こうした出力を実現できる。しかし，デフォルトでは ` ` のアキがやや広すぎる感無きにしも非ずなので，responsive.css では fine クラス指定によりこのアキを 0.125em と狭くしている。実際に用いる場合は span 要素 class 属性値 fine で ` ` を挟み込めば良い。

[*5] Copyright を捩った造語で「著作権を保持したまま二次的著作物も含めて全ての者が著作物を利用・再配布・改変できなければならない」という考え方。

第 3 講演習

演習用ファイル： euro_ict2b.zip を展開した一式

1.   の出力が fine クラス指定をした場合としない場合でどのように変化するか確かめよ
2. HTML で任意の文字列や記号を左右反転させてみよ
3. responsive.css の中身を良く確認した上で, HTML で LaTeX ロゴと XeLaTeX ロゴを出力せよ（ヒント： LATIN CAPITAL LETTER REVERSED E: Ǝ は Ǝ で入力する）

第 4 講

リスト

「表とリスト」(tables.html) ページでは HTML における表とリストを例示してあるが，ここでは，先ず，リストについて説明する。

番号付きリスト

リストは項目を列挙して示す「箇条書き」に用いる。ol 要素（番号付きリスト Ordered List)，ul 要素（番号なしリスト Unorderd List)，dl 要素（見出し付きリスト Definition List）の 3 種類がある。

ol 要素は

```
<ol>
  <li>...</li>
  <li>...</li>
  <li>...</li>
</ol>
```

のように入力する。リスト内の各項目は li 要素（List Item）で括る。デフォルトでは算用数字で 1 から番号が振られるが，この前置部分は style 属性で CSS の list-style-type プロパティを適用すれば変更できる。

具体的には，値を decimal-leading-zero とすれば「01, 02, 03, …」，upper-roman とすれば「I, II, III, …」，lower-latin とすれば「a, b, c, …」，lower-greek とすれば「α, β, ς, …」，hiragana とすれば「あ，い，う，…」，hiragana-iroha とすれば「い，ろ，は，…」，katakana とすれば「ア，イ，ウ，…」，cjk-ideographic とすれば「一，二，三，…」のように出力される。

1 からではなく任意の番号から採番するには start 属性値を与える。値は整数であれば正負どちらでもよい。

CSS を使えばリスト環境の見栄えを変更できる。responsive.css 内 .ol_var 箇所と .ol_var li 箇所を参照せよ。ol_var クラス指定で「背景色は薄青色，枠線は澄青色の破線，四隅の面取り，左右のマージン調整，上・右・下・左のパディング調整」といった設定を施し，続けて子孫セレクタで「各項目の行高を 1.2 倍，上下・左右のパディングを 0.5em・0」としているので，CSS ソースの記法と HTML 出力とを対照すること。

番号なしリスト

ul 要素の書き方は，先の ol 要素の ol タグを ul に書き換えるだけで良い。項目の前置部分にはデフォルトで「黒丸」(Disc) が付くが，style 属性で CSS の list-style-type プロパティを circle, square, none に設定すれば「白丸，黒四角，前置記号なし」となる。

responsive.css 内 .ul_var および .ul_var li 箇所を見よ。ul_var クラス指定で「ぼかし距離を 3px とする桃色の影でボックスを縁取り，背景色は #fffaf1; 色，上・右・下・左のパディング調整をした上で，左右のマージン調整，上・右・下・左のパディングを個々に行う」といった設定を施し，続けて子孫セレクタで「各項目の行高を 1.5 倍，上下のパディングを 0.5em，前置記号なし」としていることが確認できる。

今度は「::before 疑似要素」を使って前置記号を Emoji にしてみよう。疑似クラスが「要素が特定の状態にあるときにスタイルを適用する」セレクタであったのに対し，疑似要素とは「要素内の特定の文字や行を対象にスタイルを指定する」セレクタである。::before 疑似要素を使えば対象とする要素の直前にコンテンツを挿入できる（直後であれば ::after)。左端からのインデント量も 5em で調節している。😱 部分の出力をどのように指定しているか，CSS の当該箇所を確認すること。::before の content プロパティに & のような HTML で特別な意味を持つ文字を挿入する場合「エスケープ処理」が必要となり，CSS のシンタクスの則った表記をせねばならない。例えば （そこでの改行を禁じる空白スペース）は \a0 と書く。これは当該文字を先ず「HTML の 16 進数による数値文字参照表記」の   に書き換えた上で，「& を \ で置換し # x ; を削除する」という工程となっている。絵文字は Emacs からダイレクトに CSS に書き込めるが，ここでは数値文字参照表記にしておいた。

見出しに自動連番

「表とリスト」(tables.html) ページの h2, h3 要素には，ブラウザでの閲覧時に 1. 1-1. のように「自動的に連番が振られている」ことに気が付いただろうか。HTML ソースの当該箇所に num クラス指定してあることから分かるように，この仕掛けは h[1–4] 要素を対象に CSS で行っている。

responsive.css を見よ。header.num, h2.num, h3.hum の各クラス指定においてそれぞれを「章」(Chapter)，「節」(Section)，「小節」(Subsection) と見立て，各 couter-reset プロパティに chapter, section, subsection という値をカウンタとしてセットしている。こうすることで当該 h 要素のスタート時のカウンタがゼロになる。しかし，さらに ::before 疑似要素クラス指定において counter-increment プロパティに chapter, section, subsection をセットする（h2, h3, h4 にそれぞれ chapter, section, subsection というように 1 つずつずらして設定していることの意味を考えよ）ことで，当該要素が現れる度にセットされたカウンタが一つずつ増加していくため，結果として望む自動連番が振られる準備が整う。

content プロパティに counter(chapter) のようにカウンタ名を指定してやれば，自動連番されたカウンタの値を挿入できる。後は 1. 1-1. のようにカウンタ値を区切るピリオドやハイフンを付加して，自分の望むフォーマットにすれば良い。気を付けることは，これらのピリオドやハイフンを content プロパティの値部分に書き込む際は，これらが文字列であるため CSS の記法規則に従い `".\a0"` のように " でサンドイッチせねばならないということだ。なおスペースには ` ` を用いたが content に記す場合これを `\a0` と表記する必要があることは既に述べた。

第 4 講演習

演習用ファイル： euro_ict2b.zip を展開した一式

1. ol 要素内 style 属性 list-style-type プロパティの値を decimal-leading-zero, upper-roman, lower-latin, lower-greek, hiragana, hiragana-iroha, katakana, cjk-ideographic に変更し，それぞれの出力を点検せよ
2. できれば複数のブラウザでチェックせよ．全てのブラウザで望む出力が得られたか
3. ul 要素内 style 属性 list-style-type プロパティの値を circle, square, none に変更し，それぞれの出力を点検せよ
4. 1. 1-1. 1-1-1. の代わりに「1 章」，「1 章 1 節」，「1 章 1 節（その 1)」という出力を得るには CSS にどのように書けば良いか考えよ

第 5 講

表

HTML の表は

```
<table>
 <caption>...</caption>
 <tr><th>...</th><th>...</th></tr>
 <tr><td>...</td><td>...</td></tr>
 <tr><td>...</td><td>...</td></tr>
</table>
```

といったフォーマットで入力する。table 要素の中で用いている各要素は「キャプション」，「行」(tr: Table Row)，「見出し」(th: Table Header)，「セルの中身」(td: Table Data) を意味する。上のフォーマットは単純な作表の場合の例であるが，複雑な（大規模な）表になると thead, tbody, tfoot といった要素を使って表の中身を行でグループ化して意味を持たせることが多い。グループ化しておけばグループ単位で CSS スタイルを適用できるようになるため，複雑な表の場合，こちらの方が便利であるからだ。

「表とリスト」(tables.html) ページを Emacs とブラウザでそれぞれ開いてみよ。最初に現れる表の HTML ソース箇所は単に `<table>` と始まっているから CSS (今の場合は responsive.css) で表のデフォルトのスタイルを設定していることが分かる。

それでは responsive.css を Emacs で開いてみよう。I-search で table を検索すると，border-collapse: collapse; border-spacing: 0; という記述が見える。これは「隣接するセルの枠線を間を空けずに重ねて表示せよ」(デフォルト値は「離して」表示する separate)，「隣接するセルの枠線間スペースをゼロとせよ」という指定である。前提仕様として各セルが枠線を持つため，この指定により responsive.css を読み込んだ場合の表の枠線は（2 重線とならず）1 本の線となる。

table の caption 要素に対しては子孫セレクタを用い

て font-size: large; padding: 10px 0; という設定にしてある。これにより「キャプション文字列が相対的に大きな文字で，上下に 10 ピクセルのスペースを取って」表示される（キャプションの「中央寄せ」はデフォルト）。文字色が Forestgreen 色となっているのは，HTML ソースで CSS のスタイル設定をピンポイントで行っているため。当該箇所を確認すること。

th 要素は表の見出しとしてある程度目立つように，背景色を藍色とし，セル内上下・左右のパディングをそれぞれ 8px・15px とした。枠線は 1px 幅の実線で Midnightblue 色である。見出しを除いたセルの中身である td 要素のパディングと枠線の設定も th と同じである。td に数字が入る場合は「右揃え」となるよう，クラスセレクタ number を使い text-align: right; 指定した。

HTML ソースの中で pre 要素（PREformatted Text）を使っているが，HTML で特別の意味を持つ & 文字はそのままでは使えないので `&` と文字実体参照表記にしてある。

2 番目に現れる表は「中央寄せ」となっているが，暫定的にこの表のみに対してスタイルを適用しているから，CSS ファイルにではなく HTML の table 要素の syle 属性に margin-left: auto; margin-right: auto; という値を設定している。前者だけ記すと右寄せ，後者だけだと左寄せ，双方記すと中央寄せとなる。caption 要素にも CSS スタイルを当てているのでソースの書き方をよく読むこと。

さて，2 番目の表では複数行ぶち抜きの箇所があるが，ここは td 要素に rowspan 属性を与えて処理をしている。7 段まとめるのであれば，セットする値は 7 となる。同時にまとめたセル内の文字列を「縦書き」としたい場合，CSS で width: 1em; word-break: break-all; という設定を加える。「1 文字ずつ全て改行させよ」という意味である。

第 5 講演習

演習用ファイル： euro_ict2b.zip を展開した一式

1. © ® ¥ ¿ Œ ñ の文字実体参照・数値文字参照表記を調べよ
2. 上の各文字を HTML ファイル内に記し，ブラウザで出力を確認せよ
3. 表を右寄せ，左寄せにせよ
4. `width: 1em; word-break: break-all;` の記述をスタイル属性値から削除すると出力はどのようになるか

第 6 講

子孫セレクタ，複数セレクタ，lang 疑似クラス

引き続き tables.html の 3 番目以降の表を見よ。ここからは「独・日・仏・英」の 4 言語ごとに異なる色付け（それぞれ緑，青，ピンク，オレンジ系統）の作表を行うロジックを組んである。さらに 1 行おきに背景色を付け，行区切りを見やすくしている。こうした事柄を実現するために「子孫セレクタ，複数セレクタ，:lang() 疑似クラス，隣接セレクタ，:nth-child() 疑似クラス」といった手法を用いている。左端カラム（列）の「数字」（KHM 番号）が「右寄せ」となっている点にも注意すること。

「ヨーロッパ学 ICT IIB サンプル・ウェブ」の HTML ソースファイル（*.html）では，html 要素の lang 属性の値を `<html lang="ja">` のように全て ja（日本語）に設定してあったことを思い出して欲しい。lang は全ての HTML 要素で使用できる属性の一つで「グローバル属性」（Global Attribute）と呼ばれる。これが html に対して設定してある，ということは（HTML 文書内の一部の要素に対してではなく）「文書全体の言語指定」をしていることに他ならない。つまり html 要素に lang 属性値がセットされた HTML 文書内では，そのセットされた言語が当該文書の基底言語となる。同文書内の別要素中で別の言語属性値が与えられない限り，この基底言語の拘束力はずっと有効である。

さて，3 番目の表は『グリム童話集』（*Kinder- und Hausmärchen*）における 10 話の「ドイツ語原題」を KHM 番号とともに挙げたものだが，`<table lang="de">` と HTML ソースにあるように table 要素に :lang() 疑似クラスを設定している。設定内容については responsive.css の `table:lang(de)` という箇所を確認せよ。同様に table 要素の子孫セレクタ td についても同じ疑似クラス設定をし「table の大枠および td セルの外周に 1px 幅の実線の枠線を forestgreen 色で描画」するよう指定している。複数のセレクタにまとめて同じスタイルを適用する場合は，間を「,」で区切ってセレクタを並べて書けば良い。子孫セレクタの場合はセレクタ間を半角スペースで区切ることは既に述べた（38 ページ参照）。

表の見出しとしての th 要素にも lang による指定をかけ，td 要素と区別するために，背景色を seagreen とし，

セル内上下・左右のパディングをそれぞれ 8px・15px とした。枠線は 1px 幅の実線で darkseagreen 色である。見出しを除いたセルの中身である td 要素のパディングと枠線の設定も th と同じである。td には枠線以外の lang 設定をしていないので，先に table 要素内クラスセレクタ number で与えた「右揃え」指定が有効のままである。

table 要素の子孫セレクタ caption に対する :lang 疑似クラス指定は「文字色を green 色にし太字にする」というシンプルなものであるから説明の必要はあるまい。CSS の当該箇所を確認すること。ただし，ここでも ::before 疑似要素を使って表キャプションの前に「国旗絵文字」を挿入している。国旗絵文字は国際標準化機構が割り振った 2 文字の「国コード」リスト（ISO-3166）を元に REGIONAL INDICATOR SYMBOL LETTER と呼ばれる A（U+1f1e6）から Z（U+1f1ff）の文字（一種のアルファベット絵文字）を組み合わせて表現するが，クライアント（閲覧者）のデバイスが対応していなければ望む「国旗絵文字」とはならず「アルファベット絵文字 2 つが並んで表示」される。1f1ff 等は 16 進数表記であるから CSS の content プロパティの値として書き込む際には `\1f1e6\1f1f9\a0`（オーストリアの国旗）のようにすれば良いことは既に述べた（39 ページ参照．最後の ` ` の置換記法についても同様）。

隣接セレクタ，nth-child 疑似クラス

3 番目以降の表では見やすく 1 行おきに背景色が付いているが，こうした効果は「隣接セレクタ」と「:nth-child() 疑似クラス」を使えば容易に実現できる。HTML ソース内 tr 要素の style 属性に background-color プロパティを使って 1 行おきに 1 つずつ色の値を書き込んでいっても同じ出力が得られるが，この方式は明らかに煩瑣であろう。CSS でスタイル適用すれば一度の記述で済む。

`table tr:lang(de)+tr:nth-child(2n+1)` で始まるセレクタの指定箇所を見よ。隣接セレクタとは「ある要素の直後に隣接している要素を対象にスタイルを適用する」セレクタで，隣接するセレクタ同士を「+」で結んで書く。この箇所では「table 要素の子孫セレクタ tr 要素の疑似クラスが de（ドイツ語）であって次の tr 要素が奇数番目の行であれば，背景色を lightgreen にせよ」という指定をしている。偶数行ごとにしたければ nth-child の引数を 2n としてやれば良い。偶数，奇数はそれぞれ odd, even という引数でも可。

4, 5, 6 番目の表に対するスタイル適用も，色の選択以外は 3 番目の表に対する場合と全く同じである。後は CSS ソースの当該箇所を各自良く調べておくこと。

なお responsive.css では th, td 要素の枠線を実線で描画することをグローバルに指定してしまっているため，このスタイルファイルを読み込めば全ての表の th, td に枠線が引かれることとなる。例えば td 要素においてこの挙動を抑制したい場合，table 要素に noborder クラスを設定し，子孫セレクタを使って td の border プロパティに none 値をセットすれば良い。responsive.css の当該箇所を確認せよ。「邦文 Web フォント」（ivs.html）の表の箇所で実際この noborder クラスを使っているので，こちらもチェックせよ。

最後にややトリッキーな処理について説明する。tables.html で responsive.css を読み込み，html 要素の lang 属性をグローバルに ja に設定しているのであれば，table 要素に :lang(ja) 疑似クラスを適用してしまうと「全ての表がこの :lang(ja) 指定の影響を受ける」はずである。つまり tables.html の 1, 2 番目の表も同じスタイルが適用された出力とならざるを得ないが，実際にはそうなっていない。なぜか。

responsive.css の「日本語」の表を対象とした記述箇所を参照せよ。そこでの lang 疑似クラスの引数は（実際には存在しない）「ja2」という言語コードに設定してある。これにより html 要素におけるグローバルな ja 指定と衝突せず，表のスタイルを使い分けられることになる。ただし，本来であればこのようなことはすべきではないだろう。表であれ何であれ HTML 内の要素に対し「言語を対象に特別のスタイルを設定する」というロジックを組むのであれば，同じ言語でありながら別々のスタイルが宛がわれるという不整合は望ましくないし，混乱の元となる。tables.html では「表に様々なスタイルを適用する具体例を示す」ために敢えてこのようなトリッキーな手を使った次第である。

第 6 講演習

演習用ファイル： euro_ict2b.zip を展開した一式

1. responsive.css 内で言語が ja2, fr, en における lang 疑似クラスの設定を確認せよ
2. responsive.css の当該箇所に手を入れ，第 1, 4, 7, 10 行目に背景色が出力されるよう設定せよ

3. tables.html と responsive.css ソース内で ja2 とある箇所を ja と書き換えると，tables.html にある 1, 2 番目の表はどのように出力されるようになるか
4. 上の書き換えにより tables.html の他に表の出力が変わるページはどれか

第 7 講

段落，文字，見出しの制御

ICT IIA の第 14, 15 講で用いた lglf_base.css の説明の際に既に言及した（32 ページ参照）が，responsive.css においても「lang 属性が ja（日本語）のもとにある段落冒頭は 1 文字（1em）分だけインデントする」設定をしている。`p:lang(ja)` で始まるエントリがそれである。em は親要素の文字サイズを基準として n 文字分というように長さを相対的に指定できる単位である（元来は大文字 M の横幅に由来する名称．参考 en-dash, em-dash）。:lang(ja) 疑似クラスをこのように用いることで，日本語テキストの段落を 1 字下げしたい場合，各 p 要素に 1 つずつ style 属性を与えてそこに CSS のスタイルを逐一設定する煩わしさから解放される。もっとも HTML の日本語テキストの段落を 1 字下げ必要があるかどうかは別問題である。ブロックレベル要素の p で段落が括られていれば，それだけで p は「一塊の独立した要素」として表示されるため，段落冒頭をわざわざ 1 文字下げせずとも段落間の区切りは一目瞭然である。

テキストの特定部分にピンポイントでスタイルを適用したいときは div（ブロックレベル）や span（インライン）要素で括り，それらの style 属性に CSS のプロパティを書き込んでいけば良い。

ここでは全ての要素の下で使用できるグローバル属性を持つ em（強勢）や strong（強調）要素を再定義する方法を紹介する。div や span が「意味内容に関係なく区分したい領域を指定」するのに対し，em や strong は「意味内容と関連してそこを強勢・強調する」という含みがあることに注意。多くのブラウザのデフォルトでの em, strong は i（イタリック体），b（太字体）の出力と何ら変わるところがないが，i, b 要素も em, strong とは異なり「意味内容と関わりなく単にそこをイタリック体，太字体にする」という意味しか持たない要素である。

さて，今，em, strong を「色付き」で強勢，強調したくなったとしよう。いずれの場合でも CSS で各要素に color クラスを新規に設置し，color プロパティで望む色をセットしておけば，HTML の em, strong 要素のクラス属性値に color を指定するだけで望む効果が得られるようになる。後から色を変更したくなったときでも，CSS 内の color プロパティ値を書き換えるだけで済む。色ではなく「フォント・サイズを大きくする」というスタイルにしたい場合は，bigger クラスを作成し，font-size プロパティに large 等とセットすれば良い。そうすれば em, strong 要素のクラス属性値に bigger を指定することで望む効果が得られる。

スタイル（デザイン）の変更を CSS で制御する利点はここにある。しかも HTML からは `<strong class="color bigger">`（色付き，かつ，フォント・サイズも大きく）のようにスペースで各クラスを区切ることで，複数のスタイルを重ねて宛がうことができる。

「見出し」（h1, h2, …）要素は，例えば次のようにしてスタイルを設定する。

```
h3 {
    color: #ffff74;
    background-color: darkblue;
    font-size: large;
    padding-left: 0.5em;
    padding-right: 0.5em;
}
```

responsive.css では h1, h2, h3 の順に font-size を xx-large, x-large, large のように相対値で指定している。text-align プロパティを使えば位置の調整もできる。背景色を使う場合は padding（あるいは padding-left のように個々に）プロパティを使って，文字を覆う背景色部分の範囲を調整すると良い。line-height プロパティで行高を変えると，背景色の高さもそれに追従する。responsive.css では見出しに自動連番を振る仕掛けをしていることは既に述べた。

邦文 Web フォント

「邦文 Web フォント」（ivs.html）ページは，Web サーバ（コンテンツ提供者）側に特殊あるいは専門的な文字フォントをデータとして用意しておき，クライアント（閲覧者）側のデバイスにこうしたフォントがない場合でも，このフォントデータを使うことで Web ブラウザがこれらの特殊フォントを表示（Rendering）できる仕組みを用いて作成している。こうした仕組みを「Web

フォント」(Web Typography) と呼ぶ。Web フォントでは eot, ttf, woff, woff2 という 4 つのフォント形式が用いられるが，現時点で新旧全てのブラウザで動作する単一の普遍的な形式は存在しない。4 つの内で最も幅広くサポートされているのは woff である。

ivs.html では，Google によって開発が進められているオープンソースの巨大なフォントファミリである Noto と，独立行政法人情報処理推進機構によって配布されている IPAmj 明朝 の 2 種類を Web フォントとして用いている（双方ともフリー）。いずれも極めて多くの字体 (Glyph) を持っており，コンピュータで用いる「文字」についての深い議論とも関わるので，この 2 種のフォントの詳細については「No more Tofu!」(no_more_tofu.html) ページを参照されたい。

ivs.html で用いている Noto フォントは，「Google Fonts 早期アクセス」サービスを利用し，Google のサーバからレンダリングされるように CSS で設定している。noto.css の `@import` で始まる記述箇所を見よ。これは他のスタイル規則を読み込む記法で，ここでは notosans-jp.css（Noto 日本語），notosanslinearb.css（Noto 線文字 B; このフォントは ivs.html ではなく no_more_tofu.html の中で使っている），notosanstc.css（Noto 中国語繁體字), notosanssc.css (Noto 中国語简体字), notosanskr.css (Noto 朝鮮語한글）の 5 つのフォント（CJK のいずれもゴシック体）を Google のフォント・サーバから読み込ませている（Google 側の各 CSS で設定されている)。こうした上で

```
.noto {
   font-family: "Noto Sans", "Noto Sans CJK JP";
}
```

といった noto クラスの設定を noto.css に記しておき，後は HTML ソースで任意の要素に noto のクラス指定をすれば Noto フォント（ゴシック体）を読み込めるようになる。

IPAmj 明朝 フォントに関しては Google のようなサービスが提供されていないため，2018 年 1 月にリリースされたオリジナルのフォント ipamjm.ttf（44.3MB）を ipamjm.woff/woff2（29.6MB/21.9MB; woff: Web Open Font Format）に変換したものを自前で用意し，自前のサーバに置く必要がある。noto.css 内の

```
@font-face {
  font-family: "ipamjm";
  src: url("ipamjm.woff2") format('woff2'),
       url("ipamjm.woff")  format('woff'),
       url("ipamjm.ttf")   format('truetype');
  font-display: swap;
}
```

という箇所が IPAmj 明朝 を Web フォントとして用いるための設定である。url 箇所の記法から分かるように，*.woff[2], *.ttf は noto.css と同階層（つまり css ディレクトリ内）に置いてある。相対パス表記を使ってこれらのフォントを別ディレクトリに分けて設置することも可能である。ipamjm クラスの設定を CSS で行い，これを HTML で読み込む方法は Noto の場合と全く同様である。

以上のような手続きを経れば，Noto や IPAmj 明朝 に限らず自分の好きなフォントを Web フォントとして用いることが原理的には可能となる。しかし，実際の使用にあたっては「ファイルサイズの大きなフォントデータのダウンロードをバックグラウンドでクライアント（閲覧者）に強制している」という事実を十分に考慮に入れねばならない。IPAmj 明朝 ほどのデータサイズを持つフォントであれば，Web フォントとしての実際の使用は，少なくとも現時点では，控えるべきである（本講ではあくまで「サンプル」として紹介している)。例えば，Google Chrome ブラウザではファイルサイズの大きな Web フォントは表示されないので，IPAmj 明朝 は全く表示されない（部分的に表示されているものは代替フォント)。

異体字，縦書き，右から左書き

TEX における異体字の処理については ICT IIA の第 8 講（Adobe-Japan1-6 に対応する全 23,058 字）および第 13 講（Adobe-Japan1-6 を超える 58,862 字に対応している Moji_Joho）で取り扱った (28 ページを参照)。HTML でもこうした異体字を用いたい場合，クライアント（閲覧者）のシステムにこうした特殊な当該文字を含むフォントがインストールされていることは前提できないため，Web フォントという技術を用いる次第である。

HTML ソース内で異体字を記すには「元字」と「字形選択子」(VS: Variation Selector）を併記する。例えば「葛」の異体字である「葛」であれば「葛󠄀」となる。元字部分は「葛」のように「16 進数による数値文字参照」表記でも良い。

元字や字形選択子（異体字セレクタとも呼ばれる）の 16 進数コードを調べるには，インターネットで IVD_Charts_Adobe-Japan1.pdf や IVD_Charts_Moji_Joho.pdf を検索して，各々のコレクションで定義されている全文字に関するデータ（ユニコード・コードポイントや字形選択子等）が載ったこれらのファイルをダウンロードして参照すれば良い。

邦文の「縦書き」については，あまり需要はないだろうが，responsive.css のクラスセレクタで以下のように設定している。

```
.tategaki {
    -webkit-writing-mode: vertical-rl;
    -moz-writing-mode: vertical-rl;
    -ms-writing-mode: tb-rl;
    writing-mode: vertical-rl;
    display: inline-block;
    height: 15em;
    text-align: left;
}
```

見ての通り writing-mode プロパティに vertical-rl 値をセットするが，Vendor Prefix を前置し各ブラウザ向けの対応を施していることからも分かるように，現時点で各ブラウザがこの機能に完全対応しているとは言い難い。HTML での使用法については ivs.html の当該箇所を参照すること。

他にも HTML の表示に OpenType フォントを使うという前提の下で，細かな「字詰め」（自動カーニング）を制御できる font-feature-settings プロパティも（CSS3 から）実装されており，例えば「プロポーショナル・メトリクス」（palt; 約物の空きを調整）や「プロポーショナル・かな」（pkna; かなを詰めて組む）対応で処理するための値をセットできるようになっている。しかしこうした機能もブラウザに依存するので，responsive.css ではクラスセレクタで palt, pkna を使えるように設定のみ行い，HTML でこれを呼び出すことはしていない。

「縦書き」箇所の位置を変える（右寄せ，中央寄せ，左寄せ）には，縦書き箇所をさらに tbox クラス指定した div 要素で括り，responsive.css の tbox クラスセレクタ内 text-align プロパティの値を right, left, center 等にしてやれば良い（初期値は right）。

「右から左書き」（right to left）のテキスト（例えばヘブライ語）を扱う場合は，対象箇所を div 要素で括り dir（Direction）属性に rtl 値をセットする。ivs.html の『旧約聖書』創世記の箇所を参照せよ。

第 7 講演習

演習用ファイル： euro_ict2b.zip を展開した一式

1. multilang.html 冒頭段落の第 1 文を「crimson 色，font-size を large」にせよ
2. em, strong 要素のカラーを purple にし，これを HTML から呼び出せ
3. responsive.css での h1, h2, h3, h4 要素へのスタイル設定を確認せよ
4. Noto および IPAmj 明朝 の Web フォントを使って「善」の異体字を全て HTML で出力せよ（Noto では 2 文字，IPAmj 明朝 では 5 文字）
5. ivs_palt.html をブラウザで開き，palt と pkna を当てた出力とそうでない普通の出力を比較し，どこがどのように異なっているかチェックせよ
6. ivs.html 内で RTL テキストの処理法を確認せよ

第 8 講

ナビゲーション

単一の Web ページではなく複数の Web ページから成る Web サイトには，通常「ナビゲーション」（Navigation）を設置する。これはサイト内にある Web ページ同士をリンクで繋ぐ一種の目次であるが，クライアントにサイト内のあちこちを快適に閲覧してもらうための重要な役割を担っている。

HTML 内のナビゲーション箇所は nav 要素で括り，その中はよく ul や ol 要素で構成される。本講義で用いる *.html では，しかし，CSS で a 要素にスタイルを宛がうことで RWD 対応のナビゲーションとしている。index.html のナビゲーション（メニュー）については既に説明したので，以下はトップページ以外の各 HTML の上部に設置してあるナビゲーションについて解説する。

各 Web ページ上部にあるナビゲーションのスタイルは responsive.css 内の topnav クラスセレクタで始まる箇所以降で設定している。HTML では各ページのタイトル文字列にリンクを張り，特にトップページには「家マーク・アイコン」を添えることで他ページとの区別を図っている。このアイコンは画像ではなく Font Awesome 社が提供している無償の「アイコンフォント」を利用している。これはアイコンでありながら文字でもある

ので，大きさや色等を自由に変更できるという利点がある。本講義の HTML ではその他にも Font Awesome のスタイルを活用しており，そのため同社が提供している all.css と font-awesome-animation.css も読み込んでいる。

さて，上部ナビゲーションを持つ multilang.html には以下のような箇所がある。

```
<nav>
<div class="topnav" id="myTopnav">
  <a href=...><i class=...></i>...</a>
  <a href=...>...</a>
   ...
  <a href="javascript:void(0)" ...
  onclick="myFunction()"><i class=...></i></a>
</div>
</nav>
<script>
function myFunction() {
    ...
}
</script>
```

全体を nav 要素で括ったこの箇所が RWD 対応の上部ナビゲーションに関わる HTML 内での記述である。この中にはこれまで言及してこなかった script 要素部分も含まれる。ここは JavaScript の命令を書き込んでいる部分で，このプログラミング言語については後の講義で取り扱うので，今の段階ではここに JavaScript による（Web ページに動的な効果を付与するための）命令が書き込まれているという点にのみ注意して欲しい。

上部ナビゲーション処理は以下の手法による。先ず HTML でクラス属性値 topnav の付いた要素は，CSS の topnav クラスセレクタで「背景色は `#333333`，内容がボックスに収まらない場合のはみ出た部分は非表示」，およびその下の a 要素は子孫セレクタで「ブロック表示で左寄せ，文字色は `#f2f2f2` で大きさは 17px，テキストは中央寄せでリンクの下線を付けない，上下 14px 左右 16px のパディング」という具合に指定している。トップページのリンク（ボックス）には特別に active クラスを宛がい「背景色 `darkblue`，文字色 `white`」として通常リンク（ボックス）と区別する。

次に，リンク箇所である a 要素にカーソルなどが乗った際の挙動を a:hover 疑似クラスにより「背景色 `#dddddd`，文字色 `black`」とする。icon クラス指定された a 要素は非表示とし，HTML では href 属性値 `javascript:void(0)` を使って a 要素箇所をクリックしても別ページに飛ぶことを抑制する記述を加え，代わりに onclick 属性値に `myFunction()` を与えることで JavaScript プログラムを呼び出す。そして i 要素で（ブラウザの表示幅が 620px 以下となった場合に初めて姿を現す）Font Awesome の「3 本線アイコン」（ハンバーガー・ボタンとも）を「大き目に，ブルブルと震えるアニメーションを付けて」表示させている。Font Awesome 関連のスタイルは all.css と font-awesome-animation.css で読み込んでいる。

CSS におけるメディアクエリのブレイクポイント指定により，画面表示領域幅が 620px 以下の場合 a 要素は最初の子要素（家マーク・アイコンの付いたトップページ）を除いて全て非表示となり，3 本線アイコンがブロック表示の右寄せで出現する。このアイコンがクリックされると `myFunction()` に処理が渡される。すると HTML で id 属性値が myTopnav となっているブロックを DOM（Document Object Model; 後出）で呼び出し，これを変数 x に代入し，ブロック内でのクラス名が topnav であればさらに responsive というクラス名を追記し（これは常に「真（True）」であるから 3 本線アイコンをクリックされたナビゲーション内の a 要素には必ず responsive クラス属性値が追記される），クリックされない場合は topnav クラスのままとする。これにより topnav かつ responsive クラス属性値を持つ a 要素はブロックとして全て左寄せ，つまりフロートで右に回り込まないので，3 本線アイコンがクリックされると a 要素は家マーク・アイコンの下に「縦並び」配置となって現れる仕組みとなっている。

なお全称セレクタ「*」を用いて box-sizing に border-box を指定しておくことで全てのブロックは指定した width の通りに要素を表示してくれるようになる。responsive.css における当該箇所の記述を確かめよ。

外部リソースの活用

本講義の演習用ファイル *.html で用いているナビゲーションは，基本的には HTML, CSS, JavaScript の機能だけを用いて実現しているが，家マーク・アイコンや動的 3 本線アイコンといった（フリーで提供されている）「外部リソース」（External Resources）を併せて活用すると，Web ページをよりリッチに作成することができるようになる。こうした手法を使わない手はない。

multilang.html では他にも外部リソースとして

w3.css[*6] を使っている。HTML の head 内 link 要素で w3.css を読み込ませた上で

```
<div class="w3-row">
 <span lang="ja">
  <div class="w3-quarter w3-container">
   <p>... 日本語テキスト ...</p>
  </div>
 </span>
 <span lang="de">
  <div class="w3-quarter w3-container">
   <p>... Text auf Deutsch ...</p>
  </div>
 </span>
 <span lang="fr">
  <div class="w3-quarter w3-container">
   <p>... Texte en Français ...</p>
  </div>
 </span>
 <span lang="en">
  <div class="w3-quarter w3-container">
   <p>... Text in English ...</p>
  </div>
 </span>
</div>
```

のように HTML ソースに書いておくだけで 4 言語を対訳形式に 4 段組みで横並び表示できる。もちろん，画面表示領域幅が 620px 以下であれば自動的に縦並びとなる。w3-quarter を w3-half とすれば 2 段組み，w3-third とすれば 3 段組みとなる。

index.html でも「今日の福岡市城南区の天気」と「福岡大学人文学部へのアクセス」の箇所に外部リソースを用いている。後者は Google Maps API（Application Programming Interface）を利用した Google Map（博多駅から福岡大学までのルートマップ）を Web ページ内に埋め込んであり，前者は Dark Sky API を利用して「今日の福岡市城南区の天気」を表示させている。いずれも利用者登録を行えば，一定の使用条件の下，無償で利用できる。このようなコンテンツは Web Widget（和製英語では「ブログパーツ」）と呼ばれ，機能やデザインの点で Web ページにちょっとしたアクセントを添えることができる。

利用者登録の方法や各 API の具体的な使用法は割愛するが（興味のある者は自らインターネットで検索して調べること），API の供給先から取得したコードを HTML の iframe（Inline Frame）要素（文書内に別の文書などの閲覧コンテンツを入れ子状に配置する）内に貼り付けるだけで Web ウィジェットが嵌め込まれた動的な Web ページが出来上がる。

第 8 講演習

演習用ファイル： euro_ict2b.zip を展開した一式

1. multilang.html を元とし，w3.css を使って 3 言語対訳形式の HTML を作成せよ
2. 家マーク・アイコンを #ffff74 色にせよ
3. 3 本線アイコン（fa-bars）を fa-angle-double-down に替えてみよ．その他にも自分の好きなアイコンに替えてみよ

第 9 講

リンクと画像

HTML でのテキスト・リンクは

```
<a href="https://...">Dark Sky API</a>
```

といった書式で記述する。こうすることで a タグで囲まれたテキストにリンクが張られ，href 属性に記したリンク先が参照される。リンク先が HTML，画像，PDF であれば通常 Web ブラウザ内でコンテンツを閲覧でき，MS-Word や Excel 文書であれば対象ファイルを保存するかどうかといったことを尋ねるダイアログボックスが開く。href 属性値には絶対パス，相対パスいずれの表記も可能で，相対パスであれば同 Web ページ内あるいは Web サイト内のリソースを参照先に指定できる。

テキスト・リンク箇所はデフォルトでは各 Web ブラウザで決められた文字色で表示され，大抵の場合，そこに下線が付く。index.html をブラウザで開いて見よ。本文中で「Google Maps API」と「Dark Sky API」に a 要素 href 属性を使ってリンクを張ってあるが，今述べたような出力となっている様が確認できる。

しかし CSS にスタイルを適用することで，同じ a 要素のリンクでありながら「メニュー」のように下線の付かない見栄えに変更することもできる。HTML の a 要素に href 属性しか与えていないにも拘らず，双方で

[*6] 世界最大の Web 開発サイトを運営する W3Schools 社によって無償配布されている標準機能のみを使った CSS で，レスポンシブな Web サイト構築のための現代的なフレームワークを提供してくれる。

異なる出力となっていることを index.html をブラウザ，Emacs それぞれで開いて確かめよ。なお，メニューにおける a 要素の設定については responsive.css で行っているが，これについては ICT IIB 第 3 講（37 ページ参照）で既に説明した。

HTML では JPEG（Joint Photograph Experts Group），PNG（Portable Network Graphics），GIF（Graphics Interchange Format）といった画像形式（拡張子はそれぞれ jpg, png, gif）が使える。画像を使う場合 HTML には img 要素を用いて

```
<img src="./jpg/sw_leg.jpg"
     style="max-width: 100%; height: auto;"
     alt="白雪姫のエッチング画像" />
```

のように書いておく。src 属性値は画像の参照先を示すもので絶対パス，相対パスのどちらの表記でも良い。style 属性で CSS を設定できる。上例では max-width プロパティにより画像拡大の横幅最大値を 100%（つまり最大でもオリジナルの横幅一杯までしか拡大されないので解像度の劣化を招来しない．縮小はされる）とし，height プロパティ値を auto とすることで高さは幅の比率を保ったまま横幅に追従するようにしてある。alt（Alternate text）属性には「画像を使わず文字だけで表現する際に置き換わるべきテキスト」を指定しておく。

img 要素の style 属性では他にも様々なプロパティが使える。width で画面表示領域の表示幅を指定する（相対値指定が良い）。float で left, right を値として与えると，画像の右，左にテキストを回り込ませることができる。その際，画像の左・上・右・下にマージンを設けるには margin-left/top/right/bottom に適当な値を設定する。border-radius で画像の四隅の面取りができる。opacity を使えば画像の不透明度をコントロールできる（0.0 から 1.0 の値．0 に近い値ほど画像が「薄く」表示される）。テキストの回り込みをさせず画像を中央寄せにする場合は，div 要素の style 属性で text-align プロパティ値を center にセットした上で画像を括ってやれば良い。

以上，詳しくは multilang.html ソース内にある各 img 要素の style 属性の箇所をチェックし，ブラウザでのそれぞれの出力と比べてみること。

最後に index.html で用いている背景画像の説明である。トップページの背景画像（薄く EURO ICT Emacs, HTML5, EURO ICT TEX, CSS3 の各文字列が確認できる）はブラウザ・ウィンドウの幅に応じて自動的に拡大縮小する。こうした挙動は responsive.css の次の箇所で設定している。

```
body.bgImg {
 background-image: url("../jpg/bgimg_pale.jpg");
 background-position: center top;
// background-repeat: no-repeat;
 -moz-background-size: contain;
 background-size: contain;
}
```

background-image プロパティで背景画像を指定し，画像の表示位置を background-position で横方向（left, center, right の 3 択）に center，縦方向（top, center, bottom の 3 択）に top としている。background-size では画像の寸法を指定する。ここではトリミングしたり縦横比を崩すことなく画像ができるだけ大きく表示されるように自動的に拡大縮小がなされる contain を値にセットしている（他には cover, auto, 相対値指定がある．詳しくはインターネット検索で調べること）。Vendor Prefix でブラウザ向けの対応もしている。contain のデフォルトでは画像が画面領域の全てを覆うように繰り返し表示されるが，// でコメントアウトしてある箇所を活かすと繰り返し表示されなくなる。

responsive.css では h1 セレクタでコメントアウトしてある箇所がある。ここでは h1 見出しを「背景画像」とともに表示するスタイルを記述している。background-size プロパティの値を cover（画像の縦横比を崩すことなく画像ができるだけ大きくなるよう拡大縮小する．画像の縦横比が要素と異なる場合は空き領域が残らないように上下または左右のどちらかをトリミングする）としているので，画面表示幅に応じて背景画像がいつも良い具合に収まるようになっている。Web ページの最上位見出しである h1 に文字列だけではなく画像も用いたい場合は，このような方法も覚えておくと良い。

第 9 講演習

演習用ファイル： euro_ict2b.zip を展開した一式

1. responsive.css に手を加え，a 要素のテキスト・リンク文字列が teal 色となり下に red 色の点線が引かれるスタイルにせよ
2. multilang.html の img 要素 style 属性の各プロパティの値を色々と変えてみて，それぞれの出力の様を確認せよ

3. responsive.css の h1 セレクタ箇所の記述で background に画像を指定している方を活かし，そうでない方をコメントにして，出力を確かめよ

第 10 講

フォーム

Web サイトには「問い合わせ」や「アンケート」といった閲覧者からの情報を得るコンテンツが盛り込まれていることがある。これは「フォーム」（Form）という要素でベースは HTML で書かれているが，フォームに入力された情報をやりとりする際は CGI（Common Gateway Interface），PHP（PHP: Hypertext Preprocessor），JavaScript 等といった別のプログラムが必要になる。もっとも，form 要素の action 属性値を mailto とし，ここにメールアドレスを記入しておけば，method 属性値 post で閲覧者からの情報をメールで受け取ることは可能であるが，現在においてメールアドレスを HTM ソースの中にそのままの形で直接書き込んでしまうことは，セキュリティ上の観点から避けるべきである。

本講では，演習用ファイルの中で唯一（JavaScript と組み合わせて）form 要素を使っている javascript.html についてフォームの説明をする。javascript.html ソースの中で

```
<form name="tlForm">
 <input type="radio" name="tlRadio" id="de" ...
 <input type="radio" name="tlRadio" id="fr" ...
 <input type="radio" name="tlRadio" id="en" ...
 <input type="radio" name="tlRadio" id="ja" ...
</form>
```

とある箇所がフォームで「ラジオボタン」（複数用意された選択肢の中から一つを選択できるボタン）を設置している所である。他にも input 要素の type 属性値を checkbox, text, textarea 等とすればチェックボックス，テキストボックス，テキストエリア等を設置できる。

ここでは複数のラジオボタンを一つのグループとして認識させるため，input 要素の name 属性に同じ値 tlRadio を与えている（tl は translate の意）。デフォルトでは「日本語」ラジオボタンに予めチェックが付いている。

さて，操作の対象を全て「オブジェクト」（もの）として捉えてプログラミングを行うことを「オブジェクト指向プログラミング」（Object-Oriented Programming）といい，JavaScript は「オブジェクト指向プログラミング言語」（Object-Oriented Programming Language）の一つである（簡易的なオブジェクト指向プログラミング言語という意味合いで「オブジェクトベース・プログラミング言語」（Object-Based Programming Language）と呼ばれることもある）。クライアント（閲覧者）側で HTML ソースを Web ブラウザで読み込むと JavaScript が起動し，多くの「ブラウザ・オブジェクト」（ブラウザのウィンドウを操作対象とする window，HTML ソースを操作対象とする document，等々）が自動的に生成されるため，JavaScript はこれらのオブジェクトを操作できるようになる。

ブラウザ・オブジェクトは window オブジェクトを頂点とするツリー状の階層構造を持っており，これらの各オブジェクトにアクセスするには階層の順番に沿って，通常のメソッドやプロパティ同様（後述），ピリオド「.」で各オブジェクトを繋いでいく。頂点の window オブジェクトは省略可能であるため，例えば window, document, form, radio の各オブジェクトにアクセスするための記述は以下のようになる。

```
document.tlForm.tlRadio
```

ここで tl.Form, tlRadio とあるものは HTML の form 要素と form 要素内 input 要素のそれぞれの name 属性値である。このように document の下にある各オブジェクトは（1 つの HTML ソース内に通常複数個存在するため）name 属性値を使ってアクセスする。

javascript.html ソースの form 要素は「Deutsch, Français, English, 日本語」という 4 択のラジオボタンを設置（デフォルトでは「日本語」にチェック）している箇所で，クライアント（閲覧者）が目的言語の「ラジオボタン」を「クリック」するごとに，予め用意してある「日本語テキスト」がそれぞれ「独・仏・英・日」語へと切り替わる仕掛けとなっている。これには DOM（Document Object Model）という Web ページを動的に操作するための手法を用いており，HTML ソース内の目的とする要素にピンポイントで直接アクセスしているため，Web ページの切替等を経ることなく，HTML 内のテキストをその部分だけ書き換えることができる。

form 要素内の各 input 要素の type 属性は radio（ラジオボタン）で，その他にも onclick という属性が書き込ま

れており，これらの値は皆 tlTextTo(value) である。これは「ラジオボタンにチェックが入れば（つまりクリックされれば）引数を value とする tlTextTo という関数を呼び出せ」という命令となっている。onclick は JavaScript に用意されている「イベントハンドラ」(Event Handler) と呼ばれる仕組みで，これを使えば「閲覧者がラジオボタンを click するというイベント」を捉え，それに対する処理を行うことができるようになる。tlTextTo(value) 関数の中身は JavaScript を用いて script 要素内で定義している。このように HTML ソース内での JavaScript コードは script タグで括って記述する。javascript.html 内の

```
window.onload = init;
function init() {
 document.tlForm.tlRadio[3].checked = true;
}
function tlTextTo(value) {
 var idDiv = document.getElementById('khm53ja');
 switch(value)
  {
   case "de":
   idDiv.innerHTML = "...";
   break;

   case "fr":
   ...
  }
}
```

とある箇所を見よ。先ず window オブジェクトに用意されている onload イベントハンドラを使って，Web ブラウザによる HTML ソースの読み込み（ロード）が完了すると関数 init を呼び出す。init の中身は function 命令を用いて「document オブジェクトの下（つまり HTML ソース内）で tlForm 名を持つ form 要素について，tlRadio 名を持つ配列オブジェクトの 3+1 番目（つまり「日本語」ラジオボタン）の要素に予めチェックマークを付けよ」のように定義している。init のように引数を取らない関数であっても丸カッコ（Parenthesis）は省略できないことに注意。なお，プログラミングにおける 1 つの命令単位を「ステートメント」(Statement, 文）と呼ぶ。JavaScript のステートメントの終わりは「セミコロン ;」で締める。

次に function 命令によって，value という引数を取る tlTextTo 関数を定義している。ここでは idDiv という変数に「HTML ソース内で khm53ja という id を持つ要素（これは div 要素の 1 箇所のみ）」を代入しておき，多岐分岐（値に応じて処理を分岐させる）を記述する switch 命令により「引数が de, fr, en, それ以外の default（つまり ja 日本語）に応じて，idDiv 要素部分を idDiv.innerHTML に記した値で動的に書き換える」という内容となっている。多岐分岐は case ブロックにより行うが，末尾に必ず break 命令が埋め込まれていることに注意。break 命令があってはじめて現在のブロックから処理が抜けられる。これがないと処理は次のブロックへと移動してしまい，意図した結果とならない。一致する case ブロックが見つからなかった場合は，最終的に default ブロック（これはデフォルトの「日本語」テキスト）が呼び出される。

最後にユーザビリティ（Usability）への配慮に関する点を一つ。ラジオボタンにチェックを入れるのにデフォルトではピンポイントでボタン部分をクリックしなくてはならない。これではやや使い勝手が悪い。こういった場合，関連する文字列を label タグで括り，その中の for 属性値を当該ラジオボタンの id 値と同じくしておけば，以降は関連する文字列も含めてクリックできるようになる（クリックできる有効範囲が広がる）。

第 10 講演習

演習用ファイル： euro_ict2b.zip を展開した一式

1. javascript.html ソース内の break; (3 箇所ある) をコメントアウトすると出力はどのような挙動となるか
2. Français のラジオボタンにデフォルトでチェックを入れるには HTML ソースをどのように変更すれば良いか．この場合，関連して書き換えるべき箇所を全て考えよ
3. ラジオボタンのクリック時における label 要素の有効性を確かめよ（ヒント： HTML の各要素は，開始・終了タグを含め，C-c C-d で一気に削除できる）

第 11 講

ポジションとフロート

CSS による Web ページのレイアウトは主に position と float という 2 つのプロパティでなされる。position は relative（相対位置へ配置）, absolute（絶対位置への配置）, fixed（absolute と同じだがスクロールしても位

置は固定）, static（配置方法なし，初期値）という 4 つの値を取り得るが，本講ではこの中で良く用いられる relative と absolute について説明する。「ヨーロッパ学 ICT IIB サンプル・ウェブ」に含まれる HTML/CSS の全ソースファイル（*.html, *.css）の中で fixed, static は用いていない（外部ライブラリである all.css と w3.css を除く．前者では static，後者では static と fixed が用いられている）。

先ずは position プロパティの取り得る値 relative と absolute から説明する。noto.css の以下の記述箇所を確認せよ。

```
\noindent
.text_sample note {
  position: relative;
  margin-top: 1em;
  display: block;
  text-align: right;
  color: #888888;
  font-size: 70%;
}
```

上記 CSS によるスタイルは，no_more_tofu.html をブラウザで開いたときの 3 箇所の出力「Noto を使わない場合，Noto を使った場合，Noto を使った異体字の表示」に反映されている。text_sample クラスセレクタの note 子孫セレクタを使ってスタイルを定義したこの 3 箇所は，「地の文」（通常のテキスト）から「見本用テキスト」を区別して表示する際に用いており，note の部分は見本用テキストに関する一種のキャプションとなっている。

さて，今，noto.css の当該箇所で relative とあるところを absolute に変更・保存してから no_more_tofu.html をブラウザで開いて見ると，各キャプション部が「左に移動」していることが分かる。position に関連する位置決めのプロパティは top, bottom, left, right の 4 つであるが，上例では margin-top プロパティの箇所のみに top プロパティが使われており，relative では note 要素ブロックがもともとあった位置を保持しながら元の位置を基点に margin-top と text-align プロパティの値に従って表示位置が移動し「右寄せ，下寄せ」にキャプションが配置される。これに対し absolute では note 要素ブロックの元スペースは保持されず，margin-top と text-align プロパティの値に従って表示位置が決定される。absolute では text-align で right を指定しているにも関わらずキャプションが右寄せにならない（ように見える）のは，display プロパティの block 指定による。

position プロパティにおける absolute 値は responsive.css で一箇所のみ使用している。それは画面表示領域幅が 620px 以下になった場合の a 要素の出力に関連する箇所で，既に ICT IIB 第 8 講でその挙動について説明してある（46 ページ参照）。画面表示領域幅が 620px 以下になったとき a は最初の子要素（家マーク・アイコンの付いたトップページ）を除いて全て非表示となり，3 本線アイコンがブロック表示の右寄せで出現するが，このような出力となるように記述している箇所が

```
@media screen and (max-width: 620px) {
 .topnav.responsive {position: relative;}
 .topnav.responsive .icon {
   position: absolute;
   right: 0;
   top: 0;
 }
  ...
}
```

の部分である。position の relative 値により a 要素が非表示となってもその位置が保持されるため，3 本線アイコンをクリックしたときに表示される各 a 要素ブロックは適度な表示幅を維持したまま，3 本線アイコンも家マーク・アイコンの付いたトップページのブロックに重ならない。3 本線アイコンが必ず右寄せで出力されるのも position に absolute 値をセットし，right, top プロパティにそれぞれ 0 の値を与えているからである。仮にこれを relative とすると 3 本線アイコンはクリックしたときに表示される各 a 要素ブロックの下に出現してしまう。

float プロパティについては ICT IIB 第 3 講でも既に説明した（37 ページ参照）。float は文字通り対象としたい要素を「浮かばせて」左や右へと移動させる場合に用いる。responsive.css の以下の箇所を見よ。

```
.left {
  background-color: #e5e5e5;
  float: left;
  width: 20%;
  text-align: center;
  padding: 10px;
}
```

この例のように，float プロパティを使う場合は併せて横幅（Width）も指定せねばならない（画像を除く）。デ

フォルトでのブロックレベル要素は横幅 100%，つまり画面の横幅一杯に広がるように表示されるので，width プロパティで横幅を制限しておかないと float で左右に要素を寄せることができなくなる。画像はそれ自身で横幅のサイズを持っているため float での width 指定は不要である。a 要素の display プロパティで block 値をセットする場合も width 指定は不要である。a 要素のテキスト・リンク箇所の文字列により横幅が自動的に決まるためである。

float プロパティに指定する値は left, right のいずれかである（特に配置を指定しない none もある．これがデフォルト）。こうしておけば指定した方向に要素が移動し，続く要素は最初の要素の上端から回り込むように配置される。

index.html では float を活用して RWD 対応の Web ページとした（詳細は ICT IIB 第 3 講で既に説明した．37 ページ参照．設定は responsive.css による）。multilang.html では img 要素（つまり画像）に float プロパティを宛がうことでテキストの回り込みを行わせている。いずれもソース（index.html, responsive.css, multilang.html）とブラウザでの出力を良く比較すること。

欧文 Web フォント： FrakturMaguntia

Web フォントは junicode.html においても活用している。こちらで使用しているのは欧文フォント 2 種であるが，いずれも古いテキスト（欧文古典）等の専門的な出力に用いられることを想定している。

1 つは Fraktur と呼ばれるドイツ旧字体フォントの UnifrakturMaguntia であり，OpenType という文字組版に関する多くの高度な拡張機能[*7] を備えた規格のフリーフォントである。Maguntia とはラテン語でドイツの街マインツのことを言う。この名称はこのフォントの由来[*8] による。

UnifrakturMaguntia を Web フォントとして用いるための設定は，まとめて unifrak.css に記してある。eot, woff2, woff, ttf, svg といった形式の Web フォントは，「Web 上での Web フォントへの変換サービス」（無償）を行っている Font Squirrel 社を通じて，オリジナルの UnifrakturMaguntia フォントから全て自前で用意した。

また特に Fraktur で組まれた 16 世紀の欧文組版に特徴的な要素を予め反映させた UnifrakturMaguntia16[*9] の Web フォントも用意した。UnifrakturMaguntia16 は cv11（語頭での s は自動的に長い ſ で出力する），cv12（r には異体字を用いる），cv13（大文字の I は J で出力），cv14（大文字ウムラウトの場合 Ä でなく Ae と綴る），cv15（ä の代わりに小添字 e 式の å で出力），cv19（Gedankenstrich を en-Dash ではなく em-Dash とする），hlig（etc. 等を特殊字形で出力），lnum（算用数字をラテン字体にマッチする字体で出力），ss02（Und bevor を Vnd beuor で出力）がデフォルトとなっている。cv 等の略語は「異体字形」(Character Variants)，「文体セット字形」(Stylistic Sets)，「歴史的合字」(Hirtoric Ligatures）を指す。UnifrakturMaguntia16 を Web フォントとして用いる設定は unifrak16.css で行っている。

ラテン字アルファベットでは強調部分に「イタリック体」(元来「イタリア風」の意）を使うことがあるが，ドイツ旧字体のフラクトゥア体にイタリック体は有り得ない（「イタリア」風でありながら「ドイツ」旧字体とは形容矛盾！）。ドイツ旧字体での強調箇所には「隔字体」を用いるのが正書法である。デジタルフォントの FrakturMaguntia では，しかし，技術的に斜字体も可能であるから時代考証を考慮する場合はこの点に気を付けること。また，隔字体となった場合でも ch, ck, st, tz (ch ck ſt ſs) の 4 つの合字は隔字されない（sz ß を含めれば 5 字)。UnifrakturMaguntia では原則としてこうした細かなルールも反映させた出力を自動的に行ってくれるが，現時点では全てのブラウザが OpenType 機能に対応しているわけではない。非対応ブラウザの場合，正確性を追求するのであれば手作業で HTML ソースに介入する必要がある。具体的には自動合字を抑制したい箇所に ‌「ゼロ幅非接合子」(Zero Width Non-Joiner）を，自動的に合字されないが合字させるべき箇所に ‍「ゼロ幅接合子」を挿入する。HTML での隔字体は letter-spacing プロパティを使って CSS で実現する。値は 0.3em くらいを基準に考えれば良いだろう。詳しくは junicode.html の当該箇所のソースおよびブラウザでの出力を比較チェックすること。

UnifrakturMaguntia フォントが備える OpenType の

[*7] 多種の合字，時代を反映した字体の切替，プロポーショナル・メトリクス，ペア・カーニング，ベースラインの指定，等々。

[*8] 1901 年に Carl Albert Fahrenwaldt がデザインした Mainzer Fraktur を Peter Wiegel がデジタル化した Berthold Mainzer Fraktur を基に Gerrit Ansmann によって OpenType 化されたフォントが FrakturMaguntia である。

[*9] 同様に 17, 18, 19, 20, 21 がある。21 は 21 世紀の Fraktur ということで時代考証に基づくものではなく，主にデザインの観点から作成されたセットである。

拡張機能を用いれば，Der Ärger des Igels liegt auf der schönen Königsstraße. という HTML ソースでの入力から Der Ärger des Igels liegt auf der schönen Königsstraße.（UnifrakturMaguntia の場合），Der Aerger des Igels liegt auf der schönen Königsstraße.（UnifrakturMaguntia16 の場合）といった出力が得られる（Mozilla Firefox, Google Chrome, Safari での出力を確認した）。

欧文 Web フォント： Junicode

もう一つの欧文 Web フォントはヨーロッパ中世のテキストを組むための専門的フォントである Junicode である。このフリーフォントは Peter S. Baker によって開発が続けられている Unicode フォントで，17 世紀に英国オックスフォード大学の出版物で用いられていた活字を元にデザインされている。Junicode という命名は，当時のアングロ・サクソン語文献学者である Franciscus Junius へのオマージュとなっている。Junicode はヨーロッパ中世のテキストを専門的に組みたい場合とても重宝する。Junicode を Web フォントとして用いるための設定は junicode.css で設定しているので，詳しくはそちらを参照すること。

junicode.html では『アングロサクソン年代記』[*10] をインシュラー体[*11] で，4 世紀のゴート人司教ウルフィラによる『ゴート語訳聖書』をゴート文字体[*12] で出力するように記述してあるが，いずれのテキストもラテン文字による「翻字」（Transliteration）で入力しておき，それぞれ文体セット字形の ss02 および ss19 を指定することでインシュラー体あるいはゴート文字体が出力される仕掛けとなっている。

詳しくは junicode.html ソースの当該箇所の記述と Web ブラウザでの出力を比較チェックすること。もちろん，インシュラー体やゴート文字体を直接入力（Emacs でユニコードの 16 進数コードや文字の正式名称を利用）しても同じ出力が得られるが，入力に難儀する。

ハイパーリンク付きの相互参照を施した脚注

HTML で脚注を利用することはあまりないかもしれないが，junicode.html では「ハイパーリンク付きの相互参照を施した脚注」を用いているので必要に応じて参考にして欲しい。このスタイルは access_fn.css で設定している。

HTML ソース内では脚注を付けたい文字列を a タグで括り，id 属性に適当な値を入れ，href 属性値に脚注の id を # を前置してセットする。逆に脚注 a 要素の href 属性値には対応する「脚注を付けたい文字列」箇所の id をセットする。脚注部分は ol, li 要素で処理するので自動連番される。脚注を付けたい文字列の方は全てに aria-describedby 属性に footnote-label という値を入れておき，脚注の方には全て aria-label 属性に Back to content という値を入れておけば，脚注を付けたい文字列箇所にはブラケットで括られた小さな上付き文字で自動連番が振られ，相互参照ハイパーリンクも実現される。この仕掛けには WAI-ARIA（Web Accessibility Initiative—Accessible Rich Internet Applications）という「アクセスのしやすさを向上させる」ために考案された属性群を使っている。

なお，junicode.html においてアクセスするクライアント（閲覧者）のデバイス（PC, スマートフォン等）上のブラウザの表示幅に応じて「2 段になったり 1 段になったりする箇所」は w3.css で設定している。詳しくは HTML ソースをチェックすること。

第 11 講演習

演習用ファイル： euro_ict2b.zip を展開した一式

1. responsive.css および noto.css における position プロパティの値 relative と absolute を適宜入れ替え，出力の違いを確認せよ
2. 欧文 Web フォントの UnifrakturMaguntia と Junicode の入出力を比較チェックせよ
3. junicode.html に手を入れ，ハイパーリンク付きの脚注を増やしてみよ

第 12 講

Web ページのレイアウトと Web サイトの編成

Web ページのコンテンツは技術的には作成者が自由にレイアウトできる。しかし，複数の Web ページから成る Web サイトであれば，各ページの見栄えやレイアウトを揃えておいた方が見やすく美的であろう。その際，先ず Web ページ定番のレイアウトを基本にデザイ

[*10] イングランドの 7 王国時代を含む出来事が主に記された年代記。9 世紀後半アルフレッド大王の治世に編纂されたと考えられている。
[*11] アイルランドからグレートブリテンに伝わり，その後ヨーロッパ大陸に広がった中世の書体の名称。
[*12] ゴート語を表記するためにウルフィラが考案した文字。

ンし，これを踏まえて応用的なレイアウトを考えていくのが良い。

Web ページ定番のレイアウトは「ヘッダ（Header），ナビゲーション（Nav），主要コンテンツ（Main），サイドコンテンツ（Aside），フッタ（Footer）」の各ブロックから成る。ここで言うブロックとはもちろんブロックレベル要素であるが，「情報の内容ごとに他と区分したまとまり」のことを指す。これらは HTML ソース内では header, nav, main, aside, footer という各要素でマークアップされる。

「ヨーロッパ学 ICT IIB サンプル・ウェブ」に含まれる各 Web ページは，具体的な複数のサンプル提示という意味もあり，「トップページ」（index.html）とそれ以外のページでレイアウトを変えて作成している。index.html ではヘッダを h1 要素に，ナビゲーションを h3 と a 要素で左サイドバーに代えて，主要コンテンツを h2 要素以下に，右サイドバーを h3 要素に，フッタを画面下の部分に，それぞれ対応付けてレイアウトを構成している。画面表示幅が十分にある場合，index.html は Web ブラウザ上で「3 ペイン」（Pane）で表示されるが，RWD 設計により画面表示幅が 620px 以下になるとナビゲーションにあたる「メニュー」と主要コンテンツと右サイドバーに相当する「福岡大学人文学部へのアクセス」部分が縦に整列する「1 ペイン」表示となる。

index.html 以外のページではヘッダを h1 要素に，ナビゲーションをヘッダ直下の a 要素に，主要コンテンツは h2 要素以下に，フッタを画面下の部分に，それぞれ対応付けている。これらの Web ページでは RWD 対応を考慮し，敢えてサイドコンテンツを置かない設計としている。

複数の Web ページから成る Web サイトを編成する場合，予め各種ファイルの格納の仕方を決めておくと，後々これを拡張するときや保守作業の際に労力を軽減できる。具体的には css（スタイルファイル），jpg（画像ファイル），js（JavaScript ファイル），png（画像ファイル）といったようなディレクトリを作成し，同じ拡張子を持つファイルは全て対応するディレクトリの中に入れておく。様々な拡張子を持つ Web フォントは fonts, webfonts 等といったディレクりに格納すると良いだろう。これらのディレクトリ内にある各種ファイルを HTML から参照する際は，ソースに相対パス表記をしておく。

Web ページでのメールアドレス表記とコメントの活用

Web ページでのメールアドレス表記について注意を一つ。自分のメールアドレスを Web 上にそのまま晒していると（つまり HTML ソースの中にそのまま書き込んでいると），「ボット」（Internet Bot, Web Robot, Bot）と呼ばれるプログラムによってメールアドレスが自動的に収集されてしまい，後になってスパムメールが山ほど送られてくる，ということが起こり得る。そもそもメールアドレスを Web 上に公開しなければこうした事態は出来しないが，Web ページを閲覧してくれる人との通信回路は確保しておきたいのが実情である。

こちらが望まないボットにメールアドレスを収集されてしまわないようにするには，メールアドレスを画像として置くやり方から Web ページ全体を暗号化してしまう方法まで様々なレベルの対処法があるが，ここでは「コメント」表記を利用したメールアドレス記法を紹介する。index.html ソースの footer 要素内 Comment to: の箇所を見よ。そこには

```
<!-- @nhy.com++ -->ynaga<!-- 12334@...
```

という文字列で始まる部分が含まれるが，この部分で HTML のコメント表記を利用して実際のメールアドレスを記載している。ボットが自動収集・抽出するのは HTML ソース内のメールアドレス（とボットが判断する）箇所であるから，当該箇所を「ボットを錯乱させるような形」で記しておけば，望まないボットに正しいメールアドレスを引っこ抜かれてしまう可能性を減ずることができる。

なお，HTML のコメント箇所は `<!-- ... -->`，CSS のコメントは `/* ... */` で括る。CSS におけるコメントが 1 行で済む場合は `//` を前に付ける形式でも良い。JavaScript におけるコメントは CSS の場合と同じである。いずれの場合でも，これらコメント箇所は Web ブラウザや JavaScript といったプログラムの処理対象から外されるので，結果として先のメールアドレス部分は「Web ブラウザ上では正しいメールアドレス」として表示され，コピー & ペーストも普通に行えるようになっている。

Emacs では一意のキー操作 C-x C-; でコメントアウト（toggle キーなので「イン」も）できたことを思い出すこと。出力されるコメント記号は major mode に追従す

るが，Emacs 26 バージョン以降であれば，HTML 内に CSS や JavaScript のコードがあった場合，それらに相応しいコメント記号が出力されるよう自動的に切り替わる仕様となっており C-x C-; の使い勝手が向上している。

第 12 講演習

演習用ファイル： euro_ict2b.zip を展開した一式

1. index.html および index.html 以外の *.html ソース内の header, nav, main, aside, footer 要素をそれぞれ確認し，Web ブラウザでの出力と照らし合わせてチェックせよ
2. ボット対象の「メールアドレス錯乱標記」の中身を検証せよ

第 13 講

本講以降，3 回の講義で JavaScript を使った Web ページ作成法の初歩を学ぶ。JavaScript は Web ブラウザ上で動作するスクリプト言語の代表として，今では様々な Web アプリケーションの開発に欠かせない存在となっている。簡易的なオブジェクトベース言語と見なされることもある JavaScript ではあるが，アイディア次第では「動的」な Web ページ作成や Web サイト構築に関する活用において無限の可能性を持っている。受講生は，各自の興味に従い，自ら広く深く学習されたい。

JavaScript

プログラムを作成することを「プログラミング」といい，プログラムの記述には「プログラミング言語」（Programming Language）が使用される。機械だけが理解可能な「マシン語」（Machine Code）ではなく，人間も理解できる形の「高水準言語」（High-Level Programming Language）で記述されたプログラムを「ソースコード」（Source Code）と呼ぶ。ソースコードはテキスト（ファイル）である。

コンピュータの CPU（Central Processing Unit; 中央処理装置）はソースコードをそのまま理解することができないため，プログラムを実行させるには何等かの方法でソースコードをマシン語に変換する必要がある。この変換方式には大きく「コンパイラ方式」（Compiler System）と「インタプリタ方式」（Interpreter System）の 2 つがある。前者は「コンパイラ」と呼ばれるプログラム（ソフトウェア）を使ってソースコードを「オブジェクト・コード」（Object Code; オブジェクト・ファイルとも）と呼ばれるマシン語ファイルに一括変換する方式で，例えば TEX はこの方式のプログラムであると考えることもできる[*13]。後者は「インタプリタ」と呼ばれるプログラムがソースコードを 1 行ずつ解釈しながら実行する方式で，JavaScript はインタプリタ方式のプログラミング言語である。具体的には，HTML ソース（テキストファイル）が Web ブラウザに読み込まれると，ブラウザに内蔵されたインタプリタによって，HTML ソース内に記された JavaScript 用のソースコードが解釈され，実行される。

通常のテキストである JavaScript のソースコードは，HTML 内に script 要素として直接書き込むか，あるいは，ソースコードのみを別のテキストファイル（*.js）として保存しておき HTML からこのファイルを参照し読み込ませて処理させる。JavaScript のインタプリタは各 Web ブラウザに内蔵されているためブラウザごとの機能の違いがなくはないが，JavaScript の基本部分についてはジュネーブに本部を置くヨーロッパの規格標準化団体である Ecma International (Ecma: European Computer Manufacturers Association) により ECMAScript として標準化されている。

ICT IIB 第 10 講でも既に言及したが（49 ページ参照），JavaScript はオブジェクト指向のプログラミング言語である。オブジェクト指向言語では操作対象を「オブジェクト（もの）」（Object）として捉え，各オブジェクトには「プロパティ」（Property）と呼ばれるそれぞれのオブジェクトに予め用意されているデータと，「メソッド」（Method）というオブジェクトの操作手法が付随する。JavaScript でプログラムを書くということの中身は，望む結果を得られるようにこれらプロパティやメソッドを的確に活用しながらソースコードを書く，ということになる。

JavaScript プログラムの作成に際し特別のソフトウェアを調達する必要はない。ソースコードはテキストファイルであるからテキスト・エディタの Emacs がそのまま使える。Emacs は Syntax Highlight を始め，様々な入力支援機能をデフォルトで備えている。JavaScript の出

[*13] ソースコードは *.tex ファイル，コンパイラは uplatex, dvipdfmx, pdflatex 等，オブジェクトコードは *.pdf ファイル。ただし TEX を純粋なインタプリタ言語と見なす人もいる。

力結果は各種 OS 付属の Web ブラウザ（Microsoft Edge, IE, Mozilla Firefox, Google Chrome, Safari 等）で確認できる。

ただし，TEX, HTML, CSS 同様，JavaScript のソースコードを記述する際の文字エンコーディングは UTF-8, 改行コードは LF に統一しておくのが良い。こうしておけば，後々，文字コードの不整合に起因する無用のトラブルに見舞われにくくなる。

JavaScript の有効化

JavaScript ソースコードが含まれる HTML ファイルを Web ブラウザで開けば，ローカル環境であっても，直ちにプログラムの動作を確認できる。ただし，初期状態では HD（Hard Disc）内のファイルに記述したプログラムを実行できないように設定されているブラウザも存在する（例えば Microsoft Internet Explorer）。従って，このようなブラウザを使って JavaScript の挙動をチェックする場合は，ローカルファイルの JavaScript も実行できるよう相応の設定を済ませておくこと[*14]。

Emacs（テキストエディタ）と Web ブラウザの準備が整ったら，javascript.html を Emacs, ブラウザのそれぞれで開いて見よ。HTML ソースでは，head 要素内の script 要素を用いて *.js ファイル（いずれも js ディレクトリに格納されている）を複数参照していることが分かる。これらが HTML ソースの「外部」にある JavaScript ソースコードを読み込むための記述である。さらに HTML ソースの先を読み進めていくと script 要素に出くわすが，ここが HTML ソース「内」に JavaScript ソースコードを書きこむ箇所となっている。

最初に登場するファイル「内」JavaScript は myFunction() の定義に関するソースコードであるが，これについては ICT IIB 第 8 講で既に解説した（46 ページ参照）ので次の script 要素に着目すると

```
document.write("<p>.. + .. + </p>");
...
```

とある。document は Web ブラウザ上のオブジェクトの 1 つ（ブラウザに HTML が読み込まれると多くのオブジェクトが自動的に生成され JavaScript による操作の対象となる）で，ピリオド「.」を用いて次のメソッド write（HTML に文字列を出力する）と連結している。write の引数は丸カッコ（Parenthesis）の中に収めるが文字列は "..." あるいは '...' のように 2 重あるいは 1 重の引用符で括る。文字列として取り扱うのでなければ数値に引用符は付けない。文字列の中に " や ' の記号そのものが出てくる際はこれらを \ 記号を前置しエスケープして，文字列を括る役目を持つそれぞれの引用符と区別せねばならない。

write メソッドに複数の引数がある場合，各引数を「,」で区切ればそれぞれの引数による出力結果を連結できる。, の代わりに演算子（Operator）「+」を使っても良い。+ 演算子は文字列の場合は連結，数値の場合は加算となる。

プログラムにおける 1 つの命令単位を「ステートメント」（Statement）と呼ぶ。ステートメントの終わりには「;」を打つ。

platform.description は platform オブジェクトと description プロパティを連結した表記で，メソッド同様 JavaScript のプロパティは . でオブジェクトと連結することでそこにアクセスできるようになる。platform と description は「外部ライブラリ」（External Libraries）である platform.js によって定義されているオブジェクトおよびプロパティで，これは javascript.html にアクセスしたクライアント（閲覧者）の使っている OS と Web ブラウザとを「戻り値」（Return Value）として返すことからメソッドと考えることもでき，この戻り値は最終的には document.write ステートメントにより他の文字列と連結されて「クライアントの使用 OS および Web ブラウザ情報」を HTML に書き出してくれる。

noscript 要素には JavaScript の機能がオフとなっている場合に代替として表示すべきコンテンツを書いておく。JavaScript は閲覧者側の Web ブラウザで自由にオン・オフできるため，本来であれば，JavaScript の作動の有無に拘らず同様のコンテンツを閲覧できるように Web ページを作成すべきであるが，ページの仕組みとしてどうしても JavaScript を使わざるを得ない場合は，その旨をメッセージとして記しておくのが良い。

第 13 講演習

演習用ファイル： euro_ict2b.zip を展開した一式

[*14] 例えば IE では，ツール，インターネットオプション，詳細設定と辿り「マイコンピューターのファイルでのアクティブコンテンツの実行を許可する」のチェックボックスにチェックを入れる。

1. Microsoft Edge, Internet Explorer, Mozilla Firefox, Google Chrome, Safari 等々，様々な Web ブラウザから javascript.html をローカルでも，インターネット上でも開いて見よ
2. javascript.html ソース内 script 要素の document.write 箇所における " と ' の使い方をチェックせよ
3. 同じ箇所の + 演算子を , で置き換えてみよ
4. 様々なデバイスから Web 上の javascript.html にアクセスし, platform.description の戻り値を Web ブラウザ上で確認してみよ

第 14 講

日・独・仏・英語で「今日の日付」を表示

外部ライブラリ等を一切使用せず，JavaScript の標準機能だけを用いて「クライアントがアクセスした時点での今日の日付を日・独・仏・英の 4 言語でそれぞれ表示する」プログラムを作成してみよう。

こういったプログラムを作成する場合，先ずどういったことを考えねばならないのか，ということをまとめてみる。

1. JavaScript はクライアントサイドで作動するので，各閲覧者がアクセスした時点での日付を取得せねばならない
2. 月名，曜日名,「今日は何月何日何曜日です」というような表現形式は，言語ごとにそれぞれ異なっている
3. 上に挙げた 4 言語の中で月名に数字を用いるのは日本語だけである

以上の事柄を念頭に javascript.html ソース内 script 要素の以下の箇所を見よ。

```
var now = new Date();
var year = now.getFullYear();
var date = now.getDate();
var dmonth = new Array("Januar", ...);
var fmonth = new Array("janvier", ...);
var emonth = new Array("January"," ...);
var dday = new Array("Sonntag", ...);
var fday = new Array("dimanche", ...);
var eday = new Array("Sunday", ...);
var jday = new Array("日", ...);
...
```

Web ページがブラウザ上に表示される時点で自動生成される document オブジェクトとは異なり，プログラム作成者が必要に応じて自分で生成させたいオブジェクトがある。例えば，Date という「ある時点での年月日時刻データ」を操作するためのオブジェクトがこの種のもので，これは JavaScript で予め用意されている「組み込みオブジェクト」（Build-In Object）と呼ばれるものの一つである。頭字は大文字である。JavaScript では大文字・小文字は区別して用いられる。

var は変数（Variable）now（これももちろんオブジェクト）を新規作成して宣言する命令で，これに右辺の new Date() を代入している。生成され利用可能となったオブジェクトを「インスタンス」（Instance）と呼ぶが，new は新しいインスタンスを生成する演算子である。インスタンスを生成するには new 演算子に加えて「コンストラクタ」（Constructor）と呼ばれる特別な関数を使用する。コンストラクタの名前は通常オブジェクト名と同じであるから Date オブジェクトのコンストラクタは Date となる。

Date コンストラクタの引数部分が空白であることに注意。これにより「現在の日付および時刻」を持つインスタンスが生成される。明示的に指定する日時を生成させたい場合は

```
var datum = new Date(2018,8,6,18,40,36);
```

のように指定する。これは「2018 年 9 月 6 日 18 時 40 分 36 秒」の場合である。「月」は Date の第 2 引数「8」に「プラス 1」した数値となることに注意せよ。

new Date() で得られた「現在の日付および時刻」を代入した now 変数から，予め用意されている getFullYear(), getDate() といった各メソッドを用いて「年」(4 桁),「日」（1 から 31）という数値を引き出し，これを新たに宣言して作成する year, date 変数にそれぞれ代入する。

次に Array コンストラクタを用いて dmonth（ドイツ語の月名），fmonth（フランス語の月名），emonth（英語の月名）変数に，各言語による月名を「配列」（Array）にして格納しておく。同様に dday, fday, eday, jday にそれぞれの言語による曜日名を格納する。日本語の月名は独・仏・英のような固有名ではなく「数字」月という形を取るため，ここでわざわざ配列を作っていないことにも注意。ここまでの準備が済めば

```
var dmonth = dmonth[now.getMonth()];
var fmonth = fmonth[now.getMonth()];
var emonth = emonth[now.getMonth()];
var dday = dday[now.getDay()];
var fday = fday[now.getDay()];
var eday = eday[now.getDay()];
var jday = jday[now.getDay()];
document.write("<p style= ... </p>");
document.write("<p style= ... </p>");
document.write("<p style= ... </p>");
document.write("<p style= ... </p>");
```

のように，dmonth 配列の中の now.getMonth() 番目の要素（配列内の要素は 0, 1, 2, とカウントされる，以下同様）を変数 dmonth に，dday 配列の中の now.getDay() 番目の要素を変数 dday に代入する。fmonth/fday, emonth/eday, jday についても同じ工程となる。そして最後に「現在の日付」に関する各変数 year（西暦，4 桁），dmonth/fmonth/emonth（独・仏・英語での月名），dday/fday/eday/jday（独・仏・英・日本語での曜日名）を，適宜各言語表記に固有の「定冠詞やコンマやピリオドといった文字列」（「文字列リテラル」String Literal とも）と「+ 演算子」（文字列では「結合」，数値では「加算」）で結合し，document.write メソッドを用いて出力する。その際，CSS の background-color と color プロパティを使って背景色と文字色を指定していること，それらの表記においては引用符合を必要に応じてエスケープしていること，にも注意すること。

日本語の「月」情報を得る箇所が `now.getMonth() + 1` となっている点にも十分注意すること。これは getMonth メソッドの戻り値が「0 から 11」であり，「0 が 1 月，・・・，11 が 12 月」に対応しているための措置である。getDay も同様に「0 から 6」が戻り値であり「0 が日曜，・・・，6 が土曜」となっている

福岡，Berlin, Paris, London における現在日時表示

次に「福岡，Berlin, Paris, London」の現在日時をリアルタイムで一覧表示する例を紹介する。javascript.html を Web ブラウザで開き，当該箇所を見よ。一見すると単純で簡単そうに思われるが，こうした「世界時計」を実現するには

1. クライアント（閲覧者）のデバイスの日時を取得する
2. 取得した日時をリアルタイムで書き換え続ける
3. 日本とドイツとフランスとイギリスの「時差」も考慮する
4. 「夏時間」に対応させる（独・仏・英国がこの制度を導入しているため）
5. 「夏時間がいつからいつまでなのか」をプログラムに盛り込む．たとえば中央ヨーロッパ夏時間は「3 月の最終日曜日午前 2 時（＝夏時間午前 3 時）から 10 月の最終日曜日夏時間午前 3 時（＝標準時午前 2 時）まで」というように「可変」である
6. 言語ごとに異なる「表記」形式（月名，曜日名など）に対応させる
7. 「協定世界時」（UTC: Universal Time Coordinated）との時間差も記す．これは標準時と夏時間で異なってくる
8. JST: Japanese Standard Time（日本標準時），CET: Central European Time（中央ヨーロッパ時間），CEST: Central European Summer Time（中央ヨーロッパ夏時間），GMT: Greenwich Mean Time（グリニッジ平均時＝協定世界時），BST: British Summer Time（イギリス夏時間）といった付加情報も添える

といった全ての事柄を考慮せねばならず，JavaScript の標準機能だけを使って全てをゼロからプログラミングすることはそれほど簡単なことではない。

こうした場合，幸い，多くのプログラミング言語には「ライブラリ」と呼ばれる「汎用性の高い複数のプログラムを誰でも無償で再利用できる形でまとめられた」ものが存在しているので，これらを有効活用すると良い。「福岡，Berlin, Paris, London における現在日時表示」では moment-with-locales.js, moment-timezone-with-data.js という 2 つのライブラリを利用している。第 12 講で説明した platform.js もライブラリである。第 8 講の w3.css, 第 11 講の access_fn.css も同様である。よりユーザビリティの高い Web サイトを構築しようと考えるならば，こうしたライブラリの導入は現実解の一つとなる。

以下，ドイツ国ベルリンにおける現在日時を表示する JavaScript ソースコードについて説明する。Paris, London, 福岡に関するソースコードも基本的には同じである。説明の都合上，本来 script 要素の中に書かれていない（つまり JavaScript のソースコードではない）div 要素箇所も併せて掲げてある。

```
<div id="cetd"></div>
...
var updateCetd;
(updateCetd = function() {
moment.locale('de');
document.getElementById("cetd")
.innerHTML = "<p style ..."</p>";
})();
setInterval(updateCetd, 1000);
```

先ず変数 updateCetd を宣言する。なお，文字列を 1 語で記す場合，updateCetd のように区切り位置の語頭を大文字にする記法を「キャメル・ケース」（Camel Case; 大文字部分が駱駝のコブに見えるため），UpdateCetd のように先頭の語頭も含めて大文字にする記法を「パスカル・ケース」（プログラミング言語の Pascal で用いられるため）と呼ぶ。JavaScript では camelCase 記法が良く使われる。他にも snake_case, kebab-case 記法がある[*15]。

updateCetd は関数として定義し，外部ライブラリである moment-with-locales.js に用意されているオブジェクトとメソッド moment.locale を用いて「ドイツ語ロケール」を設定する。Locale とはソフトウェアに内蔵される「言語や国・地域ごとに異なる単位，記号，日付，通貨等々の表記規則の集合」のことを言う。多くのソフトウェアやプログラミング言語では，使用する言語とともにロケールを設定し，ロケールで定められた方式に基づいてデータの表記や処理を行うことができる。

次に document オブジェクトに用意されている getElementById メソッドにより，id 属性値が cetd である HTML 内の div 要素を取得する。id の値は 1 つの HTML ソース内で一意（Unique）でなくてはならない。HTML ソース内の任意の要素にアクセスするこの仕組みを DOM（Document Object Model）と呼ぶ。DOM では HTML ソースに含まれる要素，属性，テキストといったものを全てオブジェクトと見なし，これら個々のオブジェクトを「ノード」（Node）と呼んでいる。DOM の各ノードには innerHTML というプロパティが用意されており，これは文字通りノード内部の HTML を表しているが，この値を操作することによりノードの HTML を動的に変更できるようになる。

これにより関数 updateCetd は，取得した id 属性値 cetd の div 要素に innerHTML プロパティを連結させることで，元々あった div 要素箇所を「innerHTML プロパティに代入される値」で置き換える。代入値の中にある

```
moment().tz("Europe/Berlin").
format("dddd, Do MMMM YYYY HH:mm:ss (Z z)")
```

という箇所は moment-with-locales.js および moment-timezone-with-data.js ライブラリで定められている記法に従った記述である。引数を指定しない moment で「現在の日付時刻」を協定世界時で取得し，tz（Time Zone）で協定世界時との時間差を「ヨーロッパ地域，ベルリン」の時間帯に合わせて調整した上で，format によるフォーマット（dddd は Freitag のような当該言語の完全表記曜日名，Do は日にちだが単なる 1, 2, 3 ではなく言語により 1st, 2nd, 1., 2. のようにフォーマットされる，MMMM は当該言語の完全表記月名，YYYY は西暦 4 桁，HH:mm:ss は 24 時間 2 桁表記での時：分：秒，Z は協定世界時との時間差，z はタイムゾーン）で出力する。なお，これらのライブラリで使うことのできる dddd といった「トークン」（Token; ソースコードに出現する文字列の中で意味を持つ最小単位）は非常にたくさん用意されているので，詳しくは Google 等で Moment.js Documentation/Docs/Display/Format をキーワードに検索して，そこにあるドキュメントを参照すること。

ここまでだとクライアント（閲覧者）がアクセスした時点でのベルリンの現在日時が静的に固定されて表示されるだけであるが，最後に setInterval メソッドを用いて updateCetd 関数を 1000 ミリ秒毎（つまり 1 秒毎）にリフレッシュさせることで，ベルリンの現在日時が「動的」に表示されるようになる。

第 14 講演習

演習用ファイル： euro_ict2b.zip を展開した一式

1. javascript.html ソース内の「日・独・仏・英語で「今日の日付」を表示する」に関する JavaScript ソースコードを良く分析して，Web ブラウザでの出力と照らし合わせよ
2. 福岡，Paris, London の format 箇所のソースコードを子細にチェックせよ

[*15] どうしてこのような命名となっているのか，語の連結に用いている符号から類推してみよ。

第 15 講

DOM を使ったイベント処理

Web ブラウザで閲覧される Web ページ上では，クライアントが「ボタンをクリックする」，「チェックボックスにチェックを入れる」，「マウス（ポインタ）を文字列の上に乗せる」，といった操作を行ったりするが，こうした操作をひっくるめて「イベント」（Event）と呼ぶ。JavaScript にはこうしたイベントを捉えてそれに対する処理を行う「イベントハンドラ」という仕組みが用意されていることは既に ICT IIB 第 10 講で述べた（50 ページ参照）。

最終講となる本講では，DOM でイベントハンドラを用いて処理を行わせる例を 2 つ取り上げ，講義を締め括ることとする。

1 つ目の「白雪姫テキストの多言語変換」は，id 値 khm53ja を持つ div 要素内に「日本語による白雪姫テキスト」を書き込んでおき，クライアント（閲覧者）が目的言語の「ラジオボタン」を「クリック」するごとに「独・仏・英・日」語へとテキストが切り替わる例となっている。これについては ICT IIB 第 10 講「フォーム」（49 ページ）において詳しく解説したのでここでは説明を繰り返さないが，本例ではイベントハンドラの設定法を 2 種使っていることだけ注意を促しておきたい。

先ず「オブジェクトのプロパティ」としてイベントハンドラを設定する方法であるが，これは javascript.html ソース内で window.onload = init; とある箇所で用いている（HTML ソースが Web ブラウザに読み込まれると init 関数が呼び出される）。もう一つは HTML 要素の属性としてイベントハンドラを設定する方法で，これは form 要素内 input 要素の onclick 属性で使っている（ラジオボタンがクリックされると tlTextTo 関数が呼び出される）。

DOM を使った画像の切り替え

DOM でイベントハンドラを用いて処理を行わせる 2 つ目の（最後の）例は，クライアント（閲覧者）が画像の上でクリックすると別画像が現れる仕掛けとなっている（全 3 画像）。

javascript.html ソース内には，id 属性値 chgImg を持つ img 要素 src 属性値にデフォルトで表示する画像の在処を相対パス表記で記し，onclick イベントハンドラを使って，画像がクリックされたら chgImg() 関数を呼び出すようにしておく。画像は style 属性値に max-width プロパティを用いることで，オリジナルサイズを超えて描画されない（画像が大きくなりすぎて解像劣化しない）ようにもしてある。

併せて script 要素内の当該箇所を見よ。Array コンストラクタ経由でまず空の配列 img を作る。次に Image コンストラクタ経由でこの中に「添字」（Index）とともに「まだ中身を指定していない画像」オブジェクトを格納していく。こうして img は複数の「要素」（Element）を持つ画像の配列オブジェクトとなる。なお添字は 0 から始まることに注意。画像が 3 つであれば添字は 2 までとなる。

配列となった画像オブジェクト img における画像ファイルの在処は src プロパティに格納されている。src プロパティはそのまま HTML の img 要素の src 属性に対応する。従って，添字で要素を特定した img オブジェクトに src プロパティを連結し，プロパティの値を別画像の在処に変更することにより画像を切り替えることができる。

img 配列内の各要素（3 つの画像）には添字（0, 1, 2）をキーにしてアクセスできるから，後は添字を 0, 1, 2, 0, 1, 2 と回していくカウンタを作ってやれば良い。そのためには先ず変数 cnt を宣言し，初期値を 0 とする。そして onclick で呼び出す chgImg() 関数を定義する。関数の中身は「cnt が 2 となれば cnt をゼロに戻す，そうでない場合は cnt の値を 1 つずつ増やしていく」という条件分岐を if/else 命令で記述し，この cnt 値をそのまま img オブジェクトの要素番号に代入することで HTML の img 要素内 src 属性値を書き換える，というものである。

第 15 講演習

演習用ファイル： euro_ict2b.zip を展開した一式

1. 「画像の切り替え」箇所の JavaScript ソースコードを自分でも解読してみよ
2. 今まで学んできたことを活かして，自らの多言語仕様 Web ページ（Web サイト）を作成，公開せよ

おわりに

Ajax

「Google マップ」をこれまで全く使ったことがない，という人は現在では稀ではないだろうか。この Google マップは Ajax（Asynchronous JavaScript + XML）という技術を基に作られている。Ajax とは，JavaScript を利用して Web サーバと非同期通信（Web サーバと通信を行っている間 Web サーバ側の処理完了を待たずともクライアント側で操作を続けられるような通信）を行い，受け取った結果を DOM 経由で Web ページに反映させる仕組みである。最後の XML（eXtensible Markup Language）とは，HTML 同様，タグを使ってデータ構造を記述するテキストベースのマークアップ言語であり，Ajax では Web サーバとのやりとりに XML 形式のデータを使い，XML 形式のデータを処理するのに JavaScript が用いられる。

Ajax 技術を用いるには JavaScript で XMLHttpRequest オブジェクトを使う。しかし，これを Web ページに実装するには「クライアント側で動作するファイル（例えば HTML）と Web サーバ側で動作するファイル（例えば PHP）」の 2 つが必要になる。クライアント側で Web サーバ環境を準備できなければ，Ajax を使って作成したプログラムの動作確認をクライアント側で行うことはできない。

本講義では Ajax を使った Web ページの具体的な説明はしない。しかし, jQuery も含め, ごく簡単なサンプル・ページ（Ajax を用いた非同期通信のテスト，jQuery を使った画像のアニメーション処理, Boustrophedon, Rongorongo）を演習用ファイルに同梱（ajax_jquery.html）しておいたので，興味のある者はソースを確認して欲しい。ただし，このサンプル・ページでは Web サーバ側で動作する PHP プログラム（ajaxTest.php）を呼び出して使用しているため，繰り返しになるが，PHP の動く Web サーバ環境がない限り，Ajax プログラムの挙動をクライアント側でチェックすることはできない。Ajax を含めた全ての「ヨーロッパ学 ICT IIB サンプル・ウェブ」の動作を確認したい場合は，代わりに

```
https://lgs.hum.fukuoka-u.ac.jp/~ynagata/ict2b/
```

にアクセスすること。LG サーバ上では PHP も稼働しているので，ajax_jquery.html で用いている Ajax プログラムの挙動も確認できる。

Ajax は操作性やユーザビリティの改善（待たされない，Web サーバが処理中であってもクライアント側では操作を継続して行える）やパフォーマンスの向上（Web ページ全体ではなく必要部分のみを更新するので通信量を少なくでき，その結果通信スピードも上がる）という点で大きな可能性を秘めた技術であるから，受講生は向上心をもって学んでいって欲しい。

jQuery

JavaScript に限らず，プログラミング言語には便利で汎用的な機能をまとめてこれらを簡単に呼び出せるようにした「ライブラリ」と呼ばれる補助的プログラムの集合体が存在している。本講義の中でも moment-timezone-with-data.js, moment-with-locales.js, platform.js, w3.css, all.css, font-awesome-animation.css, access_fn.css といった外部ライブラリを扱ってきた。

数ある JavaScript 用ライブラリ，また Ajax 対応ライブラリの中でも特に人気が高いのが jQuery である。jQuery を使えば「JavaScript による多くの処理をもっと短いシンプルなコードで記述できる」ようになる。華やかな視覚的効果を狙ったインタラクティブな Web サイトを開発するための jQuery UI のような拡張プラグインも提供されている。しかし本講義においては，jQuery ではソースコードの記法が標準の JavaScript とはかなり異なってくるため，jQuery を扱わなかった。ただし，Ajax 同様，jQuery を用いたサンプル・ページを用意したので，興味のある者はそちらをチェックすること。なお jQuery を用いたプログラムは，Ajax とは異なり，クライアント側で動作を確認できる。

jQuery も Web サイトから無償でダウンロードして利用できる。受講生は，各自の興味に従い，是非有効活用して欲しい。compressed と uncompressed の 2 種類があるが，ひたすら実用目的のみに使うのであれば前者だけをダウンロードすれば良い。ソースコードから改行，スペース，インデント，コメント等が全て削除され，徹底的に軽量化が図られている。コメント等の付いた後者は「中身を解読」したい人向けである。jQuery を使うには HTML ソース内に

```
<script src="./js/jquery-3.3.1.min.js"></script>
```

のように書いておけば良い。こうしたライブラリの読み込み記述は通常 head 要素内に書かれるものだが，読み込みのスピードを重視し，body 要素終了タグの「直前」に書くことを勧める人もいる。

それでは，引き続き，楽しく充実した「テキスト主義 ICT」ライフを！ 平素のドイツ語，フランス語，さらには言語全般の勉強そして研究に「ヨーロッパ学 ICT」で学んだスキルを活かし，レポートであれ論文であれ，勉強や研究の成果を「説得力がありユーザビリティの高い電子形式」にまとめられることは，そうした成果にさらなる価値を付与してくれるであろうことを確信している。

学習用参考文献

[1] Debra Cameron, James Elliot et al., 半田剣一, 宮下尚 ほか.『入門 GNU Emacs』第 3 版. オライリー・ジャパン, 2007.

[2] こもりまさあき（監修）, 赤間公太郎, 原一宣.『HTML5+CSS3 の新しい教科書』. エムディエヌコーポレーション, 2014.

[3] 大津真.『3 ステップでしっかり学ぶ JavaScript 入門［改訂 2 版］』. 技術評論社, 2017.

[4] 奥村晴彦, 黒木裕介.『［改訂第 8 版］LaTeX 2_ε 美文書作成入門』. 技術評論社, 2020.