情報システム技術論'02 ＃1

久野　靖 *

2002.5.23

1　はじめに

□ 本科目の主旨→「現在の情報システムにおいてどのような技術が使われているか」

- このようなテーマで網羅的に喋るのは大変なので、開講（隔年）ごとにテーマを絞って取り上げるようにしている
- 今年度は「Web ユーザビリティ」を中心として Web システム開発の周辺から話題を選んで行く

□ 各回の進め方→レクチャー、（必要に応じて）実習体験、分担しての報告、議論

□ 成績報告→出席、各回での報告の分担、レポート

□ スケジュール

```
 5/23 #1 ユーザビリティとは/Web ページ技術/宿題
(5/30 久野出張につき休講)
 6/6  #2 ユーザビリティの具体例（報告)(以下未定)
 6/13 #3
 6/20 #4
(6/27 #5 初回に日時決定)
```

2　World Wide Web

2.1　インターネットの歴史

□ インターネットの起源→ 1969 年（ARPANET）

- 最初は非常に限られた世界

□ インターネットの研究機関への普及→ 1980 年代

- CFNET(1981)、NSFNET(1987) →研究機関が多くネットに接続
- 日本でも 1988 年 WIDE Internet により TCP/IP 接続が開始

□ インターネットの社会への普及→ 1990 年代

- IIJ →日本で最初のプロバイダ→ WIDE 関係者が中心となり 1992 年末設立

*筑波大学大学院経営システム科学専攻

2.2　初期のインターネットの利用方法…

□ 電子メール→現在でもメインのサービス

□ ネットニュース→ Web 掲示板の利用が増えたためマイナー化

- メール、ニュースは IP 接続以前から利用できていた

□ telnet →他のマシンに接続して利用するのは重要な利用形態だった

□ FTP → telnet と対にして自分のアカウント間でファイルを転送するのに利用

- 匿名 FTP →フリーソフトの配付手段として

□ 情報公開/共有の仕組みとして…

- 匿名 FTP ではファイルの置き場所が分からないと役に立たない
- ネットニュースや「匿名 telnet」の利用
- goher →メニューベースで情報を次々に見ることができる
- WAIS →キーワード検索で問い合わせに適合性の高い文書をリストアップ

□ これらはいずれも

- 別個のサービスであり、
- 個別の特性を知っていて使い分ける必要があり、
- それぞれを活用するためのコマンドも全く別のものだった
- →インターネットから情報を取り出すのは熟知した人にしかできなかった

2.3　World Wide Web のはじまり

□ 「World Wide Web とは何か」定義できます?

□ CERN(欧州粒子物理研究所) の研究者 Tim Berners-Lee が 1991 年に開発

- 研究者どうしの情報流通をスムーズにするための手段として作った
- 主要なアイデア→ハイパーテキスト、URL によるポインタ
- WWW サーバ→ CERN httpd
- 当初は文字のみのブラウザ、NeXT Station（マイナーなシステム）で稼働

□ Mosaic --- 最初のグラフィカルブラウザ

- イリノイ大 NCSA の学生 Marc Andressen が 1992 年末に開発
- 1993〜1994 に爆発的に広まる（最初は Unix/X →後に Mac、Windows 版も）
- 当然、最初は英語のみ→高田敏弘@NTT が Mosaic-L10N を開発→日本語も可能に
- 計算機研究者の間で大ブームに（1994 ころ）
- NCSA は NCSA httpd も公開

□ イリノイ大の学生グループはその後も Mosaic の開発を続けていたが…

- 大学は Mosaic を自分のところのソフトとして金儲けしようとした
- 自由にやりたい学生グループとの対立

□ Sillicon Graphics Inc.（SGI）の創立者 Jim Clark は WWW が商売になると考え、Netscape Communications 社を設立して Marc Andressen ほか学生グループの主要メンバーを引き抜く（1994）

- Netscape ブラウザ→圧倒的な支持を得てブラウザのシェア No. 1 に
- 最初から Unix/X、Windows、Mac で稼働
- Netscape 1.1 からは日本語も OK →日本でもインターネットブームに

2.4 ブラウザ戦争

□ Microsoft 社は当初は MSN(自前のパソコン通信) に顧客を囲いこもうとしてアンチインターネット路線だった

□ インターネットがもはや主流だと分かると路線転換→自前のブラウザ Microsoft Internet Explorer(MSIE) を普及させて Netscape 支配を止めようとした

- MSIE を Windows に標準付属（Win 95〜）、しかも OS 組み込みにして取り外せないように（Win 98〜）
- PC を出荷しているメーカに圧力を掛けて Netscape ブラウザを付属させないようにした（→訴訟の対象となっている行為の 1 つ）
- さまざまな「新技術」を取り入れて差別化を計る（どれくらいそれが「有難い」かどうかは…?）

□ Netscape 社はブラウザでは食べて行けなくなり、ブラウザは無償化（これまで儲かっていたビジネスが突然なくなるとは…）

- Netscape 4.x の次のブラウザはオープンソース開発モデルにより開発するとして、1998 年にソースコードを公開→ Mozilla プロジェクト
- そうしている間にも経営は苦しくなり AOL により買収
- Mozilla プロジェクトの成果をもとに Netscape 6 を公開（2000 年末）
- 現在も Mozilla(フリーソフト) と Netscape 6.x は並行して開発

□ MSIE →圧倒的シェア、標準への準拠はそこそこ、よく分からない独自機能、よくセキュリティ上の問題が発見される

□ Mozilla/Netscape 6 →よりよく標準へ準拠、シェアは小さい

□ Netscape 4 →メンテナンスのみ、企業サイトでは今でも使われている

□ あと、i-mode とか組み込みブラウザも増えている…

□ 結局、ブラウザは「WWW への単なる接続口」「WWW を見る道具」という位置付けに…

3 WWW と標準

3.1 WWW と標準の関わり

□ WWW が始まった時にまず作られたもの…

- HTML →コンテンツ（Web ページ）のためのマークアップ言語
- HTTP → Web サーバからコンテンツを取り寄せるためのプロトコル

□ 当初は Tim がとりあえず作ったものを皆で改良→協調的

- HTTP 1.0 や HTML 2.0 →改良された標準を RFC 化して標準化

□ Netscape 1 あたりから、「独自機能を増やしたものがメシの種になる」状況に→競って新機能を追加

- とくに MSIE と Netscape ブラウザの機能は IE 4.x と Netscape 4.x ではかなり相違がある
- 標準がなければ「誰でもどこでも見られる」という肝心なところが保証されなくなってしまう

□ WWW Consortium(W3C) → Web で利用する標準のとりまとめを行なう組織

- しかし取りまとめも簡単ではない
- 各ブラウザベンダーは自社の「独自機能」を「標準」にさせようと運動
- まあそれなりに機能

3.2 HTTP

□ HTTP/0.9 → Tim がとりあえず作ったバージョン

□ HTTP/1.0 → RFC 1945 (1996 年)

- 非常に簡単なプロトコル。サーバに接続して、取り寄せたいものを言うと送られて来て接続が切れる。
- 簡単なのはよいが毎回接続を張り直すのは無駄→ keep-alive 拡張

□ HTTP/1.1 → RFC 2068 (1997 年)

- プロキシ機能や接続の維持に関する機能の拡張

□ HTTP はデータ転送なのであまりもめることはない

- HTTP 1.1 のサーバ/クライアントと HTTP 1.0 のサーバ/クライアントでの接続も可能なように配慮されている

3.3 HTML

□ HTML 1.0

- Tim がとりあえず作ったバージョン、最小限の記述能力
- SGML(Standard Generalized Markup Language) の標準を参考にしていたが、完全に準拠はしていなかった

□ HTML 2.0

- 上記の問題を改良し、IETF がとりまとめたもの。RFC 1866 (1995 年)
- 最初の広く普及したバージョン
- フォームはあったがフレームやテーブルはなかった

□ この後、Netscape や MSIE によるさざまな「独自拡張」が表れる

- ... 、<blink>...</blink>、<marqee>...</marquee>
- W3C が HTML 規格の取りまとめを試み HTML 3.0 を提案する
- しかし HTML 3.0 と各ブラウザの仕様が大きく隔たっていてまとまらない

□ HTML 3.2 ←とりえあず普及したテーブル、フレーム機能などの機能を取り込んで「共通部分」をとりまとめたもの (1997 年)

- <font>タグや align 属性など表現に関わる機能も多くまざっている
- HTML 3.2 はとりあえずの標準として広く使われた
- W3C としては「構造と表現の分離」をめざしていた → HTML 3.2 は妥協

□ HTML 4.0 → HTML 3.2 の後継かつ W3C のめざす HTML の最終版 (この後は XML ベースの XHTML になる)

- HTML 3.2 互換の HTML 4.0 Transitional、厳密な HTML 4.0 Strict に分かれている (このほかフレームセット用の DTD もある)
- HTML 3.2 に入っていた表現関係の機能は多くが「非推奨」となった
- その代わりスタイルシートを用いるための仕掛けが用意され、表現はスタイルシートにより指定することとなった

3.4 そのほか…

□ 大量にあるので名前だけ。あとで必要に応じて取り上げる

- (see http://www.w3.org/TR/)
- CSS(Cascading StyleSheet) →スタイルシート記述の標準
- XML(eXtensible Markup Language) →タグを必要に応じて定義できるマークアップ言語、以下の多くの言語が XML ベース
- XSL(eXtensible Stylesheet Language)、XSLT(XSL Transfromations) → XML にスタイルをつける、XML を加工
- RDF (Resource Description Framework) →コンテンツの「内容」を記述する枠組み

- MathML（Mathematical Markup Language）→数式記述
- SVG（Scalable Vector Graphics）→図形記述
- SMIL（Synchronous Multimedia Integration Language）→実時間マルチメディアを組み合わせて制御
- XHTML →XML に基づく HTML
- DOM（Document Object Model）→ドキュメント内容等をオブジェクトとして取り扱う

4 Web 開発とユーザビリティ

4.1 なぜ「Web ユーザビリティ」か?

□ WWW が始まったころは「ページがあるだけでよかった」

- HTML で書くだけでそれなりの技術が必要で価値があった
- 内容の見せ方より内容が世界中から取れることが重要だった

□ まもなく、みんな「見せ方」を気にするようになった→表現指定など

- 最初は作る人の「自己満足」だった→今でも個人のページは自己中なものが多い
- 見てもらうことを飯の種にする「プロ」の出現
- 見てもらうにはどうしたらいいか→ユーザビリティが重要に

4.2 ユーザビリティとは?

□ 「使いやすさ」

- 気分の問題なの?

□ ある目的に対して、その目的を遂げる「効率」

- 効率を何で計測するのか→時間、操作数、疲労、…

4.3 Web ユーザビリティの特徴的な部分?

□ これまでのハード製品のユーザビリティは「買った後で分かる」

□ ソフトではユーザビリティが悪いとサポートコストがかさむ

- しかしそのコストは別建てだと思って開発されてしまいがち

□ Web ではまったく状況が違う

- Web ではまずユーザビリティに直面→その後で金を出すかどうかが決まる
- 膨大な Web サイト群→ちょっでも嫌なら「よそへ行く」

4.4 ユーザビリティと見た目は相反するか?

□ 一般的に言えば「両立は可能」

□ しかし、見た目ばかりにこだわってユーザビリティが犠牲になっている例は多くある（たとえば?）

□ 例: 画像を使うこと

- 画像を使うだけでその分だけ確実に重くなる（対策?）
- 画像はテキスト検索できない（対策?）
- 画像はレイアウト（幅など）を固定化する（対策?）

□ 例: 表レイアウト、色やフォントの指定

- 表でないものを「配置するためだけに」表としない方がよい
- 色やフォントや配置はスタイルシートを使う→それらを解釈しないブラウザでもそれなりに見える

□ で、見た目のよいデザインは重要

- それはデザインセンスなのでここでは扱わないが…
- ユーザビリティの原則を守った上でデザインしよう、ということ

□ 世の中では全然そのことが実践されていない（YN の主張）

4.5 なぜ間違ってしまうのか?

□ Web は違うということを理解していない

- Web ページはカタログの代わりだと思っている
- ちょっと情報を調べようとすると「スキャナでスキャンした PDF」にぶちあたったりする…

□ プロジェクト管理がされていない

- Web 製作は「つぎはぎ」ではなくソフトウェア開発のようにきちんと管理されているべき
- しかしソフトはつぎはぎでは動かないが、Web はとりあえず見えてしまうからなおざりに?

- 統一されたユーザビリティのためには統一された開発プロセスが必須

□ 発信側の都合による構造化をしてしまいがち

- 会社の部門ごとに対応してページを分けるとか…
- ユーザは会社の構造に従ってナビゲートしたいわけではない!

□ 見た目重視にしてしまう

- コンペやデモをやると見た目で選んでしまいがち
- 実際には見た目以外に重要なことが沢山ある

□ 長々と文章を書いてしまう

- Web 上では長い文章をじっくり読んでもらうことは期待できない
- 斜め読みに適したページ構造を設計するべき
- 詳しい情報は印刷して読むためのものを別に用意

□ 自分のサイトが孤立した/自分が構造化したままの世界だと思ってしまう

- Web はどこからどこへでもリンクが張れることが特徴!
- 検索エンジン等で「途中へ飛び込んで来る」ことも当り前
- よそのサイトで有用なところへもリンクを張るべき
- ユーザのナビゲーションのしかたを制約することは不可能 (ブックマークなど最たるもの)

5 ページデザイン

□ 実はサイトデザインの方がずっと重要だが…

- とりあえず具体的で分かりやすいテーマなので

5.1 陣取り合戦

□ ページの中で「本当のコンテンツ」がどれくらいあるか

- 下手なデザインだと肝心のコンテンツが 1〜2 割とか→「めやす」として 50%はコンテンツが占めて欲しい
- うまく空白を取ることも大切ではあるが「利用しそこない」はよくない
- 広告スペースなどが頭にあると難しい
- 余分な要素は削ってしまうべき

□ しかし言うのは簡単だけど…

□ まずページを開けた時に見える場所→1 等地→限られている

- ほとんどのユーザはスクロールなんかしない! (ホント?)
- ウィンドウサイズ (←モニタサイズ) によって 1 等地の大きさは変化
- ではどうすればいい?

□ 「このページは 1024x768 で見てください」→サイテー

□ 画面の大きさに応じてデザインを変更→夢物語? (今でも少しはできる)

□ 複数の典型的なモニタサイズでチェックする→○

- 結局、複数のサイズでうまく扱えるデザインを工夫する

5.2 プラットフォーム独立

□ 従来のデザイン→すべてのピクセルをデザイナが制御可能

- プログラムの動作はプログラマがすべて自由にできる (例: ある質問に回答するまでは先に進めない等)
- さらに OS メーカがデザイン指針を提供

□ Web でも同様のことをすればいいのか?

□ そもそも、Web では「どのプラットフォームか」が分からない

- ユーザによって使っているプラットフォームが違う ← Windows 決め打ちとかはやめて頂きたい
- ユーザを制御することはやめた方がよい←ユーザにとっては腹立たしいだけ

□ Web の難しさ…

- GUI ではそれが一般化する前に研究する時間が十分あった
- Web では「いきなり始まってしまった」から…

5.3 ユーザはどこから?

□ 画面の解像度 (VGA、SGVA、XGA、SXGA、…)

- 画面サイズは大きくなる傾向
- PDA などだとまた小さいものが増える

□ 接続速度

- 8M ADSL から 32kbps Air H"まで（もっと遅いのも?）

□ Web デザインに「WYSIWYG」はナイ

- WYSIWYG 風ツールを使うとそれがアルと誤解してしまいがち
- デザイナは「自分がすべて制御可能」に慣れている
- ブラウザ決め打ち、画面サイズ決め打ちでデザインしたくなる罠

5.4 解像度独立

□ どういうことに注意?

□ なにごとも「固定幅」で指定してはいけない

- 代わりに「画面幅のパーセンテージ」を使う

□ 文字サイズも固定フォントサイズでは絶対指定しない

- そうされてしまうと高解像度画面や目の弱い人が困る
- 代わりに大/中/小とか「標準サイズの何パーセント」を使う

□ アイコンも小さくなっても分かるデザインに

- アイコンに文字を入れる場合には特に要注意←普通は文字は入れない方がよい

□ 広い画面ならそれが活用できるように

5.5 あまり使われていない技術…

□ とは、どんなもののこと?

□ たとえば次のものは?

- 表?
- フレーム?
- レイヤー?
- スタイルシート?
- JavaScript?
- DOM?

□ 技術の変遷はかなり激しい

- あまり狭く考えると可能性が縛られると思う
- W3C 標準か、特定ベンダー独自か
- 有効でない場合でも害をもたらさない、代替手段があることが目安

5.6 インストールの法則

□ 現在では新しいブラウザへの入れ替わりは活発でない

- 新しいマシンを買ってついてくるブラウザを永遠に使う

□ 新しいブラウザでユーザが得る利益は小さい（とユーザは考えている）

- そもそもインストールなんてしない???

□ だったらどうするか…

- 新技術→ 1 年は待つべき
- 新しい技術でなければできないかどうか? NO なら無理に使わない

5.7 用途と見せ方の区別

□ …つまり semantic encoding を使おうということ

□ スタイルシートを使おう

□ 「データは永久に死なない」

- 50 年後にあなたのページがちゃんと見られるかどうか?

5.8 応答時間

□ クリックして「すぐ反応がある」と思うのはどれくらい?

□ 反応が遅れてもまあ待ってよいと思うのはどれくらいの時間?

□ 心理学の成果

- 即座に反応していると思うのは 1/10 秒が限度。それを越えたら「待たされている」と思う
- 応答が 1 秒を越えたらユーザはもはや待てない→どっかよそに行かれてしまっても文句は言えない
- →ではこれらを越えそうならどうする?

□ 対策…

- 「応答を用意している」というふうな応答をまず返す
- 「どこまで進んだ」というフィードバックを返す
- 「あとどれくらい掛かる」という情報を返す
- 常に同じ時間待たせる（予測可能にする）

□ 対策はさておき…

- ユーザは「きれい」より「速い」がずっと好き
- 「1 秒以内」にすることでユーザがよそへ行く率を劇的に減らせる
- 意味のない画像や大きな画像を使うべきではない
- 小さい画像でも見た目よくするのは難しくない

□ 作者が試している分には速い罠

□ 「http://w3in/bs」ではなく「http://w3in/bs/」としよう

5.9 一覧性

□ 出て来たページですぐ欲しい情報が分かることが重要

□ トップページはとくに「画像がなくても分かる」ように

- ALT 属性を使って画像 OFF の人でも意味があるようにする

□ 表や画像のようにダウンロードが終わるまでは整形できないものは避ける

5.10 リンクの使い方

□ リンクにはどのような使い方があるか?

□ ナビゲーションリンク→サイトの構造に従った順次移動

□ テキスト内リンク→関連のあるものを指す

□ 参照リスト→関連するものとしてこれがありますよ、という形

5.11 リンクテキスト

□ 「テキスト」の「テキスト」の部分

□ 「ここをクリックしてください」で「ここ」をリンクにする…ありがちだけどサイテー。「here 症候群」→なぜか?

□ here 症候群が悪いわけ

- ユーザによってはクリックするとは限らない
- リンク先に何があるかは行ってみないと分からない
- 印刷したときにわけが分からない
- じゃあどうするのがいいの?

□ 「○○の画像も見られます」のように普通の文章にして、「○○の画像」のところをリンクにする

- リンクした先になにがあるかを分かるようにするのが重要

5.12 リンクタイトル

□ 「...」で説明文がポップアップするようにできる

- 余分な害をもたらすことが小さいと思われる
- あまり長くしすぎない方がよい
- リンクタイトルが表示されないブラウザもあることに配慮しておく

5.13 リンクの色

□ 標準では「青で下線」→訪問していないリンク、「紫で下線」→訪問すみのリンク

- ユーザインタフェース的にいえばこれらの色は必ずしもよくないが、既に定着しすぎているので変更しようがない

□ body タグの指定やスタイルシートでこれらの色を変更することができるが…変更するとユーザの困惑を引きおこすので避けるべき

□ 関係ないテキストに上記のような表現を持たせることも避けるべき

5.14 リンクのユーザビリティ

□ 出発のレトリック→なぜ現在いるページから離れてよそへ行くか納得してもらう

- さらに詳しい (〜に関する) 情報があります、という形で明示
- よそへ行かせたくないと思ってリンクを張らないのはよくない
- よそのどんなサイトに張るかを厳選して適切に張る→もとのページの価値も高めることになる

□ 到着のレトリック→新たにやってきたこのページはどういう価値を持つかが納得してもらえるようにする

- このページは何なのか、がはっきりわかるようにする

□ 新しい窓を勝手に開いたり窓を最大化するようなリンクはサイテー。

- ユーザは自分で画面の状況を制御したい。それを邪魔するようなサイトは嫌われて当然。

5.15 外部からのリンク

□ 外からリンクを張ってもらう→自分のサイトのページビューが増えるのでありがたいこと

- じゃあ、どうすればいいのか?

□ 魅力のあるページを作ること! アタリマエすぎだが…

□ URL を固定する→途中にリンクされてもその内容がちゃんと生きている

- 「途中への直リンおことわり」サイトがあるがこれもサイテー
- 構造を整理するとき URL を変えないというのは結構苦しい

□ 1 つのページが複数の URL を持つこともあり

- 例: 「最新コラム」と「×年×月のコラム」
- 強制的にどっちかの形に揃えてしまうこともできるが…
- どちらの形でリンクしたい場合も可能になっている方がよい

5.16 有料 (登録) サイト

□ 見る前に金を取るというのは嫌われる→そもそも見てもらえない

□ 無料で利用できる価値あるページも必要

- そこから「もっと見たければ登録を」という形で勧誘できる

5.17 宣伝リンク

□ バナー広告 (valueclick 等) →広告を入れておくとお金をはらってくれる

- 入れておくだけでよい場合と、その広告をクリックしてくれた数に応じて払う場合とがある
- どっちにせよ、ユーザにとってはジャマで嫌われもの
- クリックしてもすぐ戻られてしまう

□ 本当にユーザにとって役に立つ情報を提供するようにしないと…

- 従って、特定商品の宣伝ならクリックするとその商品のところへ行くようにしないと…

5.18 スタイルシート

□ CSS を使うことで意味と表現の分離が可能に

- 1 個所変更するだけで大量のページのスタイルを全部調整できる
- 文書の内容に応じて複数スタイルを使い分けてもよい
- これらの利点のためにも、分離型のスタイルシートを使うのがよい

□ スタイルシートは分かっている人が設計し適用しないとダメ

- 専門家がデザイン・運用するようにして、ページ執筆者は利用するだけにした方がよい
- その場合、どのように使うかというマニュアルが必要
- マニュアルには、さまざまな環境でスタイル指定がどういう働きになるかを例示するべき

□ 上記にも関わらず、ページ執筆者が埋め込みスタイルで必要なものを追加することは避けられない

□ 重要なスタイルシートの指針

- フォントは 2 つ以上使うな!
- 固定文字サイズは使うな!
- 「!important」は使うな!
- 同じコンセプトの要素には class 属性を使って共通化

□ 要するに、ユーザがやろうとすることを邪魔してはいけない

5.19 フレーム

□ 「使うな!」(ホント?)(どう思います?)

□ Web ではナビゲーションの単位と情報の単位が一致している

- しかしフレームを使ってしまうとこの一致が崩れてしまう
- たとえばブックマークをつけても再度来たときに同じ状態にならない
- URL を記録しておいても役に立たない→そんなのは Web ではない

□ フレームはある程度大きな画面サイズを必要とする

- 小さい画面では画面領域が貴重なのにフレームは勝手に領域を使ってしまう

- 「ユーザのため」といいつつユーザが本当に何が欲しいのかは分からないので、それに合わせたフレームも作れるわけがない
- ナビゲーションバーを別にするためにフレーム→そんなもののために画面を削らないで欲しいと思っても避けられない

□ まだまだある…

- まともにプリントできないことが多い。特定フレームだけ印刷? 全部をうまく印刷? スクロールするとどうなの?
- HTML ファイルが沢山必要で間違いやすく難しすぎる。バグのもと
- 検索エンジンにとっても難しすぎてうまく情報が抽出できない
- 現に「フレーム版」「フレームなし版」を両方用意すると…大部分のユーザは「フレームなし版」を選んでいる!

□ だからフレームなし、1 段組の縦スクロールがベスト!

□ フレームを大丈夫にする方法…

- 「target="_top"」を指定しまくる
- インラインフレームを使う
- しかしどちらもあまり嬉しくないと思うが…

5.20 サイトの信頼性

□ きちんとしたページ構成

□ 見た目も重要だが重くしない

□ プロとしての運用

- リンク切れや更新の怠慢がない
- 内容が正しい、適切

5.21 プリントの必要性

□ ページを印刷しておく人はとても多い

- しかし画面で見るのと印刷して読むのではニーズがかなり違う
- おすすめ→画面表示用と印刷用と 2 つのバージョンを用意する

□ 「<link rel="alternate" media="print" ...>」で印刷ファイルを用意できる

□ 印刷用に PostScript や PDF を用意するのはよいこと

- しかし、これらを画面表示用と兼ねては絶対にいけない
- 印刷サイズ→ A4 でもレターサイズでも印刷できないといけない

6 次回までの課題 (2 週間)

□ 2 章「コンテンツデザイン」読んでくる

□ 各自が 1 章ないし 2 章のトピックとして興味深いサイトを選び紹介する→それに基づいて議論

□ HTML+CSS が不如意な人は勉強しておいてください (5/24 計算機科学基礎でも取り上げます)