

Non eram nescius, Brute, cum, quae summis ingeniis exquisitaque doctrina philosophi Graeco  
**簡易マークアップとAH Formatterによる**

ReVIEW  
**商業品質を満たした**

philosophari. quidam  
**技術書籍制作フローの展望**

Graecis litteris, contemnentibus Latinas, qui se dicant in Graecis legendis operam malle consum-  
ere. postremo aliquos fucros suspicor, qui me ad alias litteras venent, genus hoc sciendi, etsi sit  
elegans, personae tamen et dignitatis esse negent. [2] Contra quos omnis dicendum breviter ex-

istimo. Quamquam philosophiae quidem vituperatoribus satis responsum est eo libro, quo a  
nobis philosophia defensa et collaudata est, cum esset accensata et iniurata ab Intensiis, qui  
liber cum et tibi probatus videretur, et illi quos ego posse iudicare arbitraber, plura suscepi veritus  
ne movere hominum studia viderer, retinere non posse. Qui autem, si maxime hoc p...  
eratus tamen id volunt fieri, difficilem quandam temperantiam postulant in eo, **株式会社 トップスタジオ**

missum coerceri reprimique non potest, ut propemodum iustioribus vitamur illis, qui omnino  
avocent a philosophia, quam his, qui debet inimicis modum conciliant in reaque e...  
**武藤 健志**

**@kmuto**

**Professional use?**

**先に結論から**

**AHFormatter + CSS組版**

**はまだ本格導入には難あり**

**(※当社で制作している書籍において)**

でも

**未来**

**は感じる**

# お話すること

- 当社の紹介
- Re:VIEW+InDesign DTPとその制約
- Re:VIEW+AH Formatterの状況と展望

# (株) トップスタジオ

- 「編集プロダクション」
- 出版社（版元）が顧客のBtoB事業
- メーカー等のカタログ、マニュアル、ガイドブック等も制作
- 「印刷所の手前まで」を請け負う、総合プロダクション
- 企画、**翻訳**、執筆、**編集**、検証、校正、**組版**、カバー/紙面デザイン、**電子書籍化**

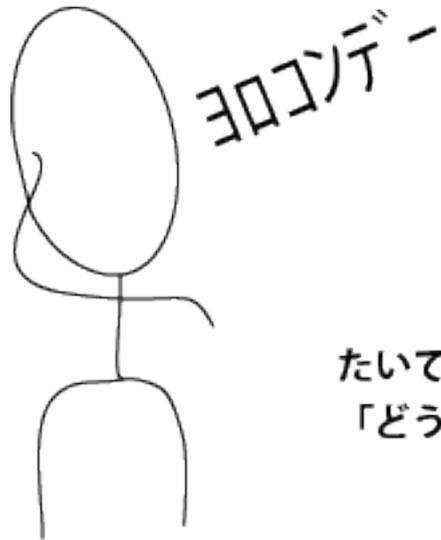
<http://www.topstudio.co.jp/>

# (株)トップスタジオ

- IT技術書
  - 技術書籍出版社の取引先多数
  - ソフトウェア、ハードウェア、プログラミング、ネットワーク、ソーシャル、モバイル、etc...
- 資格書
  - IT技術資格、福祉資格、通関士等
- カタログなど
  - Cisco社さま製品カタログ、大手メーカーマニュアル、英語辞書、文芸書 etc...
- BtoBではあるが、制作物は**Consumer**の目に触れることになる

# (株) トップスタジオ

- 顧客である**出版社の意向**が最重要



たいていの出版社にとっては  
「どうでもいいこと」の境界

「売り物」レベルの紙面

ともかく安く！

全部やっておいて

突貫スケジュール

電子書籍も一緒に

間違いのないものを

いつ初校出せるの

標準化

きれいな構造化

編集者のミス削減

効率的な組版手法

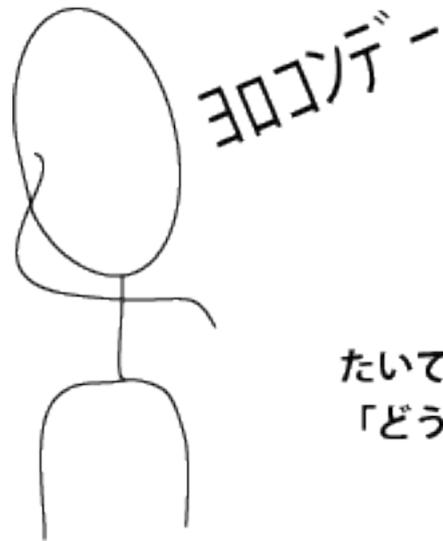
DTP オペレータの負荷削減

# 自己紹介

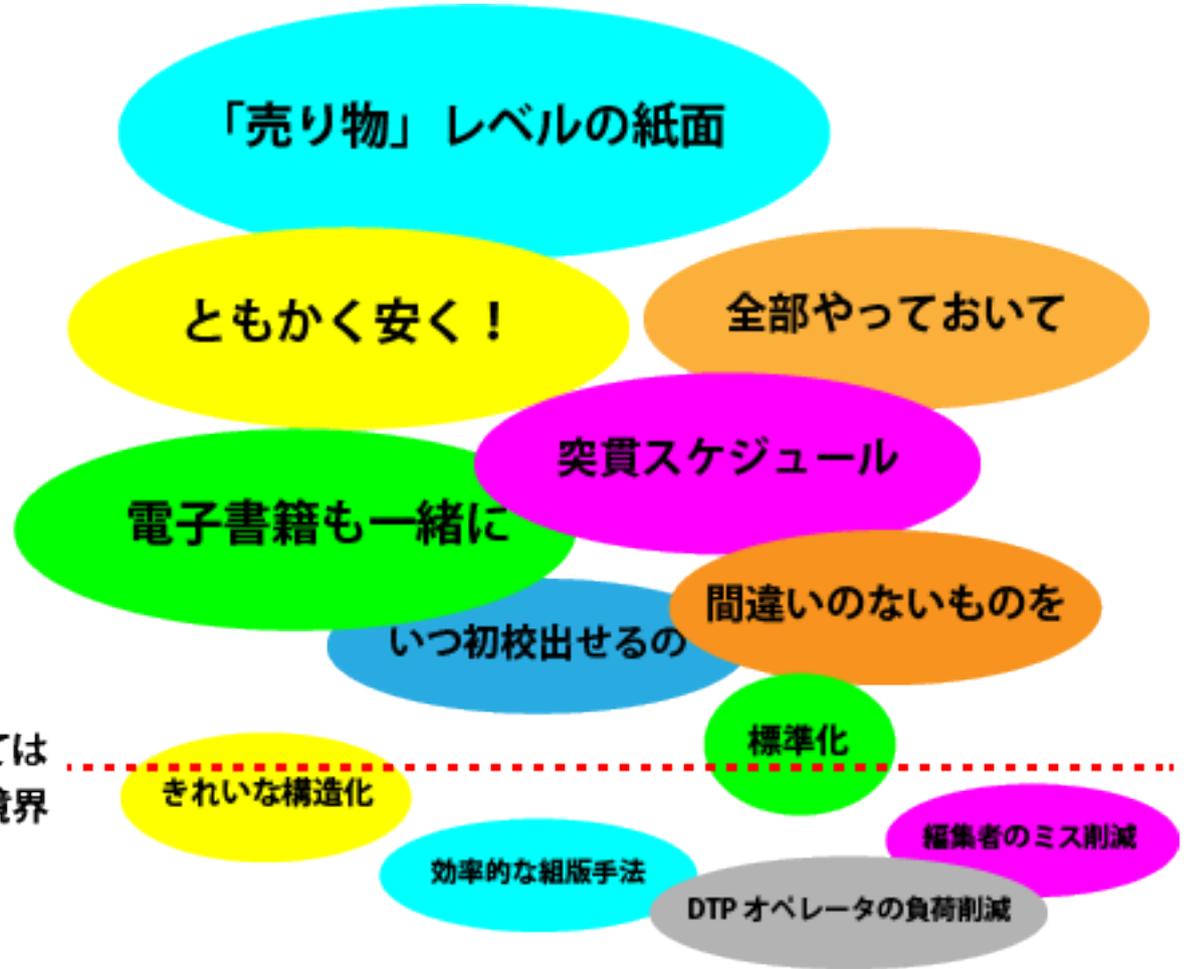
- 武藤 健志 (MUTO Kenshi、@kmuto)
- 株式会社トップスタジオ  
執行役員 (えらくないです)
- 編集、自動組版設計、  
プログラミング その他萬事
- 前職はSGML/XML処理開発
  
- Debian Project公式開発者
- 執筆、監修、翻訳
- TeX好き、ものぐさ

# 再掲

- 顧客である**出版社の意向**が最重要ではあるが...



たいていの出版社にとっては  
「どうでもいいこと」の境界



# ミッション

- 出版社の意向に応えるには...



# 制作手法の改善が必要

- InDesignファイルのようなバイナリファイルを**マスターデータ**にしない
  - DTPオペレータしか触れない代物に
  - EPUB化はちょっと凝ったデザインだけで破綻
- 原稿データを常に更新し、**文責のある人**  
(編集者 or 著者 or 訳者) が**直接**修正できる
- 原稿データから変換して最終的に組版  
(PDF) が**容易**にできる

# 制作手法の改善が必要

- ただし、
  - 紙面デザイン、要素の配置方法は本ごとに変わる
  - シリーズ書でも微妙に違ってたり
- **汎用性**があり、かつ**カスタマイズ**できる制作フローが必要

# 適切な原稿フォーマット

- では、原稿フォーマットに適切なのは何か？  
（※技術書籍において。）
- 独断と偏見と**ポジショントーク**です
- XML (InDesignに利用できる)、HTML (EPUB用)、テキスト（どうにもならなくなったときに従来のやり方で組版できる）に簡単に変換可能であること
- 特定の紙面デザインに限定されないこと

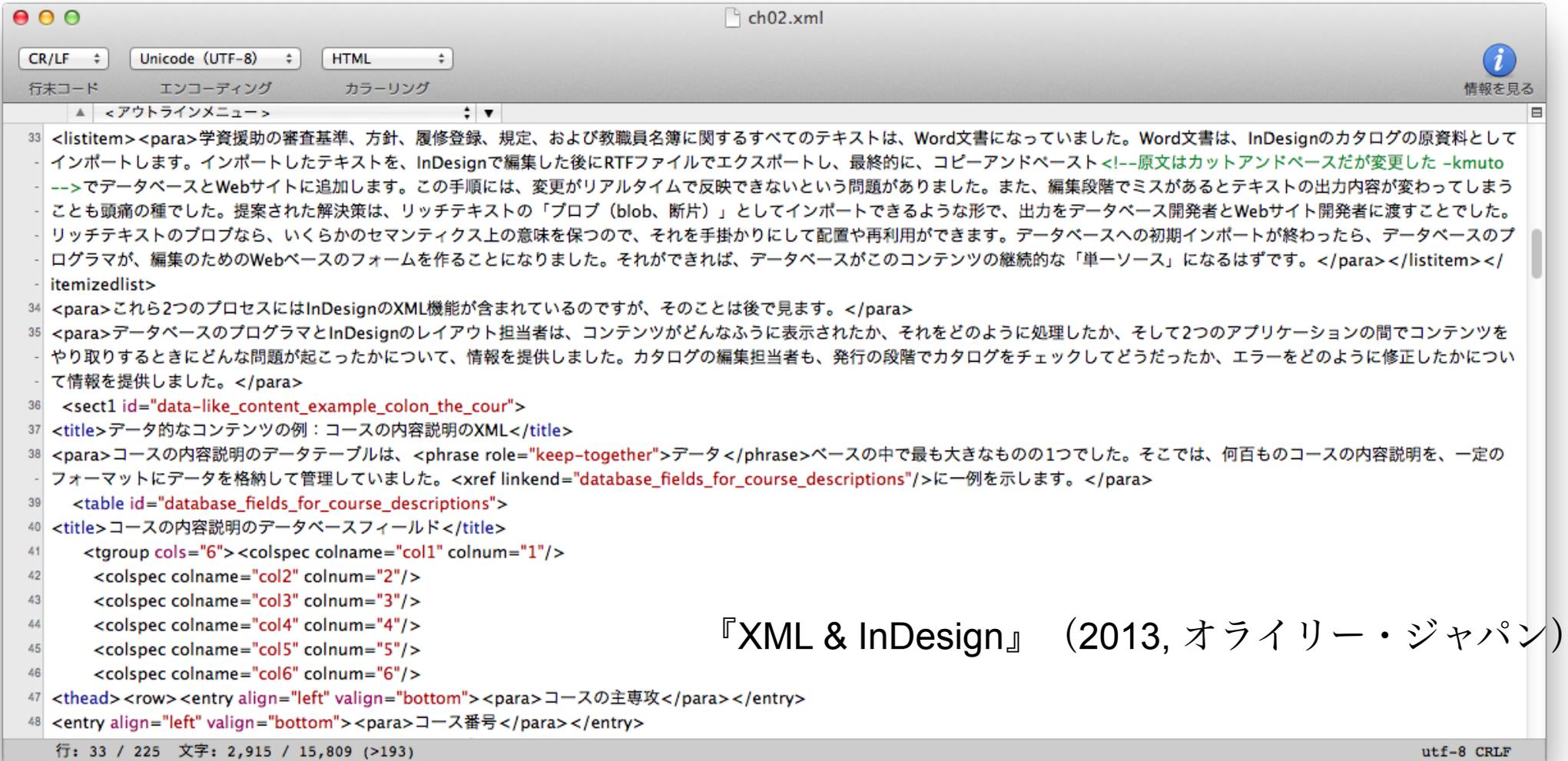
# 適切な原稿フォーマット – Word?

- スタンダード
- 構造化とは無縁
  - スタイル付けなどすごいがんばれば…でもそれちゃんと守れる？
- 組版の元データにするには厄介事が多い
  - 原稿を差し替えて組み直すのは非現実的

# 適切な原稿フォーマット - XML?

- 構造化文書の王道
- 拡張しやすい
- **マークアップ**（タグ）がたくさんで読みにくい、編集しにくい
- **メタ情報**が区別しづらい
  - 紙面には出さないが原稿には入れておきたい情報。コメントや作業連絡など
- 文字のエスケープが頻出する
  - `<→&lt;`、`>→&gt;`、`&→&amp;`;

# XMLがhuman readable?



```
33 <listitem><para>学資援助の審査基準、方針、履修登録、規定、および教職員名簿に関するすべてのテキストは、Word文書になっていました。Word文書は、InDesignのカタログの原資料として
- インポートします。インポートしたテキストを、InDesignで編集した後にRTFファイルでエクスポートし、最終的に、コピーアンドペースト<!--原文はカットアンドペースだが変更した -kmuto
- -->でデータベースとWebサイトに追加します。この手順には、変更がリアルタイムで反映できないという問題がありました。また、編集段階でミスがあるとテキストの出力内容が変わってしまう
- ことも頭痛の種でした。提案された解決策は、リッチテキストの「プロブ (blob、断片)」としてインポートできるような形で、出力をデータベース開発者とWebサイト開発者に渡すことでした。
- リッチテキストのプロブなら、いくらかのセマンティクス上の意味を保つので、それを手掛かりにして配置や再利用ができます。データベースへの初期インポートが終わったら、データベースのプ
- ログラマが、編集のためのWebベースのフォームを作ることになりました。それができれば、データベースがこのコンテンツの継続的な「単一ソース」になるはずです。</para></listitem></
- itemizedlist>
34 <para>これら2つのプロセスにはInDesignのXML機能が含まれているのですが、そのことは後で見ます。</para>
35 <para>データベースのプログラマとInDesignのレイアウト担当者は、コンテンツがどんなふうに表示されたか、それをどのように処理したか、そして2つのアプリケーションの間でコンテンツを
- やり取りするときにどんな問題が起こったかについて、情報を提供しました。カタログの編集担当者も、発行の段階でカタログをチェックしてどうだったか、エラーをどのように修正したかについ
- て情報を提供しました。</para>
36 <sect1 id="data-like_content_example_colon_the_cour">
37 <title>データ的なコンテンツの例：コースの内容説明のXML</title>
38 <para>コースの内容説明のデータテーブルは、<phrase role="keep-together">データ</phrase>ベースの中で最も大きなものの1つでした。そこでは、何百ものコースの内容説明を、一定の
- フォーマットにデータを格納して管理していました。<xref linkend="database_fields_for_course_descriptions"/>に一例を示します。</para>
39 <table id="database_fields_for_course_descriptions">
40 <title>コースの内容説明のデータベースフィールド</title>
41 <tgroup cols="6"><colspec colname="col1" colnum="1"/>
42 <colspec colname="col2" colnum="2"/>
43 <colspec colname="col3" colnum="3"/>
44 <colspec colname="col4" colnum="4"/>
45 <colspec colname="col5" colnum="5"/>
46 <colspec colname="col6" colnum="6"/>
47 <thead><row><entry align="left" valign="bottom"><para>コースの主専攻</para></entry>
48 <entry align="left" valign="bottom"><para>コース番号</para></entry>
行: 33 / 225 文字: 2,915 / 15,809 (>193) utf-8 CRLF
```

『XML & InDesign』 (2013, オライリー・ジャパン)

# 適切な原稿フォーマット - HTML?

- XMLより覚えることは少ない（マークアップの種類が少ない）
- Webブラウザで確認できる
- 拡張性は低い
- HTMLだからといって、紙とEPUBでの併用はあまり現実的でない
- メタ情報を入れにくい
- 章節項番号や図表番号等の採番機能がない
- HTMLBook?

アンテナハウスさんのアピールとは相反するかも...

# 適切な原稿フォーマット - その他

- TeX
  - 完全バッチ型、数式表現や採番処理など優れた機能
  - 紙面化を含んだフローであり、その紙面デザインおよびスタイル作成に高度な技術を要する
  - XML化やプレインテキスト化は難
- Markdown、Wiki
  - 単純なマークアップ
  - githubの人気とともに最近広まりつつある
  - 相互参照や採番機能の不足、マークアップと地の文の区別がしづらい
    - ※pandocはまあまあいいかも

# 適切な原稿フォーマットがない...

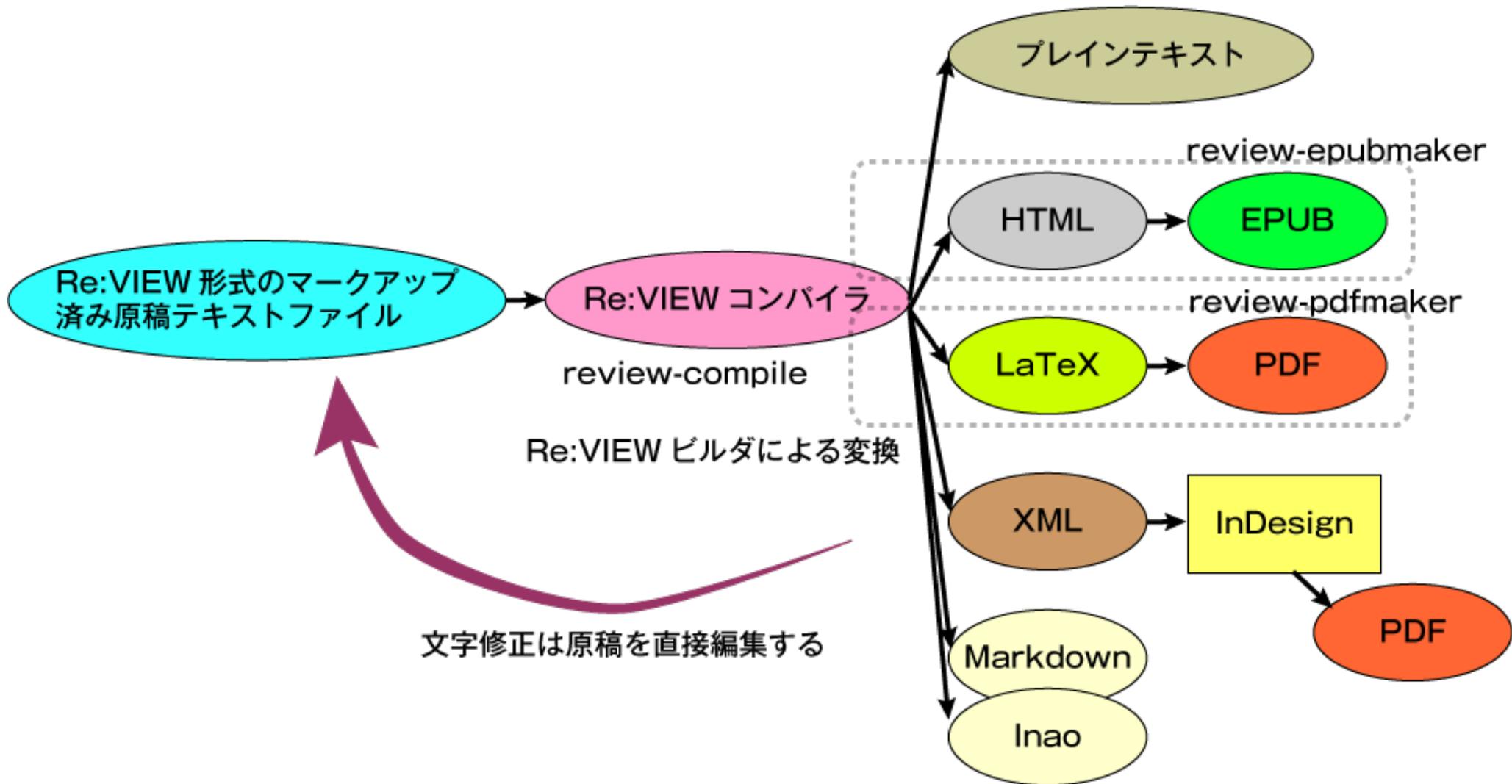
結局、マスターデータとして使い続けられるような原稿形式が**なかった**

- 2007年当時。
- 今だとpandocがわりと良いように思える

# Re:VIEW (れびゅー)

- 私が書籍の編集を担当した**青木峰郎**さんが、ご自身の執筆環境としてRe:VIEWの原型を開発
- 簡易なマークアップながら、**相互参照**や**自動採番**の機能を備え、構造化されたHTMLや、テキスト、TeXなどに変換できる
- 私が開発を引き継ぎ、いろいろな書籍で利用できるように汎用化、InDesignでの利用やEPUB生成機能など
- 達人出版会、オライリー・ジャパン等での標準形式の1つに

# Re:VIEW



<https://github.com/kmuto/review>

# Re:VIEW原稿

ch02.re

CR/LF    Unicode (UTF-8)    ReVIEW

行末コード    エンコーディング    カラーリング

1 | = スプライトを作ろう

2

3 | ゲームの世界では本来、アセットパイプラインとは開発しているゲームのビジュアルなワークフローを指す。マニュアルでゲームのメディアフォルダにファイルをコピーするだけの単純明快なものから、複雑な自動化スクリプトを書いてアートワークを生成するものまで、さまざまある。Impactを使った開発では、ゲーム内で使用されるグラフィックはすべて、スプライト (sprite) として扱う。

4

5 | == スプライトとスプライトシート

6

7 | Impactでは、アートワークの表示とアニメーションは「スプライト」という形式で行う。スプライトは、ディスプレイに表示されるシングルビットマップイメージで、この場合、HTML5 Canvas要素になる。関連するスプライトは、整理しやすいように、スプライトシートと呼ばれる1つの画像にまとめることができる。

8

9 | //image[Figure2-1][これから作成するゲームのスプライトシートの例]

10 | //}

11

12 | @<img>{Figure2-1}は、メインキャラクターの動きに用いるすべてのビジュアルを含むスプライトシートだ。スプライトシートは複数の組み合わせで構成される。この例では、160×16ピクセルの大きさのスプライトシートに、それぞれ16×16ピクセルのスプライトシートの長さを16で割れば、スプライトが10個あることがわかる。アニメーションにスプライトシートを使う場合には、ゲームエディタのアニメーションセットの一部であることを指示する。Impactの例を下記に示す。

13

14 | //emlist{

15 | this.addAnim( 'idle', 1, [0] );

16 | this.addAnim( 'run', 0.07, [0,1,2,3,4,5] );

17 | this.addAnim( 'jump', 1, [9] );

18 | this.addAnim( 'fall', 0.4, [6,7] );

19 | //}

行: 1 / 119    文字: 0 / 4,888 (>0)

『初めてのHTML5ゲームプログラミング』 (2012, オライリー・ジャパン)



# EPUB変換



2章  
スプライトを作ろう

ゲームの世界では本来、アセットパイプラインとは開発しているゲームのビジュアルなワークフローを指す。マニュアルでゲームのメディアフォルダにファイルをコピーするだけの単純明快なものから、複雑な自動化スクリプトを書いてアートワークを生成するものまで、さまざまある。Impactを使った開発では、ゲーム内で使用されるグラフィックはすべて、スプライト (sprite) として扱う。

## 2.1 スプライトとスプライトシート

Impactでは、アートワークの表示とアニメーションは「スプライト」という形式で行う。スプライトは、ディスプレイに表示されるシングルビットマップイメージで、この場合、HTML5 Canvas要素になる。関連するスプライトは、整理しやすいように、スプライトシートと呼ばれる1つの画像にまとめることができる。

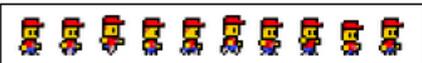


図2-1 これから作成するゲームのスプライトシートの例

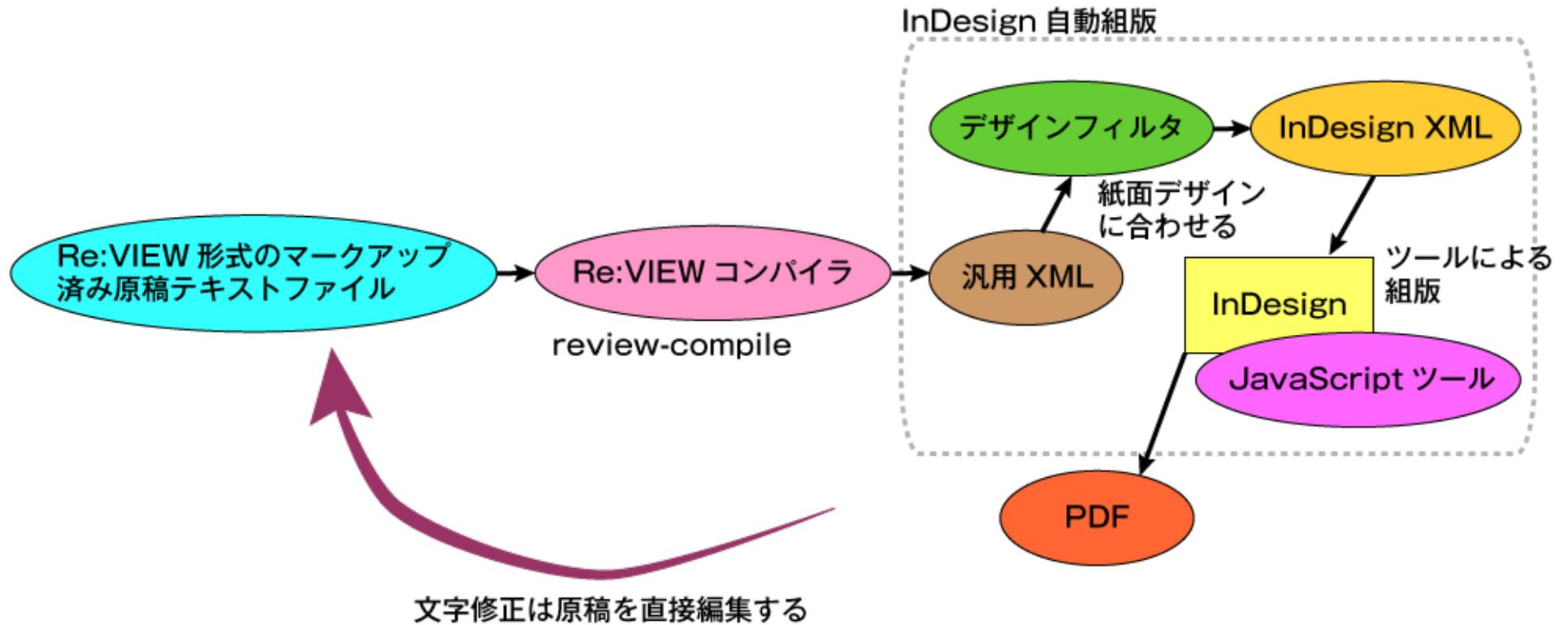
図2-1は、メインキャラクターの動きに用いるすべてのビジュアルを含むスプライトシートだ。スプライトシートは通常、計算しやすいサイズやアイテム数の組み合わせで構成される。この例では、160×16ピクセルの大きさのスプライトシートに、それぞれ16×16ピクセルのスプライトが並んでいる。つまり、スプライトシートの長さを16で割れば、スプライトが10個あることがわかる。アニメーションにスプライトシートを使う場合には、ゲームエンジンに対して、どのスプライトがどのアニメーションセットの一部であるかを指示する。Impactの例を下記に示す。

Chrome+Radiumで表示

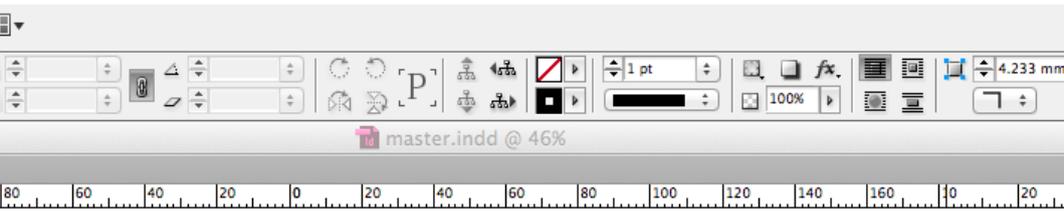
# Re:VIEW+InDesign自動組版

- ① Re:VIEW原稿から汎用XMLに変換
  - ② 汎用XMLを書籍の紙面デザインに合わせた加工フィルタに通し、InDesign XMLへ
  - ③ InDesignの紙面テンプレートに投入し、**手での組版を近似**する手法の独自のJavaScriptで紙面作成
  - ④ 文字修正はRe:VIEW原稿を編集し、必要なら**再度**InDesignで紙面作成
- <http://www.topstudio.co.jp/> に  
近日ムービー掲載予定

# Re:VIEW+InDesign自動組版



# Re:VIEW+InDesign自動組版

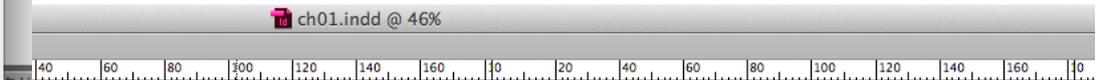


XMLドキュメント生成メニュー

- 実行のキャンセル
- ステージ1: ドキュメントの初期化とファイルの読み込み
- ステージ2: 特殊文字処理-章扉
- ステージ2.1: 自動チェック
- ステージ3: 表続き
- ステージ4: 背景ケイ
- ステージ5: タグを一時解除してPDF生成(万が一に備えInddは事前にコミットせよ!)
- サポートA: ページがオーバーフローしているときに実行
- サポートB-1: ページ参照用ラベルを取得
- サポートB-2: ページ参照を反映

OK

キャンセル



レベルエディタ

Impactの中で重畳される複数の1つが、レベルエディタであるWellmeisterだ。Impactプロジェクトのlibs/wellmeisterフォルダにある。レベルエディタについては次の章で詳しく見ていくが、ここでは機能と使い方を軽く紹介しておく。

プロジェクトのメインのルートへ移動し、wellmeister.htmlを開くことでレベルエディタを起動できる。起動すると、次のような画面が表示される。

図1-3 Wellmeisterの起動直後の画面。グリッド番号を操作するには、レイヤーを選択する

エディタを最初に起動したとき、空のentitled.jsdファイルが表示される。上掲のメニューには「Save (保存)」「Save As (別名で保存)」「New (新規作成)」「Load (ファイル読み込み)」などのコマンドがある。イメージの変更を表示できる。いちばん右にある大きな矢印は、レイヤーを表示させたり、隠すときに使う。その下にはマップのレイヤーが表示されている。デファルトで、エンティティのレイヤーがある。そこにプレイヤーやモンスター、その他のゲーム内の要素が配置される。新規レイヤーを追加するには、「Layers (レイヤー)」パネルの右にあるプラス記号をクリックすればよい。

レイヤーは、ビントソフトにあるスタンダードと同様に、あるステップにレベルタイプを編成できるようにするだけのものである。経路が作成するゲームのレベルが実際に編成せよのである場合は、レベルのタイプを背景、中景、前景などレイヤー分けしたり、衝突検知や経路を通知するためのレイヤーを作成する必要がある。ゲーム内で動く要素はすべて、エンティティレイヤーに属する。

編集と作成する前に、グラフィックを作成しなければならぬ。夜幕では、アセットパイプラインについての説明と、Impactで使えるグラフィックの作成方法について見ていく。

クラスとそのメソッドについての詳細は、Impactのドキュメンテーションにある、<http://impact.jp.com/documentation> を参照してほしい。

### 内部クラスの仕組み

従来のクラスベースの開発環境では、1つのクラス構造の中に別のクラスを入れることができる。これを内蔵クラスという。Impactにも独自の内部クラスがあり、単一のモジュールファイルに2つ以上のクラスを追加できる。

内蔵クラスの作成は、通常のクラス作成に似ているが、メインとなるクラスモジュールの最後尾に追加するという違いがある。内蔵クラスは継承にも対応している。手短かに説明するため、1つのモジュールに2つのクラスがある例を下記に示す。

```
01 ig.actor({
02   game.entitles.myclass'
03 })
04 .requires({
05   'impact.actity'
06 })
07 .define(function(){
08   Entity.prototype = ig.entitle.extend({
09     // フロントメソッドをここに追加する
10   });
11
12   Entity.prototype.inventClass = ig.entitle.extend({
13     // フロントメソッドをここに追加する
14   });
15 });
```

詳細は本書の後半で述べるが、このテクニックはコードを整理しておくのに非常に役立つ。

# InDesign自動組での表現の課題

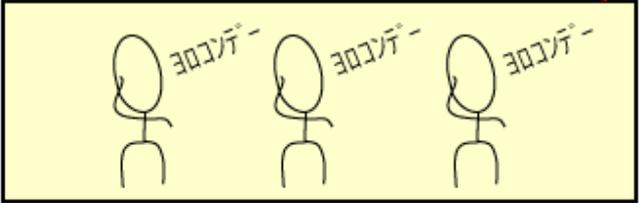
- **50冊以上**作ってきた結果
  - オライリー・ジャパンさまのシリーズ書のほか、各社の千差万別な紙面
  - おおむね満足いただけている模様
- どうにもならない制約も見えてきた
  - あきらめて「手でがんばる」ならなんとでもなるが...



# InD課題1: フロートの扱い

- 「図版を本文から独立して版面の上(あるいは下)に合わせたい」

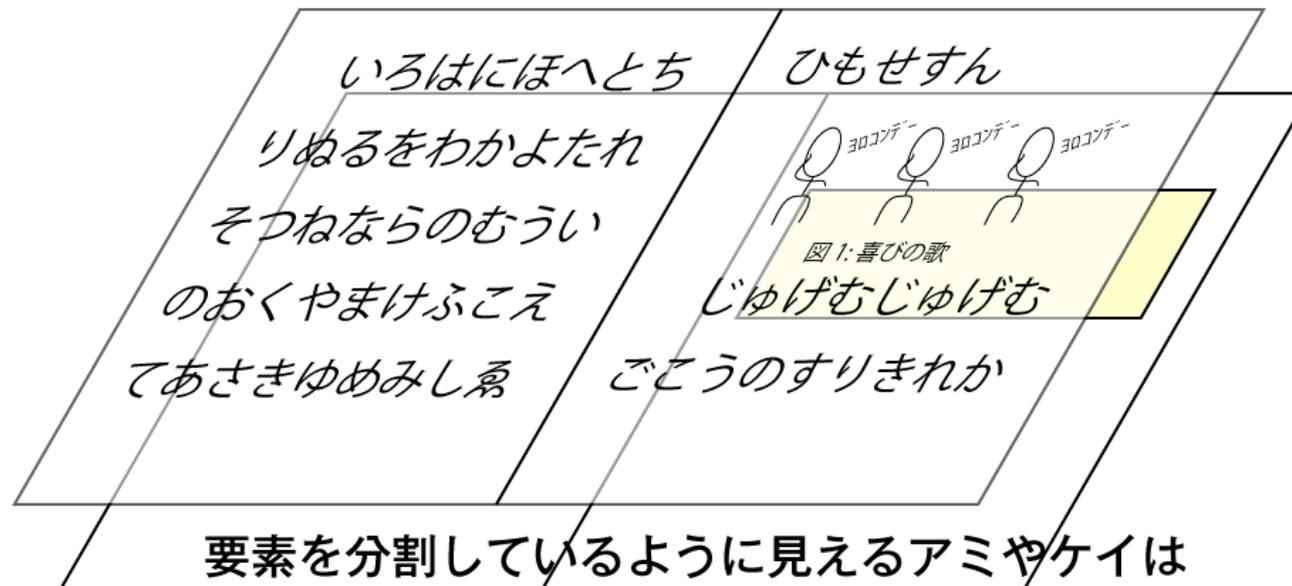
段落を切って、紙面上部合わせで図を配置

<p>いろはにほへとち りぬるをわかよたれ そつねならのむうい のおくやまけふこえ てあさきゆめみしゑ</p>	 <p>図1: 喜びの歌</p> <p>ひもせすん じゅげむじゅげむ ごこうのすりきれか</p>
---	---

# InD課題1: フロートの扱い

- InDesign自動組版は、コンテンツが一連のストリームになっていることが前提
  - フロートに見える囲みなどはレイヤー機能
  - ストリームからの切り出しは配置が困難

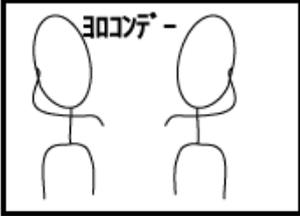
コンテンツは途切れない一連のストリーム



要素を分割しているように見えるアミヤケイは  
レイヤーで表現

# InD課題2: サイドバイサイド

- 「手順で左に番号と説明、右に図を入れて、関係が明示的にわかるようにしたい」

<p>①「某が紐を解いた程に、 わごりよ蓋を取らませ」</p>	
<p>②「それならば身共が蓋を取 らう程に、随分煽がしま せ」</p>	
<p>③「心得た」</p>	
<p>④「煽げ煽げ」 ⑤「煽ぐぞ煽ぐぞ」</p>	

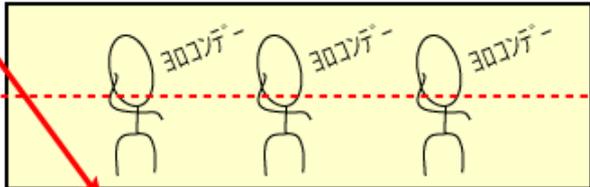
# InD課題2: サイドバイサイド

- 課題1と同様の理由で、一連のストリームにする必要がある
  - 「グルーピングしてインラインテキストフレームとして配置する」手動組版の王道が使えない
- 実際に配置しないと、箇条書き間にどの程度の空白行が必要かわからない
- 実際に配置してから迂闊にJavaScriptでいじるとクラッシュする

# InD課題3: バーティカルリズム

- 「見開きで左右のページの段落ベースラインが揃うようにしたい」

ベースラインのリズムに合わない要素。上下アキを適切に調整して後続をリズムに戻すことが望ましい

<p>いろはにほへとち りぬるをわかよたれ そつね puts "HELLO?" ならのむういのおく</p>	 <p>図1: 喜びの歌</p> <p>やまけふこえてあさ きゆめみしゑひもせ すん</p>
---	--

# InD課題3: バーティカルリズム

- 「配置してから自動で調整」は**困難**（不安定化）、「配置してから手で調整」は**きりが無い**
- プログラムコードや図表など、標準のベースラインと合わないサイズのものが多発すると、揃える労力が多大
  - 手動組版だと、そのような場面のたびに必死に調整していることが...

# AH Formatterによる解(1)

- InD課題1: フロート
  - float、-ah-float-move、-ah-float-x、-ah-float-yといったCSS属性で版面の上や下への合わせが可能
  - ただし、-ah-float-moveで無茶するとクラッシュ
  - top/bottom等は紙面の見栄えを見て恣意的に人間が判断することには変わりはない。
- ※HTMLの指定要素のclassパラメータを調整することになるので、手間はかかる

# AH Formatterによる解(2)

- InD課題2: サイドバイサイド
  - 課題1と同じフロートで対処できる
  - ブロック化してスタイル付け はHTML+CSSの得意分野

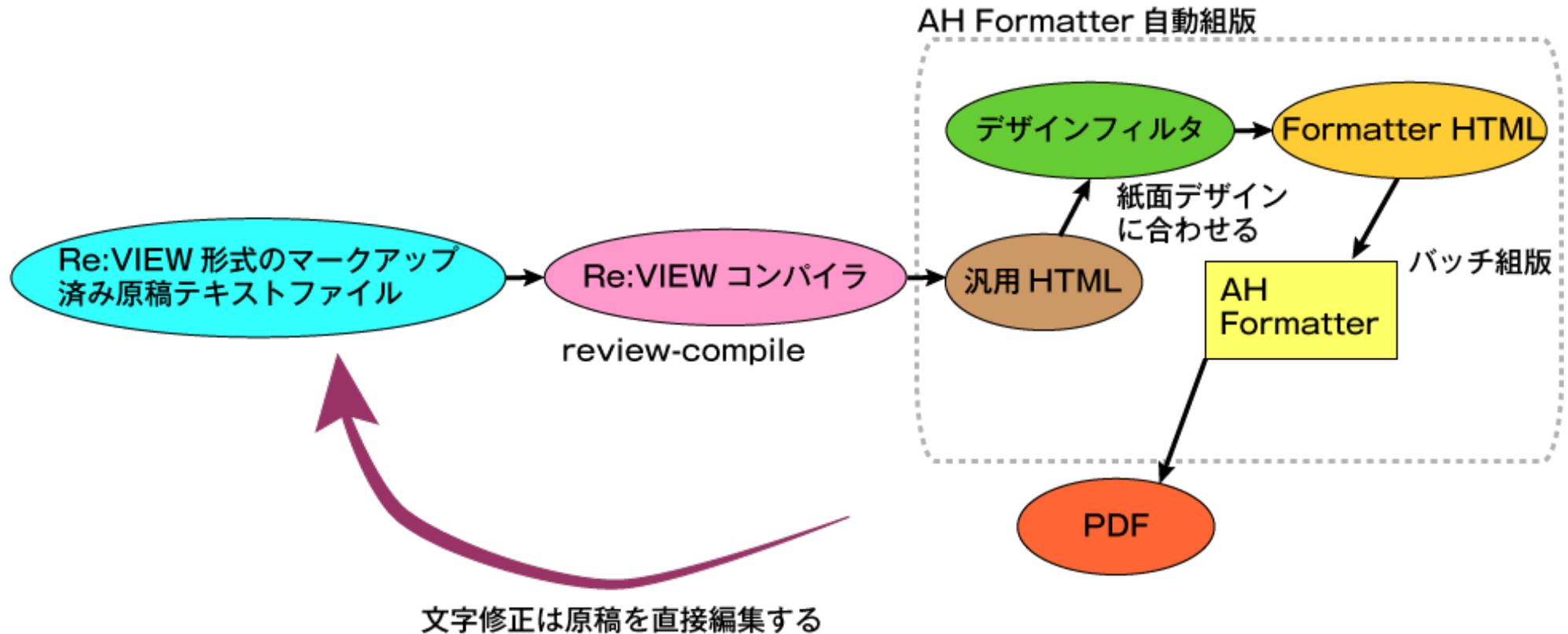
# AH Formatterによる解(3)

- InD課題3: バーティカルリズム
  - column-fill属性など
  - 標準で段落以外の要素間アキを自動調整しておおむね「**いい具合**」になっているように見える
- まだ向上の余地ありというアンテナハウス村上さんのtweetがあった

# Re:VIEW+AH Formatter

- ① Re:VIEW形式の原稿をマスターデータとするのは同じ
- ② Formatter用の汎用HTMLに変換
- ③ 紙面に合わせたHTML調整フィルタに通す
- ④ Formatterで組版。修正があれば再びRe:VIEW原稿を編集

# Re:VIEW+AH Formatter



# AH FormatterのためのHTML

- Re:VIEW標準のEPUB共通汎用HTMLではAH Formatterには不都合
  - 要素に付いているid属性の値を書籍全体で**固有**に  
`<h2 id="h1-2">`→`<h2 id="pre1-h1-2">`
  - 脚注を**インライン**化  
`<p>...<div class="footnote"><p class="footnote">脚注内容</p></div>...</p>`
  - **表見出し行**に**<thead>**追加  
`<table>`  
    **<thead>**`<tr><th>見出し</th><th>見出し</th></tr></thead>`  
    **<tbody>**`<tr>...</tr></tbody>`

# 組版例

サイズのスプライトシートに、それぞれ16×16ピクセルのスプライトが並んでいる。つまり、スプライトシートの長さを16で割れば、スプライトが10個あることがわかる。アニメーションにスプライトシートを使う場合には、ゲームエンジンに対して、どのスプライトがどのアニメーションセットの一部であるかを指示する。Impactの例を下記に示す。

```
this.addAnim( 'idle', 1, [0] );  
this.addAnim( 'run', 0.07, [0,1,2,3,4,5] );  
this.addAnim( 'jump', 1, [9] );  
this.addAnim( 'fall', 0.4, [6,7] );
```

この例でわかるように、スプライト0が停止状態で、0~5が動いているアニメーションだ。スプライトを使ったアニメーションの設定については、ゲームの設定を行うところで詳述する。

ここで注意しておきたいのは、JavaScriptの配列はゼロを基点とすることだ。一番目のスプライトは必ず0で、スプライトごとに値を1ずつ増やしていく。

スプライトシートは、階層のタイル画像などのように、アニメ化されていないグラフィックにも便利だ。

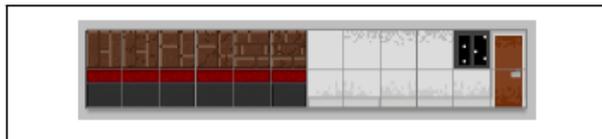


図2-2 階層を構成するスプライトはTileset (タイルセット)と呼ばれる

マニュアルでアニメーションを登録する代わりに、どのタイルが、設計している階層の壁、装飾、その他のアートワークであるかを、ゲームエンジンに指示すればよい。

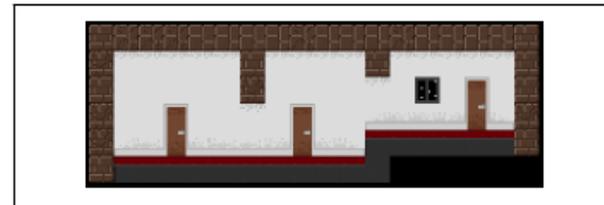


図2-3 ゲームにおけるタイルの用途を示すImpactのレベルエディタの画面

スプライトシートの基本をおさえたので、ゲーム開発において、これを実際に作成する方法について見ていこう。

ピクセルベースのゲームを開発する際にスプライトを作成する場合、素晴らしいイメージエディタがいくつかある。本書ではPhotoshopを使用した例を示すが、フリーソフトではGrafX2 (<http://code.google.com/p/grafx2/>) と GIMP (<http://www.gimp.org/>) を試すことをおすすめする。

## Photoshopのスク립トを使う

Photoshop上で自動化スク립トを作る方法はいくつかあるが、ここではJavaScriptを使った例に特化した。基本的には、PSDのレイヤ間をループし、スプライトを列に整理して単一ファイルに出力するスク립トを作成する。ゲームにおけるグラフィック生成で最も重要なポイントは、正し

# CSSの管理

- AH Formatter向けのCSS管理が**煩雑**になりやすい
  - 紙面のためのいくぶん複雑な構造
  - さまざまな書体や級数（文字サイズ）設定
  - AH Formatter独自のCSS属性が多いため、現時点ではテキストエディタで書くしかない

# CSSの管理

- **Compass**を導入（Rubyツール）
  - CSS上位互換のSASSあるいはSCSSで、変数やネスト、各種プラグインを使ってCSSを記述
  - Compassツールで標準のCSSに変換
- AH Formatter用の支援プラグインがあるとよいかも

# AHFormatterでの表現の課題

- 私の勉強不足なだけかも
- 「案件ごとに紙面デザインがばらばらなので作り込んでいられない」という背景

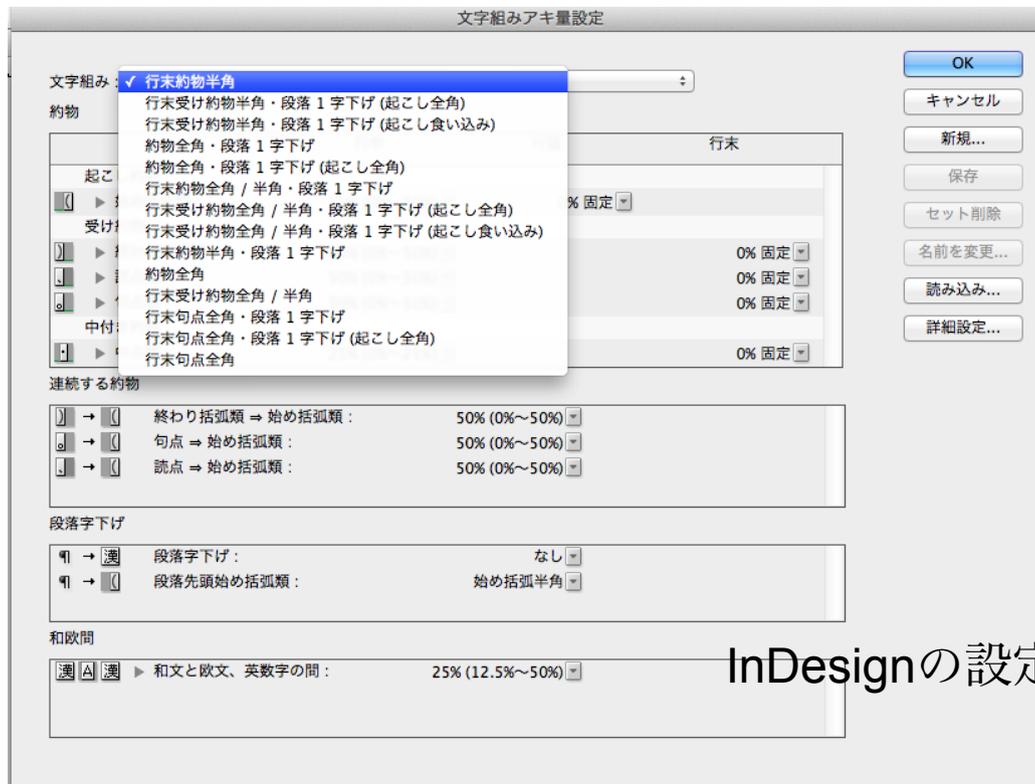
# AHF課題1: 文字組の設定

- 段落文字組み、禁則、和欧文アキ量設定などパラメータは相当数あるが...

-ah-punctuation-spacing, punctuation-trim, -ah-append-non-end-of-line-characters, -ah-append-non-starter-characters, -ah-except-non-end-of-line-characters, -ah-except-non-starter-characters, hanging-punctuation, -ah-auto-letter-spacing, -ah-avoid-window-words, -ah-justify-nbsp, -ah-kerning-mode, -ah-letter-spacing-side, text-autospace, -ah-text-autospace-width, text-justify-trim

# AHF課題1: 文字組の設定

- **プリセット**がないので設定が大変
- InDesignの設定に比べて柔軟性に欠ける



# AHF課題2: 見開きの概念

- 左ページ、右ページの概念はあるが、見開き（スプレッド）の考慮がない
- 製本ならではの概念

## 見開き前提のケイ線

10 | 1章 Impact 入門

SoundManagerは音声の読み込みと、Ig.MusicやIg.Soundのインスタンスに音声演奏制御を任す。SoundManagerのインスタンスは、Ig.SoundManagerのIg.main()関数により、自動的に作成される。

**System**  
Ig.Systemは、ルームの開始を終了を管理し、実行のゲームオブジェクト上で、.run()メソッドを呼び出す。Ig.Inputにいくつかのユーティリティ機能も提供し、管理にひと役買っている。

**Timer**  
Ig.Timerには2つのモードがある。1つは、現在時刻とタイマーの設定時刻（コンストラクター、もしくは、.set()で設定）の、.delta()を呼び出す方法。もう1つは、最後のコールから現在までのティックを、.tick()で求める方法。

クラスとそのスプレッドについての詳細は、Impactのドキュメンテーションを、<http://impactjs.com/docs/>から確認しては。

### 内部クラスの仕組み

従来のクラス（オブジェクト指向）では、1つのクラス構造の中に別のクラスを内包させることができる。これを内部クラスという。Impactにも独自の内部クラスがあり、単一のモジュールファイルに2つ以上のクラスを追加できる。

内部クラスの作成は、通常のクラス作成に似ているが、メインとなるクラスモジュールの最後尾に追加するという違いがある。内部クラスは継承にも対応している。手短かに説明するため、1つのモジュールに2つのクラスがある例を下記に示す。

```
01 Ig.module(  
02   game.entities.myclass'  
03 )  
04 .requires(  
05   'impact.entity'  
06 )  
07 .defines(function(){  
08   EntityClass = Ig.Entity.extend(  
09     //プロパティやメソッドをここに追加する  
10   );  
11  
12   EntityInnerClass = Ig.Entity.extend(  
13     //プロパティやメソッドをここに追加する  
14   );  
15 });
```

情報は本書の後半で述べるが、このテクニックはコードを整理しておくのに非常に役立つ。

### レベルエディタ

Impactの中でも重要される機能の1つが、レベルエディタであるWeltmeisterだ。Impactプロジェクトのlib/weltmeisterフォルダにある。レベルエディタについては次で詳しく見ていくが、ここでは機能と使い方を軽く紹介しておきたい。

プロジェクトのドメインのルートへ移動し、weltmeister.htmlを開くことでレベルエディタを起動できる。起動すると、次のような画面が表示される。



図 1-3 Weltmeisterの起動画面。グリッド編集を始めるには、レイヤーを選択する。

エディタを最初起動すると、空の「title.html」ファイルを開く。上左のメニューには「Save (保存)」「Save As (名前保存)」「New (新規作成)」「Load (ファイル読み込み)」などのコマンドがある。「Reload Images (イメージ読み込み)」を使うと、ファイルは完全には読み込まれず、マップイメージへの変更を表示できる。いちばん右にある大きな矢印は、レイヤーを表示させたり、隠すときに使う。その下にはマップのレイヤーが表示されている。デフォルトで、エンティティのレイヤーがある。そこにプレイヤーモンスター、その他のゲーム内の要素が配置される。新規レイヤーを追加するには、「Layers (レイヤー)」パネルの右にあるプラス記号をクリックすればよい。

レイヤーは、ペイントソフトにあるスタンツールと同様に、あるステージにレベルタイルを配置できるようにするだけのものである。読者が作成するゲームのレベルが非常に複雑なものである場合には、レベルのタイルを複製、中継、前後などにレイヤー分けしたり、衝突検知や詳細を追加するためのレイヤーを作成するとよいだろう。ゲーム内で動く要素はすべて、エンティティレイヤーに含まれる。

階層を作成する前に、グラフィックを作成しなければならぬ。本章では、アセットパイプラインについての説明と、Impactで使えるグラフィックの作成方法について見ていこう。

# AHF課題2: 見開きの概念

- Bad knowhow...

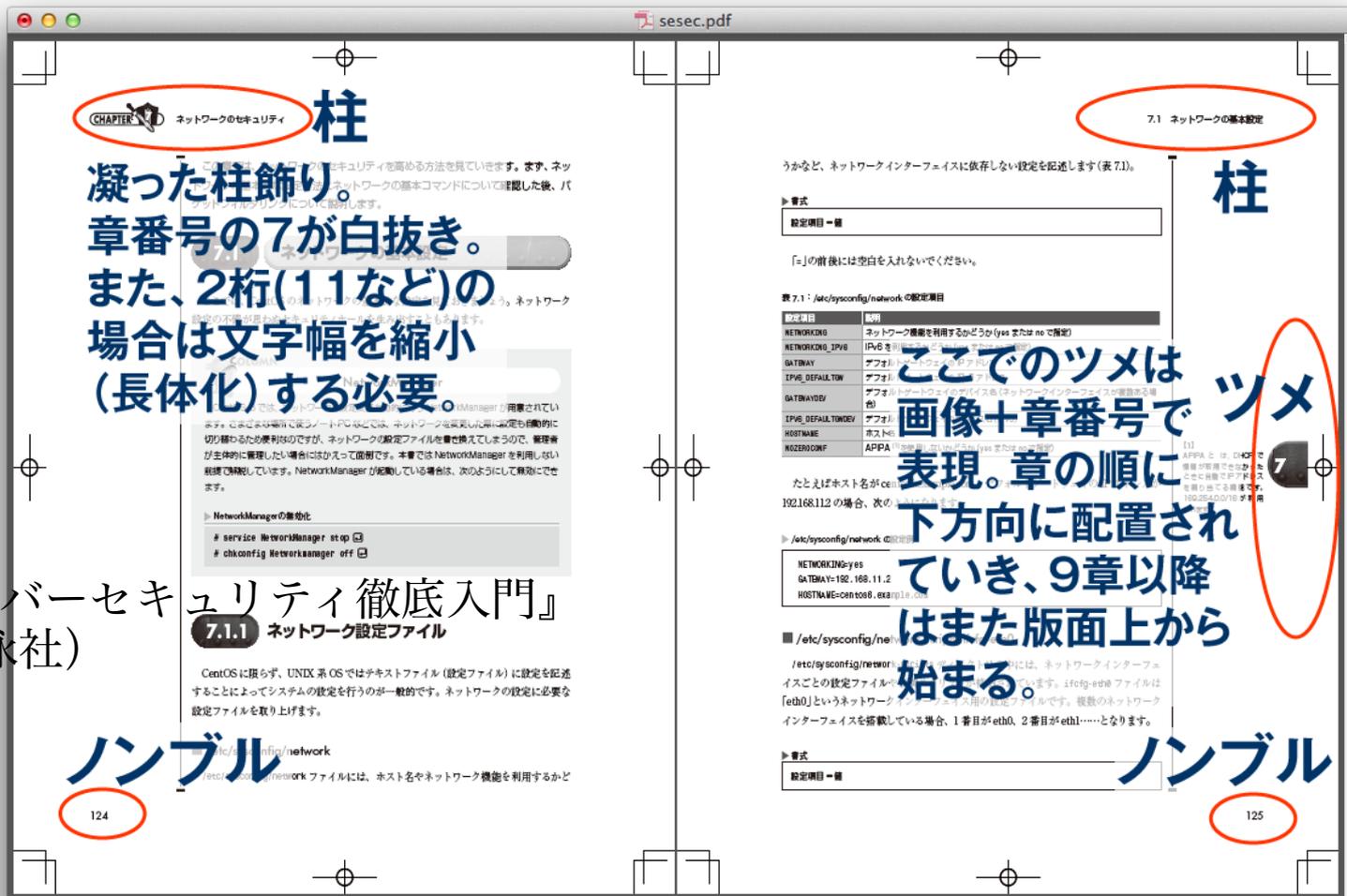
- 「紙面」背景に絶対位置指定の画像で配置

```
@page Chapter: right {  
    Background-repeat: no-repeat;  
    Background-image: url(line.svg);  
    Background-position: 0mm 16mm;  
}
```

- 紙面デザイナーや出版社にとっては見開きをもって1つのデザインと捉えることがある
- 関連して「章の最後のページ」も判断できない

# AHF課題3: 版面外の要素調整

- 柱、ツメ、ノンブル
- デザイナーの魅せドコロとなることも

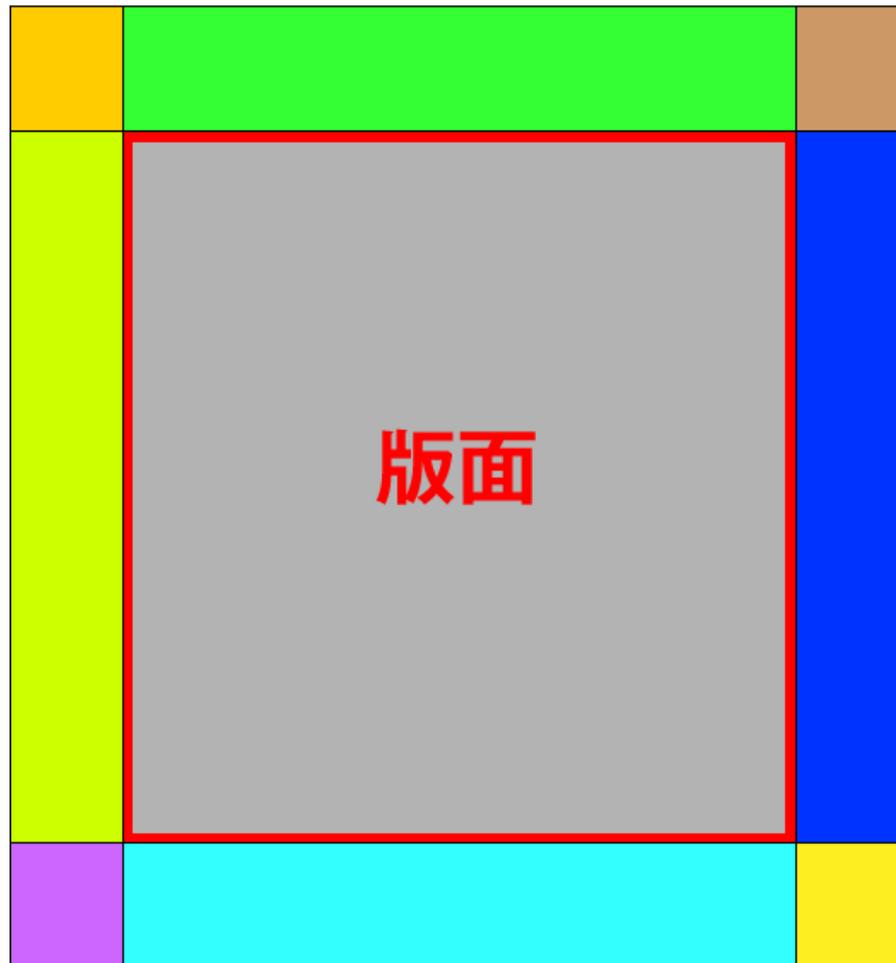


『Linuxサーバーセキュリティ徹底入門』  
(2013, 翔泳社)

# AHF課題3: 版面外の要素調整

- AH Formatter/CSSの紙面要素概念

紙面



# AHF課題3: 版面外の要素調整

- 紙書籍 固有の話
- 版面の中を操作するCSSは豊富
- しかし、**版面外**の指示があまり細かく書けない
- 「左上」などの既定位置をまたがりにくい
- content属性やstring関数の融通が効かない
- 改善予定？

# まとめ(1)

- InDesignでは手間のかかることが、AH Formatterなら**華麗にこなせる**ケースがある
  - 完全な自動化
  - フロート、サイドバイサイド、バーティカルリズム、etc...

## まとめ(2)

- AH Formatter/CSSで出版社水準を満たす技術書組版には、**まだ解決の必要な課題あり**
  - 組設定、見開き、版面外描画
  - これまでのアンテナハウスの活発な開発実績を見るに、**解決は遠くないと期待**

# まとめ(3)

- 書籍のマスタータータには、扱いやすい  
原稿フォーマットの導入を
  - 原稿からの一方向変換での組版/EPUB作成手法
  - HTMLを中心にするのは疑問
  - **Re:VIEW**形式オススメ  
<https://github.com/kmuto/review>

# まとめ

- InDesignでは手間のかかることが、AH Formatterなら**華麗にこなせる**ケースがある
- AH Formatter/CSSで出版社水準を満たす技術書組版には、**まだ解決の必要な課題あり**
- 書籍のマスターデータには、扱いやすい**原稿フォーマット**の導入を

本・メディアづくりのさまざまなご要望に「ワンストップ」でお応えします。  
株式会社トップスタジオ <http://www.topstudio.co.jp/>

# 補足

- 株式会社 トップスタジオ
  - <http://www.topstudio.co.jp/>
- Re:VIEW
  - <https://github.com/kmuto/review>
- 『はじめてのReVIEW』 (TechBooster)
  - <https://techbooster.booth.pm/items/12746>