

# 「電子書籍」関連の補足 2024

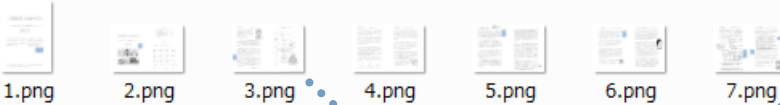
2024年7月 加藤厚

— WebP/AVIF を含む諸画像形式の検討と固定型 ePub の作成など —

PNG の原画 ↓ と非可逆圧縮の例 : WebP ↓ AVIF ↓

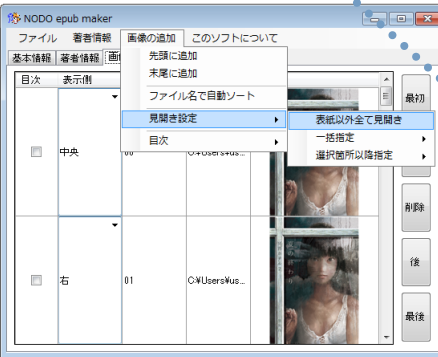


画像の出典 : 黒井白 (2019) 異界ホテル2 夏の終わり

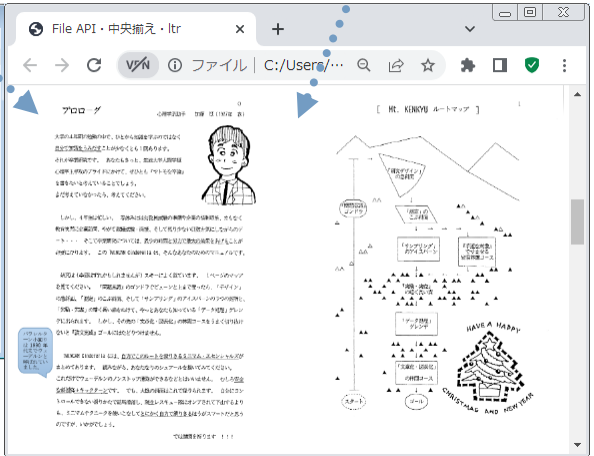


非可逆圧縮の WebP や AVIF も十分閲読可能ですが、明瞭さでは PNG に及びません。(02)

名前	サイズ
1.avif	106 KB
2.avif	393 KB
3.avif	386 KB
4.avif	510 KB
5.avif	606 KB
6.avif	240 KB
7.avif	734 KB



NODO epub maker なら見開きを一括設定できます。(03)



CSS と HTML5 です。

選択画像の一覧を表示する処理です。

```

<title>File API · center · rtl</title>
<style>div{text-align:center;direction:rtl}</style>
<input type="file" accept="image/*" multiple onchange="view(this)">
</div>
<script>
function view(elem,output='') {
  Array.from(elem.files).map((file)=>{
    const blobUrl=window.URL.createObjectURL(file)
    output+=`<img src=${blobUrl}>`))
  elem.nextElementSibling.innerHTML=output
</script>

```

表紙以外が「見開き」なら script はより単純になります。(06と07)

## 目次

- 01 画像形式に久しぶりの新展開が進行中です。 . . . . . 1
- 02 WebP や AVIF の圧縮はどれくらい有効でしょう？ . . . . . 3
- 03 固定型 ePub 作成アプリは NODO がオススメです。 . . . . . 5
- 04 結局、画質重視なら PNG、サイズ重視なら AVIF…。 . . . . 7
- 05 まず script を右左両綴じ対応にしてみます。 . . . . . 9
- 06 別発想の見開き画像縦読み script を書いてみます。 . . . . 11
- 07 連番化不要 script に挑戦してみます。 . . . . . 13
- 08 やはり「餅は餅屋」かもしれませんが…。 . . . . . 15
- オマケ：（コメントを追加した）紙資料の固定型 ePub での net 公開例 17

### 【文献】

あちらこちらと彷徨う本冊子の内容中、最も「実用的」なのはこの後半（Web アプリによる PNG の圧縮と ePub 作成）でしょう。

小品には縦書きがよく似合う…かも（加藤厚 2024a）

<http://mmua.html.xdomain.jp/kato/pdf/tategaki.pdf>

昔の道具で窓からプチ家出（加藤厚 2023）

[http://mmua.html.xdomain.jp/kato/pdf/evading\\_p.o.pdf](http://mmua.html.xdomain.jp/kato/pdf/evading_p.o.pdf)

「電子書籍」関連の補足 2022（加藤厚 2022）

<http://mmua.html.xdomain.jp/kato/pdf/e-book2022.pdf>

簡単に楽しむ電子書籍（加藤厚 2017）

[http://mmua.html.xdomain.jp/kato/pdf/kantan\\_e-book.pdf](http://mmua.html.xdomain.jp/kato/pdf/kantan_e-book.pdf)

A 5 判見開きでの閲読を想定して作成した本冊子の内容は、筆者が Windows 7 を OS とする PC 上で確認した結果をまとめたものです。お気づきの点などありましたら [atsushi.kato.1958@gmail.com](mailto:atsushi.kato.1958@gmail.com) までお知らせ頂ければ嬉しく思います。

本冊子を含む PDF や ePub の「閲読」には以前より viewer20 をオススメしていますが、同アプリには「コピペ（copy & paste）不能」という制約があります（書店提供のアプリですから無理ありませんが…）。

ということで、コピペをしたい場合には本冊子のような PDF からならブラウザや SumatraPDF などを、DRM フリーの ePub などからなら Bibi などを使用しましょう。（↑次頁の用語集の CBZ/CBR ・ WebP の項などや p.2 ・ 15 でも言及）

[Ctrl]+[R]ごとに「開き」を左  
⇔右入れ替え (Key = Shift + M  
なら[Shift]+[M]ごとに左⇔右)。

```
Shortcuts [  
[  
  Cmd = CmdToggleMangaMode  
  Key = Ctrl + R  
]
```

## 【用語集】

[Ctrl]+[...]: [Ctrl]キーを押しながら他のキーを押す (07 の例: File API の[ファイル選択]からフォルダを開き任意の1画像をクリック選択の上[Ctrl]+[A]で全選択←[A]=All。上の□内は SumatraPDF で[Ctrl]+[R]などに機能を追加する例)。

AVIF (AV1 Image File Format) : 可逆/非可逆が選択可能で PNG から可逆で半分、非可逆なら1割近くまで圧縮可能な画像形式。主要ブラウザで表示可能だが、固定型 ePub 素材としては viewer20 も Bibi も未対応、画像書庫の場合については本冊子 01 の p.2 参照。

CBZ/CBR : (Comic Book Zip/Rar) : 画像書庫の拡張子を zip/rar から cbz/cbr に rename したファイル。ZipPla や viewer20 で鑑賞でき ver. 3.0~の SumatraPDF では鑑賞に加えて PDF 変換も可能です。

dpi (dot per inch) : 1 inch=2.54cm あたりの dot 数で示す解像度の単位。15.6 型ワイド液晶は縦 7.65inch なので画面解像度が 1366×768 なら約 100 dpi (768÷7.65≒100) のため、適切な表示には 100 dpi 以上、拡大表示を考慮すれば 200~300 dpi でのスキャン (画像としては縦 1000~3000 px 程度) がオススメです。

ePub : 国際電子出版フォーラム (IDPF) が策定した電子書籍の形式。リフロー (流動/文字) 型については加藤 (2024a)、フィックス (固定/画像) 型については本冊子 03 参照。

function(e) : e(event)=頁で起こること (キー押しなど) の処理。例: onkeydown=function(e){switch(e.keyCode){case 37:次();break;~}なら、キー押し=onkeydown の keyCode に応じ=switch、それが 37=[←]キーなら=case “次()” を実行して{~}から抜ける=break。

PNG (Portable Network Graphics) : イラスト向きのロスレス (可逆) 圧縮の形式。写真向きの jpg と比べるとサイズは大きめですが、滲まないの文字を含むコンテンツなどにもオススメです。

WebP : ePub 3.3 で対応が勧告された画像形式だが、固定型 ePub 素材としては viewer20 も Bibi も未対応。画像書庫の場合 ver. 2.4~の SumatraPDF なら単体で、ZipPla なら ifwebp.sph ↓ で鑑賞可能です (資料 1 参照)。<https://github.com/uyjulian/ifwebp/releases>

Web アプリ (Web Application=Web App) : 特定の OS (例: Windows) 上で作動するネイティブアプリ (Native App) とは異なり、ブラウザ上や (ブラウザを経由して) サーバ上で処理を行うアプリ。ブラウザ上で作動する拙作※の SQP や SPDA もこの一種です。

※<http://www.vector.co.jp/vpack/browse/person/an021211.html>

PDF の右開きは SumatraPDF-settings.txt の Shortcuts [~]内への Cmd と Key (本頁上右の例参照) の書き足しで可能。“sumatrapdf 右綴じ”でネット検索してみましょう。

## 01 画像形式に久しぶりの新展開が進行中です。

「簡単に楽しむ電子書籍」（加藤 2017）以来、いくつかの小冊子で取り上げてきた電子書籍に関する結論は概略以下のとおりでした：

①自室での閲読は A5 版（＝見開きが A4 版）程度の PDF※などを 15 型程度以上の画面の PC# で行うのが適切。（加藤 2017）

※長所：レイアウト確定なので頁が有意味。テキスト付 PDF なら検索も可能…。# 15 型≒A4 横なので画面≒見開き。PC は手が空くので tablet などより楽。

②小画面での閲読には流動（リフロー）型 ePub が適切。情報的内容には横書きが便利で、情緒的内容には縦書きが適切。（加藤 2024）

③日本語未対応の OS 上での閲読や必要な font がない場合は画像群＋script、画像書庫＋ビューア、固定型 ePub※が適切。（加藤 2022）

※05～06 の画像群を「オマケ」の方法で↓固定型 ePub 化した例。

[http://gojinya.web.fc2.com/bibi/?book=TC\\_C.epub](http://gojinya.web.fc2.com/bibi/?book=TC_C.epub)

「簡単に楽しむ電子書籍」でも触れたように、PDF の閲読と作成は現在では多くのアプリで可能です※。やや専門的な流動（リフロー）型 ePub の閲読と作成・公開についても「小品には縦書きがよく似合う…かも」（加藤 2024）で紹介した viewer20、Sigil や Calibre、Bibi などで容易に行えるようになってきました。

※筆者は主に viewer20 で閲読し、writer と CubePDF で作成しています。

そんな中、画像形式に久しぶりの新展開が起こっています。「写真は jpg で非可逆圧縮、イラストなどは png で可逆圧縮」という長年の常識※は現在も有効ですが、2010 年には jpg ・ png いずれと比べても 20～30% 小さくできる WebP を Google が、また 2019 年には WebP 以上の高圧縮が可能な AVIF を Alliance for Open Media (AOMedia) # が発表し、主要ブラウザはこれらの形式に対応済みです。

※jpg の発表は 1992 年、png の発表は 1996 年で、共に「前世紀」の技術。

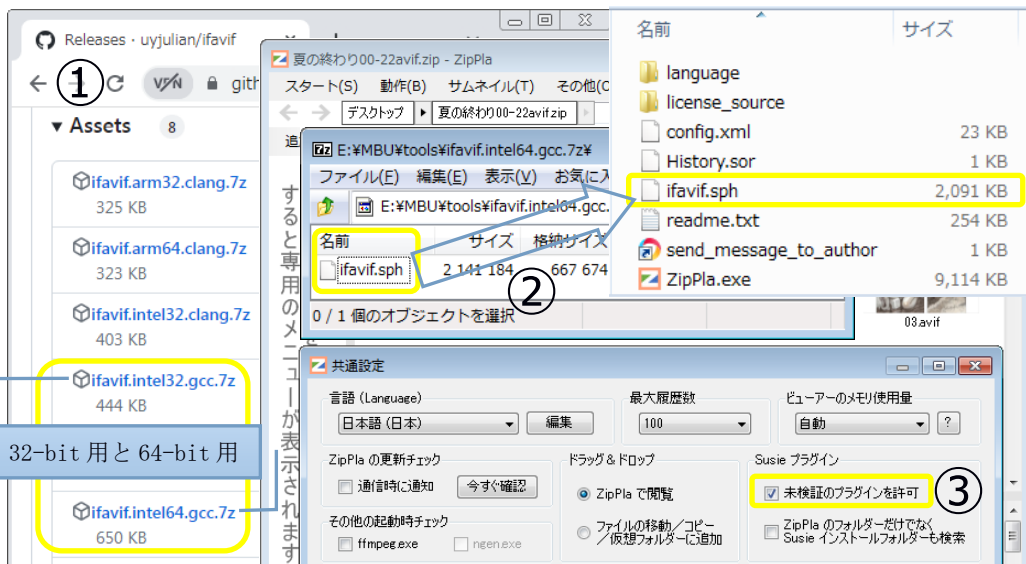
# Google、Amazon、Netflix、Intel、Microsoft、Mozilla などが構成する NPO。

他方、ePub 3.3 で WebP 対応が勧告されたのが 2022 年（AVIF 対応は未記載）のためアプリの対応は途上で、両新形式の画像書庫を鑑賞可能な Windows 7 対応ビューアは SumatraPDF※くらい、ZipPla などでは Susie プラグインが必要#なの現状です。

※WebP は ver.2.4～、AVIF は ver.3.5～（除 32-bit 版？）が対応。

#プラグイン（…sph）で画像書庫を鑑賞する方法については資料 1 参照。

新しい画像形式の導入目的が net を流れる情報の軽量（＝高速）化である以上、ブラウザから始まった変革はいずれリーダーやビューにも及ぶことでしょう。そこで、本冊子ではブラウザでの利用にも注目しつつ、各画像形式の特徴（圧縮効率と明瞭度など）とその活用法、及び固定型 ePub の作成と公開に関する追加的検討を行い、その成果などを紹介していきます。



- 資料 1 ①CPU に応じたプラグイン（…sph）を DL <https://github.com/uyjulian/ifavif/releases>  
②7-Zip で圧縮ファイル（…7z）を展開し ZipPla フォルダにプラグインを copy  
③「未承認のプラグインを許可」をチェックし（Zippla への D&D など）書庫を鑑賞

## 02 WebP や AVIF の圧縮はどれくらい有効でしょう？

まず、WebP と AVIF の圧縮効果の確認を目的として、PNG の画像群を両方式で圧縮してみました。素材としては加藤（2022）の 07 と 08 で作成した 23 枚の png 画像（計 12.1MB）を、変換アプリには上記 08 と同様 XnConvert※を使用しました。

※<https://www.xnview.com/en/xnconvert/#downloads>

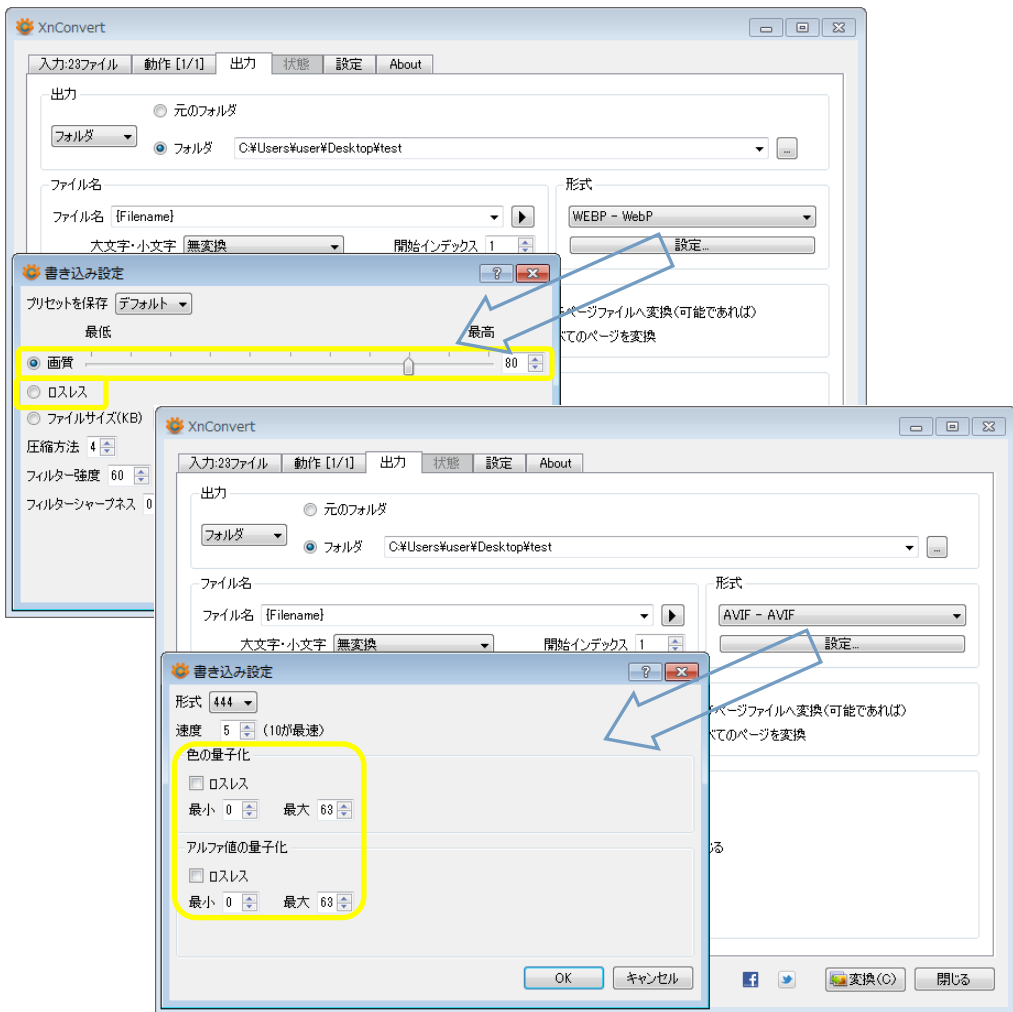
WebP で可逆と非可逆※、AVIF でも同様に圧縮したところ、出力された画像 23 枚の合計サイズは順に約 7.3 と 1.2MB、6.4 と 0.8MB でした。※WebP はデフォルトが非可逆＝画質 80%（資料 2 の上）で[ロスレス]を選択すると可逆、AVIF もデフォルトが非可逆（〃中）で「色の粒子化」などの[ロスレス]をチェックすると可逆になるようです。

特に非可逆の場合、圧縮後の画像サイズは WebP でも 1/10、AVIF では実に 1/15 になっており極めて強力な圧縮が達成されています。圧縮後の画質については、例として吹き出しに文字を含む 1 コマを資料 2 の下に示したように、PNG（63KB）の原画と比べ非可逆圧縮した WebP（9KB）及び AVIF（6KB）とも、細部の濃淡は若干失われるものの十分閲覧可能でした※。

※モチロン「明瞭さ」や「詳細さ」は PNG>WebP>AVIF ですが…。

では、WebP や AVIF の画像を素材とすればサイズが 1/10～15 の固定型 ePub が作れるか、ということ（残念ながら）筆者が試した範囲※では WebP や AVIF を有効な画像として認識・利用できる Windows 7 対応の編集アプリは見つかりませんでした（…10 や 11 用のアプリなら利用可能かも）。そこで、次節では原点に戻り、PNG や jpg を素材とする固定型 ePub の作成とその圧縮について検討します（なお、Web アプリによる PNG の高度圧縮についてはオマケの後半でも言及しています）。

※Sigil 1.9.30、Calibre 3.48 の ebook-edit、LeME 0.45.0、MangaZip 1.6、NODO epub maker 1.9.0.0…。



資料2 XnConvert による WebP (上) 及び AVIF (中) のデフォルト指定 ↑  
PNG の原画 ↓ と非可逆圧縮の例 : WebP ↓ AVIF ↓



画像の出典 : 黒井 白 (2019) 冥界ホテル2 夏の終わり

### 03 固定型 ePub 作成アプリは NODO がオススメ。

筆者オススメの固定型 ePub 作成アプリは NODO epub maker※です。その理由は（固定型が簡単に作れるもう一つのアプリである）LeME と比べてより自由だからです#。

※[https://vivibit.net/epub\\_maker/](https://vivibit.net/epub_maker/)

# LeME ではライセンスを購入しないと ePub の最終頁にクレジット画像が自動挿入されます。

基本情報などを入力／指定し、画像群をフォルダごと D&D したら[画像の追加]→[見開き設定]→[表紙以外全て見開き]で最初の画像のみ表紙として中央に、それ以降は（右綴じの場合）自動的に右→左→右→左…に表示する指定が一括で可能です（資料3の上）。

情報と画像の入力／指定が終わったら[ファイル]→[EPUB に保存]で指定した書名の ePub として保存されます。23 枚の画像群のサイズ 12.1MB に対し、出力された ePub のサイズも 12.1MB で固定型 ePub のサイズはその殆どが画像のそれであることが示されています。

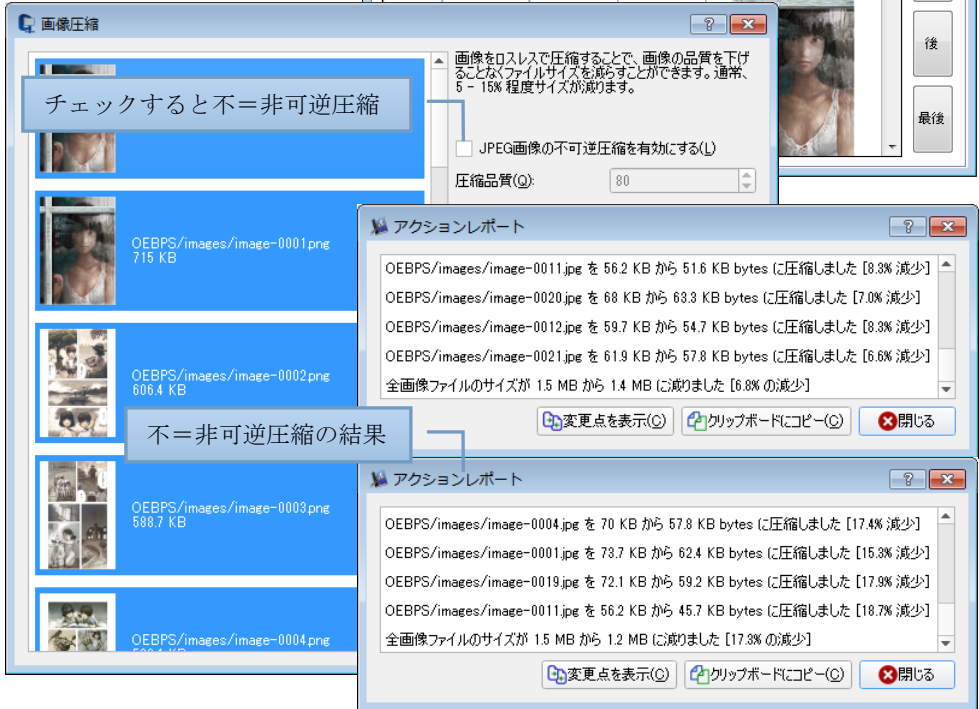
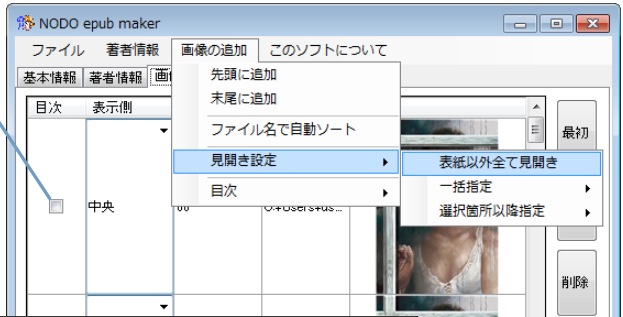
ここで加藤（2024）の 08 と同様に ebook-edit で可逆圧縮を試みましたが、圧縮過程でエラーが発生して別名保存できませんでした。そこで、XnConvert で画像を PNG から jpg に変換（画質はデフォルトの 80%）してみたところ、画像群 23 枚の合計サイズは 1.5MB で、1.2MB の WebP や 0.8MB の AVIF には及びませんがかなり小さくなりました。

この jpg を読み込んで NODO epub maker で作成した ePub のサイズは 1.5MB、それを ebook-edit でロスレス＝可逆圧縮すると 1.4MB、デフォルトの 80% で非可逆圧縮すると 1.3MB でした。次頁の下部に示したようにその画質は PNG > jpg = ロスレス > 不可逆で、jpg～でも十分に閲覧可能ですが「明瞭」ではありません。



チェックするとタイトル欄の内容が目次として示されます。

資料3 NODO epub maker  
での見開き一括設定  
(右) 及び jpg 画像の固定型  
ePub の ebook-edit による  
ロスレス/不可逆圧縮 (下)



【無劣化 = 品質重視の PNG とサイズ重視の他の圧縮画像との比較】

PNG > jpg = ロスレス > 不可逆



画像の出典：黒井 白 (2019) 冥界ホテル2 夏の終わり

## 04 結局、画質重視なら PNG、サイズ重視なら AVIF…。

結局、結論は There is no panacea. (万能薬はない) …「より小サイズでより明瞭」という“魔法”は見つかりませんでした。固定型 ePub の素材としては「明瞭さを重視するなら PNG、小サイズが必要ななら jpg + 可能であればそれらの可逆圧縮」が現実的な選択で、WebP や AVIF はまだ試行的対応の途上ようです。では「画像群を WebP や AVIF 対応のブラウザで提示するスクリプト」の可能性はどうでしょうか？

01 で触れたように、情報的内容なら横書き PDF を PC などの広い画面上で、情緒的内容なら縦書きの流動（リフロー）型 ePub をスマホから PC まで画面サイズ不問で利用すればよく、固定型 ePub や画像群自体の利用が必要な場面はそもそも限定的です。AVIF や WebP の画像書庫も ZipPla と Susie プラグインや SumatraPDF で鑑賞可能（資料 1）だし、加藤(2023, 2024)で触れたように ePub であれば流動型／固定型×ローカル／net 上のいずれの組合せでも Bibi で閲読可能です。従って、画像群自体の利用が必要な場面はそれらをブラウザでローカル※で鑑賞あるいは net 上で公開する時くらいでしょう。

※ZipPla で画像書庫を鑑賞できる以上「屋上屋を架す」に他なりません…。

ということで（必要性は小ですが）筆者の JavaScript の知識・技能の update と経験の拡大を目的として、加藤(2022)の「09 画像を見開きで表示する script の作成」を基礎としつつ「素材と場面に応じた極力簡明な script の作成」を以下にいくつか試みてみました※。

※“reinventing the wheel”（車輪の再発明）は「頭の体操」になりますし…。

まず上記の script を再確認しておく、その内容は資料 4 の下に示したとおりで「機能の多くをブラウザに丸投げすることで、わずか 10 行で（最新の画像形式の AVIF にも対応する）見開きリーダーが実現」（p. 17）されています（資料 4 の上の解説は同頁からの引用）。

「簡易」ということで、汎用性は期待せず、表示する画像ファイルの範囲（最小～最大）と種類（拡張子）は script 内に記述し、右開き（＝右とじ＝左側に読み進む・縦書やコミック）と左開きは別 script とします。まず、<div>～</div>の部分で最初の 2 画像（例では“左”という name(名称)で 2.png を、“右”という name で 1.png）を中央揃えで呈示しておき、<script>～</script>の部分で指定した最小～最大の範囲の画像の src（source：所在＝画像[番号]+拡張子）を新たに指定することで各名称（“左”と“右”）の内容＝画像ファイルを順次入れ替えます。入れ替えの指示は呈示画像のクリック（onclick）あるいは[←]/[→]キーの Code（37/39）を取得して“次”か“前”の function（関数）を呼び出して行います。

#### 資料 4 加藤（2022）で作成した script の解説（上）と内容（下）

```
<title>見開きリーダー（右開き用） 1-5@png</title>
<div style="text-align:center">
"></div>
<script>
最小=1;最大=5;拡張子=".png";頁=最小;画像=new Array()
for(i=最小-1;i<=最大;i++){画像[i]=new Image();画像[i].src=i+拡張子}
document.onkeydown=function(e){switch(e.keyCode){case 37:次();break;
case 39:前();break}}
function 次(){if(頁<最大)頁++;左.src=画像[頁].src;右.src=画像[頁-1].src}
function 前(){if(最小<頁)頁--;左.src=画像[頁+1].src;右.src=画像[頁].src}</script>
```

主ファイル名が連番で拡張子同一の右綴じ専用 script で、[←]/[→]キーやクリックで画像が入れ替わり、画像群の web 公開などに適切です。最大=5 を 23 とし、1～23.png の画像群※と同じ folder で開いて[←]を 18 回押すと 19～18.png の見開きが示されます（右図）。



※1.png と 23.png には表紙と裏表紙的に 2.png と同じ画像を使っています。

## 05 まず script を右左両綴じ対応にしてみます。

画像群を web 公開する場合、その形式と枚数、綴じ方向は様々なので、それらを群ごとに設定する資料 4 の script (加藤 2022) にも一定の用途はありそうです※。他方、ローカルでの画像鑑賞にはより汎用的な = 1 つで様々な画像群を鑑賞できる script が便利でしょう。

※画像の種類と連番は統一済み、画面フィットは「手動」とします。

[[Ctrl]+[+]/[-]などで↑]

そこでまず[L]キー (←Left=左) を 1 回押すとデフォルトの右綴じが左綴じに切り替わる script を考えてみました (資料 5 の上)。本質的な追加は水色背景の部分のみ※で、**case 76 = [L] キー押下で配列 : 画像の要素を逆順に入れ替え、頁 = 要素番号を最大 (←値は最大-1)**

つまり下の①の状態から②に切り替えます# ([ | ]は表示頁) :

頁 = 最大

頁 = 最小

①右綴じ : 15.png←14.png←13.png ··· 3.png←[2.png | 1.png]

②左綴じ : 1.png→[2.png | 3.png] ··· 13.png→14.png→15.png

※関数名を (綴じによって意味が変わる次と前から) **L** と **R** に変更しました。

# 最小と最大の--は関数 L と R で存在しない画像 (16 や 0) を指定しないため。

併せて[←][→]で二頁、[↑][↓]で一頁移動としました。

この 12 行の script で左綴じの画像 : 1~15.png (内容は「オマケ」のそれと同一) ※を開くと[→]では移動=改頁せず、[←]では改頁しますが見開きが左右逆です。そこで[L]キーを 1 回押下してから[→]すると正しい見開きでの左綴じ閲読が可能になります (資料 5 の右)。  
※04 の例と同様、1.png、2.png、15.png は調整用のダミーの表紙です。

ところで主ファイル名の連番化も 10 個以上になればアプリの出番、ここではくりまさん作成の E-Rename※を紹介しします。例えば Explorer の右クリックの[名前の変更] (用語集参照) で一括 rename した png ファイルの主ファイル名 TC (1)~TC (15) を 1~15 にしたい場合、まず  
※<https://kurima.sakura.ne.jp/download.html>

```

<title>←↑↓→・左綴じ切替可能</title>
<div style="text-align:center">"
"></div>
<script>
  最小=0;最大=14;拡張子=".png";頁=最小;画像=new Array ()
  for (i=最小-1;i<=最大;i++) {画像[i]=new Image ();画像[i].src=i+拡張子}
  document.onkeydown=function (e) {switch (e.keyCode) {
  case 37: L ();L ();break;case 38: L ();break;
  case 39: R ();R ();break;case 40: R ();break;
  case 76: 画像.reverse ();最小--;最大++;頁=最大;break}}
  function L () {if (頁<最大) 頁++;左.src=画像[頁].src;右.src=画像[頁-1].src}
  function R () {if (最小<頁) 頁--;左.src=画像[頁+1].src;右.src=画像[頁].src}</script>

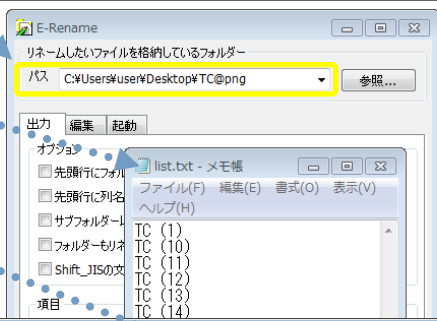
```

資料5 右綴じ→左綴じ切替可能 script  
 の一例 (上) と提示例 (右)、  
 E-Rename を使用して Explorer で一括  
 rename した主ファイル名からメモ帳な  
 どの置換機能で不要部分を削除→上書き  
 して連番化する過程の例示 (下)。



erename.exe のWクリックで起  
 動した E-Rename のパス欄に png  
 ファイルを保存したフォルダを  
 D&D して [新規編集] ボタンを押  
 します。主ファイル名一覧  
 のテキストファイル (list.  
 txt) がメモ帳で開かれるの  
 で [編集] → [置換] で不要な  
 "TC ("と")" を "" に [すべて  
 置換] = 削除したら [キャン  
 セル] し list.txt を [上書き  
 保存] (メモ帳は終了)。続  
 いて E-Rename の [適用] をク  
 リックすると主ファイル名が  
 1~15 = 連番のみになります。

- TC (1).png
- TC (2).png
- TC (3).png
- TC (4).png
- TC (5).png
- TC (6).png
- TC (7).png
- TC (8).png
- TC (9).png
- TC (10).png
- TC (11).png
- TC (12).png
- TC (13).png
- TC (14).png
- TC (15).png



"エクスプローラ 一括名前変更"  
 でネット検索してみましょう。

## 06 別発想の見開き画像縦読み script を書いてみます。

綴じ方向の切替は 05 の script で可能になりますが、考えてみれば見開き頁の左右やその移動方向が問題になるのは、元々見開きの二頁を左右に分割し、さらにその置き換えで「横方向」に拘っているからです。ここで頭を切り替えて、元々の見開き状態＝二頁分を一枚の画像とし、それらを（左や右ではなく）縦読み※すれば綴じ方向の切替も不要となってより簡単な script で実現できそうです。

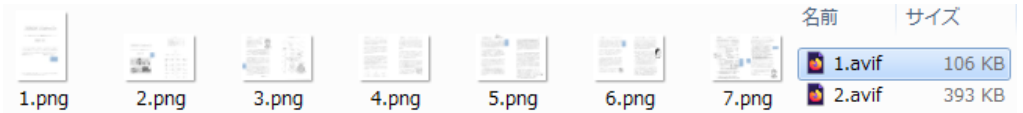
※4コマ漫画や Webtoon の形式です。

そこでまず、見開きの二頁を結合して一枚にします（…元々の画像が「見開き」なら結合は不要）。今回は ice.age さん※の「画像結合アプリ」を使用して 3.png～14.png を結合し、ダミーの表紙を 1.png、他を 2～7.png に rename した結果は資料 6 の上に示したとおりで、7枚の合計サイズは 9.1MB でした（綴じ方向の切替を行わないため調整用の 1.png と 15.png は不要＝不用）。これらの画像群を XnConvert (02 参照) で非可逆の AVIF に圧縮すると 2.9MB になりました。

※<http://hp.vector.co.jp/authors/VA058198/free/cmb/index.html>

提示画像が 1 枚なので name は“中”、移動は[↓][↑]（画像クリックで[↓]）、function も下()なら最大の直前まで頁++、上()なら最小の直前まで頁--という簡単な script で最低限の機能が実現できました（資料 6 の下の後半 7 行）。そこで、試みに画面フィットを手動※から自動にした加筆が 2～3 行目です。2 行目の {} 内は「画像の幅の上限を画面の 90% とし、高さは幅に連動させる」指定で、名称を（の後の）mxw として 3 行目の class="mxw" で<div></div>内に適用させます。結果、画像は常にブラウザ画面から食み出さず画面内に収まるようになります#（…2 行目と class="mxw" を削除すれば手動に戻ります）。※画像の大きさは[Ctrl]+[+]/[-]、ブラウザ画面の広さは[F11]や縁の D&D…。# が、同時に[Ctrl]+[+]で拡大しても画面幅の 90% を超えられなくなりますので、この機能は付けず手動調整のままの方が「柔軟」かもしれません。

PC 画面 (9:16) と比べて A4 見開きは縦長 (1:√2) なので左右に余白を取らないと上下が食み出ます。



資料6 表紙以外の画像を二頁見開きに結合した png (上:計 9.1MB)  
 それらを非可逆圧縮した AVIF とそのサイズ (右:計 2.9MB)  
 見開き画像の縦読み (+画像の画面自動フィット機能) script の一例  
 及び同 script による avif 画像の表示例 (下:7.avif を表示中)

名前	サイズ
1.avif	106 KB
2.avif	393 KB
3.avif	386 KB
4.avif	510 KB
5.avif	606 KB
6.avif	240 KB
7.avif	734 KB

蛇足的な  
自動フ  
ィット機能

最低限  
の機能

```
<title>2P 縦読み@AVIF1-7</title>
<style>.mxw img{max-width:90%;height:auto}</style>
<div class="mxw" style="text-align:center">
</div>
<script>最小=1;最大=7;拡張子=".avif";頁=最小;画像=new Array()
for(i=最小-1;i<最大;i++){画像[i]=new Image();画像[i].src=i+拡張子}
document.onkeydown=function(e){switch(e.keyCode){
case 40:下();break;case 38:上();break}}
function 下(){if(頁<最大)頁++;中.src=画像[頁].src}
function 上(){if(最小<頁)頁--;中.src=画像[頁].src}</script>
```

## 07 連番化不要 script に挑戦してみます。

04～06 の script は、筆者が 2000 年代※に習得した知識・技能を遣り繰りして書いたものです。他方、JavaScript は 2010 年代のライブラリ（例：jQuery）の隆盛を経て、近年では HTML や CSS などとの分離と統合を前提とする「複雑で高機能なもの」になっているようです。  
※Windows は XP、ブラウザは IE 6.0 や Netscape 6.2 だったころ。

とはいえ「便利な機能」は使ってみたい、特にローカルファイルを読み込み可能な HTML5 の File API は、これまでの script で必須だった連番化などが不要な点で大いに魅力的です。そこで収集・検討してみたその利用例中、最も感心したのが OpenCode への投稿でした※。  
※<https://open-code.tech/post-1691/#toc-mokuji-wrapper>

そのソースを加除修正した script は資料 7 の上に示したとおりで、表示領域の<div></div>の書式として<style></style>で要素の中央揃えと右→左提示 (rtl=右揃じ・右開き) を指定、<input>は画像ファイルのみ ("image/\*") で複数可 (multiple) とし、指定終了 (onchange) で関数 view を起動します。view は各画像ファイルの blobUrl (仮 URL) を src として追記した output を innerHTML として表示・・・という処理を行っているようです (・・・が筆者の理解を超えます) 。 ; -)

この script をブラウザで開き [Tab] で [ファイル選択] → 任意の 1 ファイルを選択して [Ctrl]+[A] で全選択 (資料 7 の右・・・用語集も参照) → [開く] で画面の上から下及び (見開きの画像 2 枚以上が収まる幅なら) 右から左※に画像が提示されます。見開きの 2 枚が収まるように画像の倍率とブラウザ画面の幅・高さを調整した例は資料 7 の下の左と右に示したとおりで [↓]/[PgDn] や [↑]/[PgUp] など次や前の見開きに進めば「連番化不要で画像数・拡張子不問の鑑賞」が可能です。  
※"rtl"指定の結果です。"ltr"にするか direction 指定なしなら要素の提示方向は 左→右=左揃じ・左開きになります (資料 7 の下左は ltr ・下右は rtl) 。





## 08 やはり「餅は餅屋」かもしれないが…。

AVIF なども含む画像群の鑑賞を主目的とする script を 05～07 で 1 つずつ考えてきました。05 のそれは（一般的な CBZ や CBR の展開で得られる）一頁一画像用で、デフォルトの右綴じ用を [L] キーで左綴じ用に切り替えられる点が長所です※。06 のそれは見開きの二頁一画像用 # で、自動フィット機能を除けば 7 行に収まる簡素さと（05 では必要な）調整用画像が不要な点が長所です。しかし、これらの script では事前に主ファイル名を連番のみとし、その最小・最大値と拡張子を画像群ごとに指定しておく必要があります。

※「表紙」の画像を最初と最後に追加しておくとおポロが出ません。  
# 元々が「見開き」のキャプチャ画像なら分割の手間がいりません。

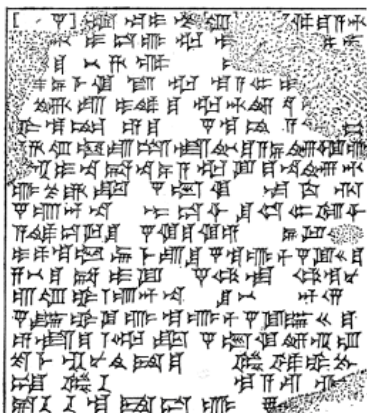
07 のそれは連番化と最小・最大値及び拡張子の事前指定が不要な点が画期的で、一画像が一頁でも二頁でも鑑賞できる点も便利です※。他方、縦長の巻物を [↓][↑] キーなどでスクロールする鑑賞方法のため、頁切替や頁めくりの快感の欠如は否めません。

※一頁一画像で右綴じの場合のみ rtl の指定が必要です（二頁一画像なら不要）。  
rtl 用と ltr 用の 2 script を作っておくのも一案でしょう。

加藤（2022）の 08 でも触れたように一頁一画像の PNG などなら PDF や CBZ にすれば viewer20 で鑑賞可能※だし、AVIF や WebP の書庫も SumatraPDF や ZipPla+プラグインで鑑賞可能です。従って、現実的には「餅は餅屋」、あれこれ工夫してブラウザで「ビューアもどき」の script を作ることは「頭の体操」以上の意義は無さそうです。他方、05～07 で試みたように 10 行程度の記述で「ビューアもどき」になるブラウザと script というのは大した、そして面白い技術だと率直に感じます（…「波打ち際で遊んでいる子ども」の感想に過ぎませんが）。ということで、File API などの理解と駆使、圧縮・展開ライブラリの活用といった“彼方の峰や雲”を遥かに眺めたところで今回の探索は※ZipPla なら二頁一画像でも（zip 圧縮するだけで）正しく鑑賞できます。

終了とします。人間、「憧れを無くしたら終わり」ですが、同時に「引き際の現実感覚」も（特に高齢者においては？）重要でしょう。

約 500 万年前、乾燥化によって東アフリカの森から草原に追い出された猿人は石器を工夫してなんとか生き延び、火を使い始めた原人を経て 30～20 万年前にはホモ・サピエンスに進化しました。その後、彼らは新天地を求めて出アフリカを試み、現在では地球上のあらゆる場所で相変わらず憧れたり悩んだりしています。資料 8 はスラウエシ島の洞窟で発見された世界最古=約 4 万 5 千年前のイボイノシシの絵と世界最古（約 5 千年前）の文字である楔形文字で綴られたギルガメシュ叙事詩（紀元前 1800 年頃成立）の冒頭部分です。話し言葉の成立時期にはまだ定説が無いようですが、30～20 万年の現生人類の歴史の 98% に文字は無く、80% には絵すら無かったのかもしれませんが。蔡倫が実用的な紙を和帝に献上（105 年）、グーテンベルクが四十二行聖書を印刷（1455 年）、エジソンが電気照明会社を設立（1878 年）、嶋正利が世界初の CPU 4004 を開発（1971 年）…これらの画期的進歩は全てホモ・サピエンスの歴史の最後の 1% の出来事です。現在これらを当り前に使い、それを遥かに上回る環境（PC・ネット…）を享受している我々は、この先一体どこへ、そしてどこまで行くのでしょうかネ？



資料 8 世界最古の具象的な壁画※（上）  
※<https://www.bbc.com/japanese/55661511>  
楔形文字で綴られたギルガメシュ叙事詩（右）

## オマケ：（コメントを追加した）紙資料の固定型 ePub での net 公開例

05～07 で左綴じの画像群の例として使用した B5 版 12 頁の印刷資料  
※を固定型 ePub にして公開する過程をメモしておきます。

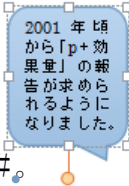
※1987 年に助手（現在の助教）としてワープロ専用機で作成した卒論ガイド。

①複合機で 300 dpi※でスキャンし、明瞭で滲まない PNG に画像化。  
※用語集でも触れたように 300 dpi で十分です（300 dpi のスキャンで B5 一頁は 2150×3035、A4 一頁は 2480×3508 の画像になります）。

②PNG 画像を JTrim※などで傾き修正し、併せてノイズなどを除去。  
※[イメージ]→[任意角度回転]で 0.1 度単位の回転ができます。

<https://www.woodybells.com/jtrim.html>

③Writer の文書に画像を貼り付け、ダミーの表紙と（図形→の吹き出しの）注釈を追加。

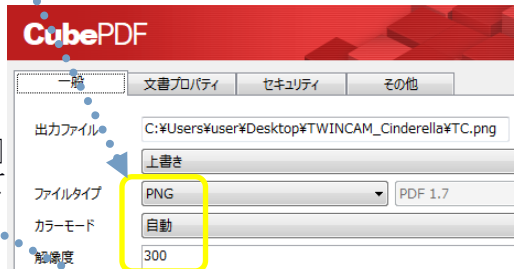


⑩ 図では、基本的には目的変数を縦軸に、説明変数・説明変数が環境要因と個人要因でその交互作用環境要因（例えば教授法）の方を横軸にとり、+を記号（○、●、等）で図中に示すとわかりや

文章は、書いて2～3日後に読み直し、推敲する  
とても良くなります。

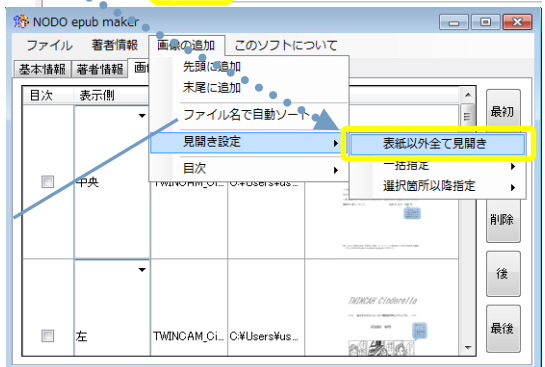
それから「統計的に有意」ということの意味と限界の  
のとか、pの値とかも考え合わせて考察しましょう

④画像出力も可能な CubePDF※  
で 300 dpi の PNG として出力#。  
※<https://www.cube-soft.jp/cubepdf/>  
#画像 13 枚で計 10.0MB。  
↑2479×3508 px



⑤NODO epub maker (03 参照) で  
[基本情報]の「ページ送り順」  
を[左から右]※、[見開き設定]  
を[表紙以外全て見開き]にして  
[ファイル]→[EPUB に保存]で  
保存場所とファイル名を指定  
して ePub 出力#。

※「左綴じ・左開き」のため。  
#ePub は (zip 圧縮の結果)  
画像群よりやや小さくなります  
(10.0MB→9.2MB)。

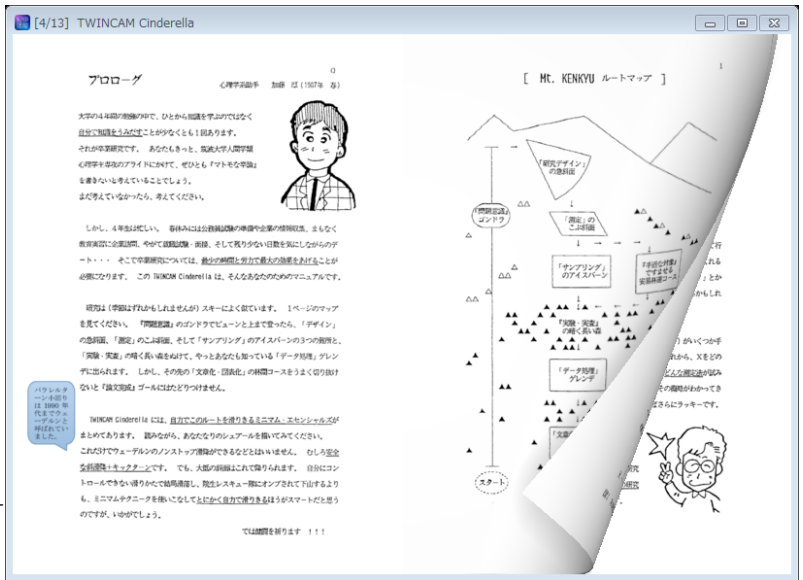


ファイル名の連番が同じ桁数 (例：  
01, 02, 03・10, 11, 12) なら自動整  
列します。桁数が異なる (例：1,  
2, 3・10, 11, 12) と順番が不正にな  
るのでこの項目で整列させます。

⑥viewer20  
でePub  
を開くと  
頁めくり  
効果付で  
閲読でき  
ます。→

⑦Bibi※経  
由でURL  
を指定す  
るとbibi-  
bookshelf  
フォルダ  
にULしたePubをブラウザでonline閲読できます。↓

※<https://bibi.epub.link/> Net公開の方法などについては加藤(2024a)の用語集と09を参照。ちなみに、下の表示の場合 bibi¥presets内のdefault.jsで"use-slider"(及び"resume-from-last-position")で"no"を指定しています。



CubePDF や NODO epub maker のように OS 上で作動するアプリに加えブラウザ上／経由で作動する Web アプリも充実してきました。そこで「300 dpi のままより小さい（例えば 3 MB 未満の）ePub を作成する」という課題を後者で追求してみました。

④の画像群計 13 枚（10MB）を Web アプリの [iloveimg](https://www.iloveimg.com)※で圧縮したところ 2.8MB まで圧縮でき、見かけ上の劣化もホボありませんでした。

※<https://www.iloveimg.com/ja/compress-image/compress-png>



これらの PNG 画像から NODO epub maker で作成した ePub のサイズは 2.7MB で既に 3 MB 未満でしたが「固定型 ePub の作成も Web アプリで…」という方針に従い急急如律令さん作成の「漫画の EPUB 3 作成ツール」※を使ってみました。ページ方向、表題、著者などを指定／入力し、表紙画像（今回は TC-01.png）と本文＝画像ファイル（今回は TC-02～13.png）を [ファイル選択] ボタンで選択して（画像順が不正なら [名前順に整理] で整えた後） [保存] ボタンを押すと PC のダウンロードフォルダに ePub ファイルが保存されます #。「目次の編集」の [追加] ボタンなどで項目名（例では「ルートマップ」）と画像（TC-05.png）を設定すると Navigation（次頁下右）の編集が可能です（…ePub のサイズは少し大きくなります）。

※<https://kyukyunyorituryo.github.io/FixedEpub3JS/>（右頁上部参照）

# [http://gojinya.web.fc2.com/bibi/?book=TC\\_C.epub](http://gojinya.web.fc2.com/bibi/?book=TC_C.epub)

Bibi で公開されている ePub は `<a href="URL/bibi-bookshelf/FILE名.epub">DL</a>`（次頁の※参照）という html ファイルを作成してブラウザで開き、DL（←例…文字列は任意）に設定されたリンクを右クリック→[名前を付けてリンク先を保存…]で DL できます。

AVIF や WebP が素材の固定型 ePub も作成可 (ただし viewer20 や Bibi では表示不能)。

FixedLayout Epub3.js

本の形式

本の綴じ方向

ページ方向  右から左  左から右

パネルビュー

拡大したときの移動方向  漫画 (通常)  四コマ

本の情報の入力

書名

TWINCAM Cinderella

著者 1

加藤厚

著者 2 (任意)

著者名 2 (任意)

言語選択

日本語

表紙画像

ファイルを選択 TC-01.png

画像の幅

2048

画像の高さ

2898

本文ファイルの読み込み

ファイル選択 12 ファイル

目次の編集

目次の選択

Cover

表紙

TC-03.png

目次

TC-05.png

ルートマップ

追加 削除

ファイルの出力

保存

クリックで拡大表示。

必要に応じて追加/削除します。

Bibi は index.html へのショートカットを作成して「お気に入り」に (D&D で) 登録すればスタートメニューから起動できます。

※URL と FILE 名は特定の指定が必要。  
特定の指定の一例：

<a href="http://gojinya.web.fc2.com/bibi-bookshelf/TC\_C.epub">DL</a>

DL.htm

新しいタブで開く

新しいウィンドウで開く

シークレットウィンドウで開く

名前を付けてリンク先を保存...

リンクのアドレスをコピー

検証

Navigation

表紙

目次

ルートマップ

TWINCAM Cinderella

加藤厚

Bibi